Simulation of a Complete Shock Tunnel Using Parallel Computer Codes

Thesis by

Richard J. Goozée, BE (Hons)

Submitted for the Degree of
Doctor of Philosophy

Division of Mechanical Engineering
The University of Queensland
Brisbane, Australia

2003
(Submitted April 14, 2003)

Statement of Originality

The work presented in this thesis is, to the best of my knowledge and belief, original, except as acknowledged in the text. The material has not been submitted, either in whole or in part, for a degree at this or any other university.

Richard J. Goozée

Abstract

Impulse facilities provide a unique capability in being able to reproduce aspects of the hypersonic flight environment in the laboratory. The usefulness of these facilities is limited by non-ideal aspects of their operation and by significant gaps in our understanding of these flows. In this thesis, simulations have been performed of a reflected shock tunnel facility, operated at the University of Queensland, with the aim of providing a better understanding of the flow through these facilities. In particular, the analysis of the simulations focuses on the premature contamination of the test flow with driver gas and the generation of the high levels of noise experienced in the test flow. The substantial computational effort required by these calculations has, in the past, meant that only parts of a facility have be modelled in any one simulation. The assumptions associated with modelling only part of the facility has meant that these simulations have not been successful in predicting either driver gas contamination or noise levels.

The multi-block Computational Fluid Dynamics (CFD) code MB_CNS, which is based on a finite-volume formulation of the compressible Navier-Stokes equations was used in the calculations. As an alternative numerical technique, Smoothed Particle Hydrodynamics (SPH) was investigated for advantages that it may provide in modelling flows involving gaseous interfaces. It was determined that this technique is limited in its applicability to shock tunnel flows due to problems including the treatment of solid boundary conditions.

The computational requirements of simulating a complete shock tunnel have been approached with the use of large parallel supercomputers. Both MB_CNS and the SPH code were parallelised using the shared memory approach of OpenMP and the distributed memory approach of MPI. The performance of the parallel codes are examined on various computers, including the APAC National Facility and the QPSF Facility, located at the University of Queensland.

The shock wave induced deformation of bubbles, of both a light and heavy gas, is examined as a test case for the shock tunnel modelling. This case has experimental photographs that can be compared directly with the simulations. These simulations demonstrate that MB_CNS can accurately model the shock induced instability and deformation of interfaces between different gases.

The simulations presented here assume an axisymmetric flow and mesh the complete facility, from the driver section to the dump-tank. By doing this, the current simulations eliminate many of the assumptions previously made in shock tunnel simulations. The simulations incorporate an iris-based model of the rupture mechanics of the primary diaphragm, an ideal secondary diaphragm and account for turbulence in the shock tube boundary layer with the Baldwin-Lomax eddy viscosity model. Supporting these simulations are three sets of experiments, with different operating characteristics, that were conducted by Dr. D. Buttsworth in the Drummond Tunnel facility. The results from these experiments are used as validation for the simulations.

Through the shock tunnel simulations, a better understanding of the mechanisms leading to driver gas contamination has been developed. It has been shown that the contamination is driven by a complex interaction between the reflected shock and the incident gases, which result in the generation of vorticity in the driver gas. In the tailored case that was simulated, a vortex ring was shown to form at the head of the driver gas, which moved along the centreline of the shock tube into the test flow resulting in the observed contamination. Due to the complex, transient nature of this process, it could not be predicted using simplified models, or even numerical simulations using only a part of a facility. The instability resulting from the interaction between the reflected shock and the contact surface is driven by the same mechanisms that are studied in the simulations of the shock wave interaction with the bubbles.

It is shown that the simulations performed in this thesis can provide an estimate of the noise levels experienced in the test flows produced by shock tunnels. This estimation is possible since the simulations include some of the noise producing mechanisms occurring throughout the facility. The generation of this noise is demonstrated, along with its propagation into the test flow. The noise levels measured in the simulated test flow were in agreement with noise levels measured in the test flow of the Drummond Tunnel.

Acknowledgements

I would first of all like to thank my wonderful parents for their continuous and loving support. This thesis would definitely not have been possible without having you both there.

I would definitely like to thank my supervisor Dr. Peter Jacobs who's knowledge and experience with everything that has gone into this thesis has made such a difference. A few minutes of advice and guidance would answer any questions that I had, often after days of searching myself.

Thanks to my sister Justine, who has always been there for me as a great friend and, in particular, thanks for all of the advice with this thesis.

Tack så mycket to Emma Mosesson for the support and encouragement that has been so valuable to me.

Thanks to Dr. David Buttsworth for providing the experimental data and for providing lots of very important help as well.

I would also like to thank my good friends in the Department throughout this: James Faddy, Dr. Chris Craddock, Ben Stewart, Vince Wheatley, Dr. Kevin Austin, Rowan Gollan, Judy Odam, Michael Scott and Scott Rowan. I would like to say thanks for something to do with this thesis, but I should just say thanks for the good time that I've had!

I would like to thank Steve Kimball for all of his help with a considerable amount of computing and networking support. I need to thank Valerie, Rose and Elizabeth in the office for helping me through some stressful times. I would also like to thank everyone at the APAC National Facility and Martin Nicholls at the University of Queensland for their time in providing the supercomputing facilities used in this thesis.

Richard J. Goozée

Contents

List of Tables

Nomenclature

A	: area, m^2
a	: sound speed, m/s
C_p, C_v	: specific heats, J/(kg.K)
E	: total specific energy, J/kg
e	: specific internal energy, J/kg
$\overline{F}$	: array of flux terms
f	: species mass fraction
h	: smoothing length
$\hat{i}, \hat{j}$	: unit vectors for the cartesian coordinates
k	: coefficient of thermal conductivity
m	: particle mass
M	: Mach number
n	: direction cosine
$\hat{n}, \hat{p}$	: unit vectors for the cell interface
P	: point in the (x, y)-plane
p	: pressure, Pa
Q	: array of source terms
q	: heat flux, W/m^2
R	: gas constant, J/(kg.K)
r	: radial coordinate, m
r, s	: normalised coordinates
S	: control surface of the cell
T	: temperature, degree K
t	: time, s; independent parameter for the Bezier curves
U	: array of conserved quantities
u, v	: velocity, m/s
V	: cell volume, m^3
W	: kernel smoothing function
x, y, z	: cartesian coordinates, m
ρ	: density, kg/m^3
μ, λ	: first and second coefficients of viscosity, Pa.s
γ	: ratio of specific heats

Subscripts, Superscripts

i	: inviscid
is	: species index
n	: normal to the cell interface
p	: tangent to the cell interface
v	: viscous
x, y, z	: coordinate directions
a, b	: particle indices

CHAPTER 1

Introduction

Over the past 100 years of aeronautics, the reproduction of the flight environment in wind tunnels has played a fundamental role in the development of flight vehicles. Wind tunnels have been used in the development of vehicles across a range of flight conditions and flow regimes, including the Wright brothers flyer, the Bell X-1 and the Space Shuttle Orbiter [85]. The most demanding flow regimes are those relating to hypersonic flight. These high speed flows are experienced by vehicles travelling to and from orbit. Research in this regime is required for the development of future forms of aerospace propulsion, such as the Scramjet engine [181]. Physical experiments on these vehicles can be carried out using models placed in ground based impulse facilities such as *shock tunnels* and *expansion tubes*. Characterised by high test flow speeds, of up to 13 km/s, and short test times, of the order of 10 μs to 10 ms, impulse facilities are uniquely capable of generating high energy flows. Continuous wind tunnels are not viable for use in the hypersonic flow regime due to their high power requirements and the prolonged exposure of tunnel material to the extreme temperatures that are generated during the processing of the test gas [105].

Griffith et al. [85] described the technical problems experienced during the first re-entry flight of the Space Shuttle Orbiter. One problem resulted from a significant difference between the predictions of hypersonic pitching moment measured in wind tunnel tests of sub-scale models and the moment experienced in actual flight. As a result of this difference, a body flap deflection of twice that predicted was required in order to maintain the stability of the orbiter during re-entry. The design of the orbiter was based on data obtained in wind tunnels, which could not reproduce all of the real-gas effects present in the flight environment. Combining numerical simulation with wind tunnel test data provides an effective method of extrapolating the available test data to the design of vehicles. This approach was shown to be successful in further research conducted on Space Shuttle re-entry aerothermodynamics combining experimentation and numerical simulation [80, 241, 81]. Computer based simulations have an important role to play in the interpretation of wind tunnel data in the context of the flight environment.

Much of the hypersonic test data used in aerospace research is obtained in free-piston driven reflected shock tunnels. Full scale facilities around the world include: T4 (The University of Queensland) [41], T3 (Australian National University, Canberra) [178], T5 (GALCIT, California Institute of Technology) [83], HEG (DLR, Germany) [61], the Large Energy National Shock (LENS) tunnel (Cornell Aeronautical Laboratory) [34], Von Karman Gas Dynamics Facility (Arnold Engineering Design Center (AEDC), Tennessee) [6], and HIEST (NAL, Japan) [163]. Research in these facilities is often augmented with numerical simulations of the flows using Computational Fluid Dynamics (CFD). An important component of the research conducted relates to the relationship between numerical simulation and the physical experiments. Numerical simulations aid in the interpretation of experimental results and can be used to extend results obtained in impulse facilities to the real flight environment. Conversely, physical experiments in impulse facilities provide the data required to validate the numerical techniques used in aerospace vehicle design.

In this thesis, numerical simulations of the flow through the Drummond Tunnel facility have been developed. The Drummond Tunnel is a relatively low enthalpy reflected shock tunnel facility and is one of the impulse facilities operated within the Centre for Hypersonics at the University of Queensland [7, 54]. The simulations aim to provide a better understanding of the performance of shock tunnel facilities. In particular, the simulations focus on the premature contamination of the test flow with driver gas and the generation of the high levels of noise experienced in the test flow.

The shock tunnel simulations in this thesis use the multi-block CFD code MB_CNS [114]. It is based on a finite-volume formulation of the compressible Navier-Stokes equations. An alternative numerical technique, known as Smoothed Particle Hydrodynamics (SPH), was also investigated for the advantages that it may provide in modelling flows involving gaseous interfaces. A CFD code based on the SPH technique was developed in order to assess the abilities of the technique in modelling two dimensional compressible flow problems.

The substantial computational effort required by detailed simulations of shock tunnels has meant that only parts of a facility could be modelled in previous studies [43, 247, 240]. The simulations presented in this thesis use an axisymmetric mesh covering the whole facility, from the driver section to the dump-tank and, in doing so, eliminate many of the assumptions previously made in simulations of shock tunnels. The simulations incorporate an iris-based model of the rupture mechanics of the primary diaphragm, an ideal secondary diaphragm, and they also account for turbulence in the shock tube boundary layers. The computational requirements of solving transient flow fields on such large meshes have been approached with

the use of large parallel supercomputers. Both MB_CNS and the SPH code have been parallelised using the shared memory approach of OpenMP and the distributed memory approach of MPI. The performances of the parallel codes are examined on various computers, including the APAC National Facility and the QPSF Facility located at the University of Queensland.

In support of the current numerical work, three sets of experiments, with different operating characteristics, were conducted by Dr. D. R. Buttsworth in the Drummond Tunnel facility. Measurements recorded throughout these experiments can be compared directly with corresponding values from the simulations. In addition to this, the shock wave induced deformation of bubbles, of both a light and heavy gas, is examined as a multi-component gas test case for the CFD code MB_CNS.

1.1 Shock Tunnel Flow Characteristics

Although the shock tube concept was proposed in 1889 by Vieille, it was not until the 1940s that a group at Princeton University laid down the foundations of modern shock tube practice [98]. These facilities were operated by expanding a high pressure reservoir of driver gas into a lower pressure test gas, thereby forcing a shock wave through the test gas. This shock-accelerated gas, which is quasi-steady, is then passed through the test section. This straight through configuration could produce test flow stagnation temperatures of up to 600 K for typical durations of between 20 μs and 40 μs [203].

In 1950, at the Cornell Aeronautical Laboratory (CAL), the addition of a simple divergent nozzle to the end of a shock tube saw the creation of the first shock tunnel. By expanding the supersonic flow behind the shock wave through this diverging nozzle, the range of test Mach numbers was extended into the hypersonic range. The trade off for the higher Mach numbers was a reduction in test times.

The shock tunnel was developed further by adding a convergent section to the nozzle, thereby reflecting the shock back through the oncoming test gas. The reflected shock tunnel stagnates the test gas, forming a high temperature, high pressure reservoir, which can then be expanded through a nozzle to produce the test flow. This increased the energy of the gas in the nozzle reservoir, extending the range of test Mach numbers achievable to 24 and extending the test flow duration to 15 ms [203]. Operation in the tailored interface mode enabled the flow duration to be extended even further [248]. Further improvement in the operation of reflected shock tunnels was achieved through the addition of a free piston driver [219]. The free piston driver works by adiabatically compressing and heating the driver

gas, thereby increasing the shock Mach number that can be achieved in the shock processing of the test gas [220].

The development work that has been carried out on the designs of shock tunnel facilities has, over time, resulted in a significant increase in the range of test conditions available to experimenters; however, there remains significant concerns over the quality and usefulness of the test flows produced [105]. These flow quality issues manifest themselves in differences between the test flows produced in shock tunnels and the real flight environment. Further research into the operation of shock tunnel facilities is necessary to ascertain the causes of these differences and to develop methods of reducing their effect.

At the University of Queensland, the *Hyshot* test programme has been conducted in order to quantify the differences between the test flow produced in the T4 shock tunnel, and atmospheric flight conditions [108]. This test programme aims to compare the performance of the same prototype scramjet engine in the T4 shock tunnel and in atmospheric flight on board a Terrior-Orion Mk70 sounding rocket. Numerical modelling of shock tunnel flows can be used to approach the investigation of these differences from another perspective, that is by investigating the non-ideal processes that occur within shock tunnel facilities.

Shock tunnels, and impulse facilities in general, can only generate test flows of very short durations. Therefore, careful consideration of the experimental conditions are required in order to ensure that a steady flow can be established and maintained for a sufficient duration to take measurements [130]. Depending on the operating characteristics of the facility, the test flow may be limited by different mechanisms: pressure changes due to tailoring waves or the reflected expansion, or driver gas entering the test flow [212].

The test flows produced by high-performance reflected shock tunnels are also affected by thermochemical effects. This results from the high temperatures in the stagnated gas at the end of the tube, which is subsequently expanded to become the test flow. When this high temperature gas is expanded through the nozzle it is effectively frozen at nonequilibrium conditions. The Drummond Tunnel experimentation, and corresponding simulation, do not contain temperatures above 1600 K and so issues associated with these effects are not addressed in this thesis. These effects would otherwise need to be accounted for in the numerical simulation through the use of models for thermochemical nonequilibrium.

The quality of data obtained in impulse facilities for the purpose of studying high-speed transitional flows may be compromised by the strong influence of laminar-turbulent transition on the noise experienced in the test flow [172]. The test flow

noise levels are often more than an order of magnitude larger than flows experienced during flight [202]. Noise appears as fluctuations in the properties of the test flow from the mean flow values, which occurs throughout the test time. This noise is generated by non-ideal processes occurring throughout the operation of the facility, including the growth of turbulent boundary layers and reflected shock interaction processes, and propagates into the test flow.

The stagnation of the test gas with the reflected shock is an essential part of the operation of reflected shock tunnel facilities. Many of the problems experienced through the operation of these facilities result from the non-ideal interactions that occur during this process. The reflection of the shock from the contoured nozzle entrance results in the generation of flow fluctuations in the nozzle supply region and the generation of vorticity near the nozzle centreline [111]. As the reflected shock moves back upstream, the energy deficient boundary layer does not have enough energy to cross the normal shock, and instead boundary layer material builds up at the foot of the shock and is carried with it. This causes the flow to separate, and the shock to bifurcate into a lambda structure [141]. As this structure moves upstream, the flow through this bifurcated foot can be projected downstream relative to the flow which is stagnated by the normal shock [240]. Further upstream, the reflected shock reaches the incident contact surface. The subsequent interaction results in the generation of a significant amount of vorticity in the driver gas at the interface. In addition to introducing fluctuations to the nozzle supply region, these interaction processes also result in the jetting of driver gas towards the test flow.

The full potential of free piston driven reflected shock tunnels in high enthalpy operation has not been reached in practice due to the premature contamination of the test flow with driver gas [221]. As a result, a significant amount of research, both experimental and numerical, has been conducted into this phenomenon [228]. Skinner [212] obtained experimental evidence of driver gas contamination in the T4 shock tunnel using a mass spectrometer in the test flow. This data has proven difficult to interpret, although the experimental study was not supported by numerical simulations. The mechanisms which were believed to contribute to the premature contamination of the test flow with driver gas were outlined. Skinner concluded that the jetting of driver gas through the initial opening of the diaphragm was the driving mechanism and the reflected shock interaction processes were not thought to be responsible. The simulations performed in this thesis aim to investigate the mechanisms that lead to driver gas contamination.

1.2 Numerical Modelling of Shock Tunnels

If impulse facilities are to be used to their potential in the reproduction of hypersonic flight conditions, a greater understanding of their operation is required. Numerical simulations have the potential to significantly improve our understanding of shock tunnel flows, allowing the investigation of the causes of present limitations and the development of improvements to the facilities.

Computational modelling of these facilities provides a useful method for investigating the problems associated with the operation of shock tunnels since they are not subject to the experimental difficulties experienced with the real facilities. Such difficulties include the short flow durations and the extreme properties of the gases. In addition to this, numerical simulations can provide information on the flow throughout the whole facility, including the development of the flow along the inside length of the shock tube. In physical experiments, measurements can only be recorded at particular points in the flow where probes are located and, for practical reasons, only a limited number of these probes can be used in the flow in any experiment. Experimental results are complicated as the presence of the probes themselves has an effect on the flow and their measurements are affected by experimental noise. The simulations can aid in the interpretation of experimental results by providing information on other flow properties at the location of experimental probes and can be viewed in the absence of experimental noise.

Numerical simulations allow fine details of the operation of the shock tunnels to be investigated and, since numerical simulations are not limited by the physical arrangement of the facilities, they can be used to consider variation in the operating conditions outside of those already prescribed. They can also be used to investigate the effect of modifications to the facilities, thus reducing the associated expense and risk of experimental development [44].

Further, numerical simulations can be used to bridge the gap between the test flows produced by shock tunnels and the real flight environment. By producing a virtual shock tunnel, and thereby virtual test flows, experimental results can be compared with a simulated test flow, rather than an idealised representation. The differences between the simulated shock tunnel test flow and the simulated flight environment can then be identified and studied.

A demonstration of a virtual shock tunnel flow is shown in Figure 1.1. The upper half of the frame shows a shadograph image of an Apollo re-entry vehicle model during free flight in a wind tunnel at the Arnold Engineering Development Center. The lower half of the frame shows the same vehicle in a virtual shock tunnel test flow. The end section of the shock tunnel nozzle can be seen in the frames.

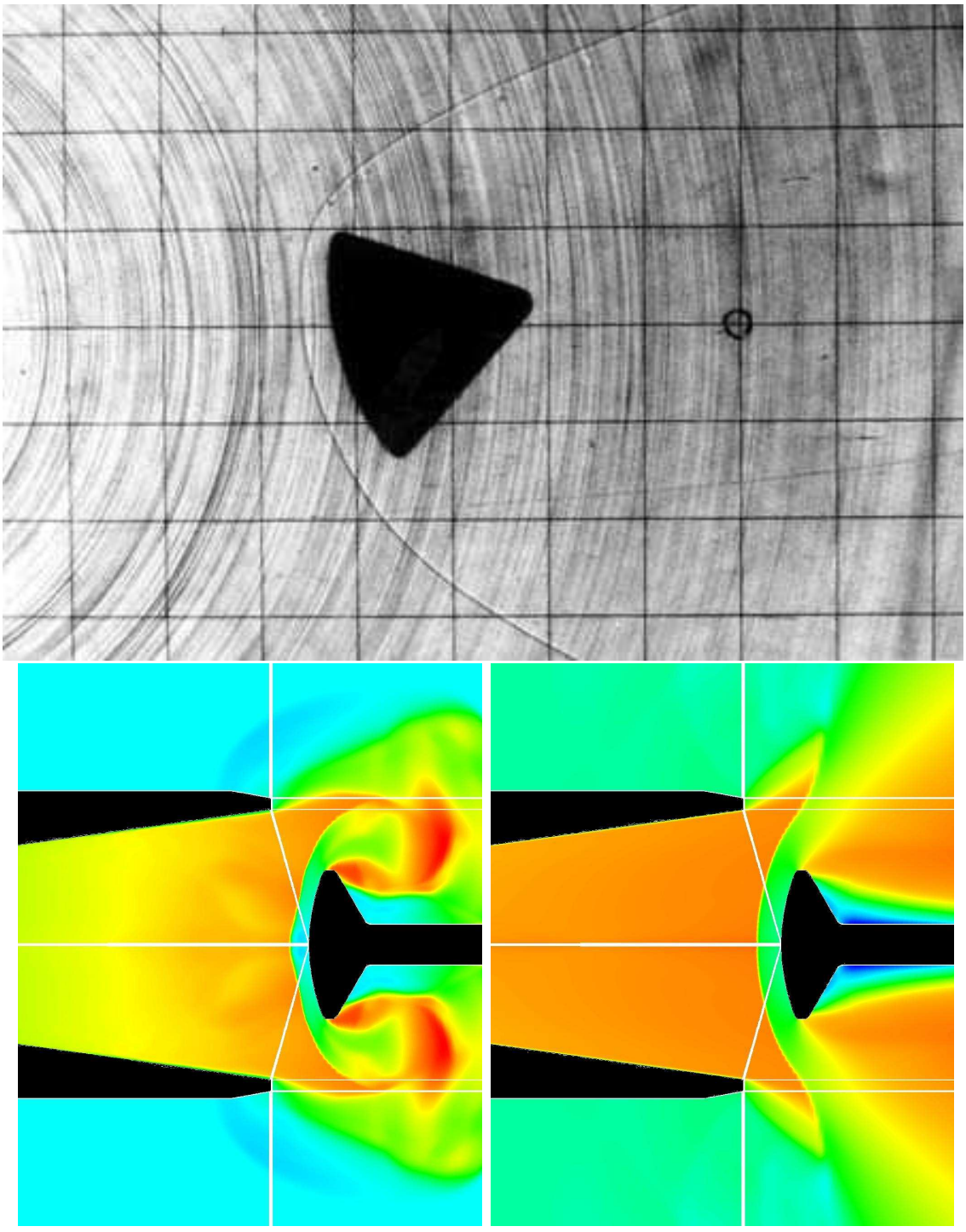

Figure 1.1: Demonstration of a virtual shock tunnel flow. The upper half of the figure shows shadograph image a model of an Apollo reentry vehicle in a wind tunnel (reproduced from AEDC Photo #67-0441 [6]) and the lower half of the figure shows a similar model in a virtual shock tunnel test flow. The left side shows the model during flow development and the right side shows the model during the steady test period.

Limitations of Previous Modelling

Detailed analytical and empirical based models of the flow of gases in impulse facilities have been developed. An outline of these models is provided in Sudani and Hornung [228]. These models have included attempts at the prediction of driver gas contamination, but they have never been successful across a range of operating conditions. The investigation of these facilities through more complex and detailed numerical simulation has been made viable through the combination of computers with sufficient capability and the development of suitable numerical algorithms. A significant volume of research has since been published based on the numerical simulation of shock tunnel facilities. To date, this modelling has predominantly been in one of two categories: quasi-one dimensional simulation and axisymmetric simulation. Full, three-dimensional flow simulation of a complete facility remains at the limit of present computing technology.

Quasi-one dimensional simulations model an entire shock tunnel facility, with the assumption that the flow properties vary only along the length of the tube. Mass loss models must be used to model the effect that the boundary layers have on the core flow [64]. They are computationally efficient and have been shown in previous studies to be effective in predicting the performance of shock tunnel facilities [113]; however, since the variation of flow properties across the radius of the facilities is neglected, they cannot be used to investigate fundamentally multi-dimensional flow features, including driver gas contamination and noise generation mechanisms.

Attempts at the axisymmetric simulation of reflected shock tunnels have typically only modelled particular parts of a facility and, in doing so, usually focused on one flow process in isolation. Typically the very end of the shock tube and the nozzle are modelled and an inflow condition derived from empirical relations is used at the upstream end of the flow domain [206, 246, 45]. The results obtained in the simulations are heavily dependent on these assumptions. In the past, this has been necessary due to the lack of computing power available.

A large proportion of axisymmetric simulations have aimed specifically at modelling the interaction of the reflected shock with the boundary layer, with the aim of interpreting its effect on driver gas contamination [246, 8]. These simulations have been unable to reproduce the experimental measurements of driver gas contamination with any certainty. The complexity of the driving mechanisms and the dependence on the flow development through the whole facility are thought to be responsible for this.

Numerical Modelling in This Thesis

The simulations presented in this thesis extend the numerical study of shock tunnel facilities by modelling a complete facility, from driver section to dump tank. By making fewer assumptions, the simulations are expected to be more effective at predicting the changes in the performance of the facility caused by modifications to the operating conditions or the facility itself.

Simulation Using MB_CNS

The main shock tunnel simulations performed in this thesis use the multi-block CFD code, MB_CNS [114]. This code is based on a finite-volume formulation of the compressible Navier-Stokes equations. It has a shock-capturing capability through the use of a limited reconstruction scheme and an adaptive flux calculator. The adaptive flux calculator switches from AUSMDV [238] to the Equilibrium Flux Method (EFM) [139] where large compressive velocity gradients are detected. The numerical details of this code are discussed in Chapter 3.

There are a number of additions to MB_CNS that have been included in this study to specifically address the simulation of shock tunnel facilities. The Baldwin-Lomax eddy viscosity model [11] was used to model the effect of turbulence in the boundary layers. Coefficients used in the model were first obtained from the literature [121] and then were adjusted using the experimental results. Multiple gas components were modelled by solving additional conservation equations for each of the components. An iris based diaphragm rupture model was included for the primary diaphragm, which defined the opening according to the profile measured experimentally by Rothkopf and Low (1974) [197]. An ideally rupturing secondary diaphragm was also included.

Simulation Using SPH

The characteristics of the contact surface between the driver and driven gases can have a significant effect on the operation of a shock tunnel. This interface has complicated properties and can be susceptible to hydrodynamic instabilities. A computational technique that is based on the Lagrangian description of fluid flow may provide advantages in modelling these flows, where fluid interfaces are important. As a result, the Smoothed Particle Hydrodynamics (SPH) method, being Lagrangian in nature, may be effective in shock tunnel modelling. In this method fluid interfaces are maintained through the movement of the elements of fluid, which are referred to as particles. The particles move with the flow and so the particles representing the fluid on one side of the interface will remain on that side.

Quasi-one dimensional CFD codes based on the Lagrangian description, such as L1d [111], have been shown to be effective in shock tunnel simulation and are computationally efficient. The investigation of the SPH method for its applicability to the simulation of shock tunnel flows aimed to extend these qualities to multi-dimensional modelling. The method is suitable to implementation in parallel and so the code will also be discussed further in the context of parallel computing. Even though a lot of effort was expended on the the SPH code, it was found to not be suitable for the main shock tunnel simulations discussed in Chapter 7.

Solution of the Flow Field in Parallel

In order to model an entire shock tunnel facility with sufficiently fine detail, large computational meshes are required. The fine mesh simulations of the complete Drummond Tunnel facility used in Chapter 7, require one month of CPU time, even on the fast processors of the APAC National Facility. Running the solution in parallel on four processors reduces this time to one week, which is a much more practical time frame using these simulations. If these simulations were run in parallel using 24 processors they could be reduced to the period of about one day. As a result of exploiting parallelism in the solution, the grid resolution achieved across the entire facility is comparable to, or better than, the grid resolution achieved for any previous numerical studies which simulated only a part of the facility.

Additional work, known as overheads, are incurred in managing the parallel execution and results obtained by each of the processors must be shared with the others. When sections of the code must be solved on a single processor, additional processors remain idle. These factors introduce work that would not otherwise have to be done and although the turn-around time may still be reduced, the parallel solution can be inefficient and the time advantage of parallelism greatly reduced.

1.3 Experimentation and Validation of the Simulations

In order to achieve a reliable representation of the flows, numerical simulations of shock tunnel facilities must be validated using experimental results. This is essential since there are many factors that can affect the accuracy of the simulations, including applicability of the gas models, the correct representation of the geometry of the facility and the validity of the assumptions that are made.

MB_CNS has been validated using many test cases outlined by Jacobs [111, 114]. In addition to this, the code has been used in many previous numerical studies, for

example, the simulation of expansion tube flows [113], which further demonstrates its ability with a range of compressible flow problems. Two further pieces of validation of the numerical techniques used are provided in this thesis: the interaction of shock waves with cylindrical bubbles and the experimental results obtained from experiments conducted in the Drummond Tunnel facility.

The Drummond Tunnel

This study uses the Drummond Tunnel to establish modelling techniques because it is relatively simple in operation, using a single, mechanically pierced diaphragm. It is also small in size, meaning that fine resolution of flow features can be achieved. The Drummond Tunnel operates in a regime in which the limited stagnation temperatures mean that the simulations are not dependent on the modelling of the thermochemical effects of molecular excitation, ionization and dissociation. Operating at relatively low enthalpy, the facility appears less susceptible to driver gas contamination than the larger T4 free piston shock tunnel.

Three series of experiments were performed in this facility. These experiments used two different operating conditions, one roughly tailored and the other over-tailored. The experiments provide sets of measured data which can be used to validate the numerical simulations. The simulations are provided with only the geometry of the facility and the initial operating conditions. This means that, in order to reproduce the observed experimental results, the modelling techniques implemented must be accurate, the assumptions made must be appropriate and all of the relevant physics must be included.

Shock Induced Deformation of Bubbles

The accurate modelling of the interfaces separating multiple component gases is essential to the shock tunnel flow simulations. The interaction of shock waves with bubbles of light gas and heavy gas is studied as a test case for the ability of the code in modelling the shock induced deformation and instability of these interfaces. Unlike the interface interactions occurring in a shock tunnel, this case has detailed experimental photographs, obtained by Haas and Sturtevant [89], which can be compared directly with the simulations.

1.4 Outline of the Thesis

The primary motivation for this thesis is to provide a better understanding of the flows in shock tunnel facilities through the development of numerical simulations. Towards this, the specific aims of this thesis are:

1. To develop time dependent simulations of a reflected shock tunnel facility operated at the University of Queensland, covering the complete facility, from the driver section through to the dump-tank.

2. To use these simulations to investigate the non-ideal flow phenomena that occur in shock tunnel facilities. This investigation focuses on the premature contamination of the test flow with driver gas and the generation of the high levels of noise experienced in the test flow.

3. To develop efficient parallel versions of the CFD code MB_CNS, using OpenMP and MPI, in order to use the large computational meshes that are required to simulate the complete facility with sufficient resolution.

4. To investigate the applicability of the numerical technique known as Smoothed Particle Hydrodynamics (SPH) by developing a full CFD code based on the technique and applying it to compressible flow test cases. As this technique is computationally expensive, it is also necessary that this code be able to solve the flow field in parallel.

The thesis is arranged as follows:

Chapter 2 discusses the operation of a reflected shock tunnel, first assuming ideal operation and then examining the non-ideal processes occurring during the operation of a real shock tunnel. The premature contamination of the test flow with driver gas is discussed in detail because the investigation of this phenomenon is one of the principal subjects of the simulation and discussion covered in Chapter 7. Chapter 2 also describes the experiments that were performed in the Drummond Tunnel by Dr. D. R. Buttsworth and the results obtained by these experiments.

Chapter 3 describes the numerical methods used by both MB_CNS and the SPH code. It is demonstrated that, although the MB_CNS uses a finite volume Eulerian formulation and SPH uses a particle-based Lagrangian formulation, the two approaches model what are fundamentally the same equations describing the motion of a compressible fluid. A review of previous numerical models of reflected shock tunnel facilities is also provided in this chapter.

Chapter 4 provides a review of parallel computing technology, in the context of CFD applications, both in software and hardware. This review provides a basis for the selection of the most suitable methods for compressible CFD applications. The importance of stable standards in computing is demonstrated and, as a result, OpenMP (using shared memory computers) and the Message Passing Interface (MPI) (allowing the use of distributed memory) are selected for use in this thesis and are investigated in Chapter 5.

Chapter 5 discusses the implementation of parallelism. A simple program for calculating π is used to introduce the concepts. The parallelisation of the CFD codes (MB_CNS and the SPH code) is then described in detail. Following this, the performance of MB_CNS in parallel using the QPSF SGI Origin 3400 and the APAC Compaq Alphaserver SC is examined. As the SPH code is not used in the shock tunnel simulations, the parallel performance of the SPH code is discussed in Appendix A.

Chapter 6 describes simulations of the interaction of shock waves with cylindrical bubbles of light and heavy gases, which is used as a test case. The simulation of multiple component gases, in which the interface between the gases experience hydrodynamic instability, is demonstrated. With the aid of experimental photographs, simulating this interaction process provides a means of validating the numerical techniques used in the shock tunnel simulation in Chapter 7. Unlike the shock tunnel flows, this case has experimental data in the form of Shadowgraph images, which can be compared directly with the simulations. MB_CNS is shown to provide accurate models of the shock-induced instability and deformation of interfaces between different gases. This interaction process also provides an interesting transient fluid dynamics case in its own right.

Chapter 7 describes simulations of the Drummond Tunnel facility. The details of the setup of the simulations in MB_CNS are described. Following the validation of the simulations with the experimental results described in Chapter 2, the simulations are used to investigate the flows through the shock tunnel. The simulation results are analysed in the context of particular processes occurring in the facility, focusing on the mechanisms leading to the contamination of the test flow with driver gas and the generation of the high levels of noise observed in the real facility.

Chapter 8 provides conclusions, outlines the contributions and the limitations of the simulations presented in this thesis and provides suggestions for future work in this area.

CHAPTER 2

Shock Tunnel Theory and Experimentation

The development of reflected shock tunnels has been driven by the requirement for producing high speed, high temperature flow in a ground based facility. The early history of impulse facilities, leading to the development of the reflected shock tunnel was described briefly in Chapter 1. Despite the development work that has taken place over time, aspects of the operation of reflected shock tunnels are not fully understood and the potential for significant development work remains. This chapter will begin by discussing the operation of a reflected shock tunnel. Following this, Section 2.2 will continue by describing experiments that were conducted in the Drummond Tunnel, a small reflected shock tunnel operated at the University of Queensland.

2.1 Shock Tunnel Operation

In this section, the operation of a shock will be described, first using idealised descriptions and then examining some of the non-ideal flow phenomena that affect their operation. Particular attention will be given to the processes that are thought to contribute to the premature contamination of the test flow with driver gas.

2.1.1 Ideal Operation

The starting condition for an idealised shock tunnel is shown in Figure 2.1. A high pressure *driver gas* is separated from a low pressure *driven gas* by the primary diaphragm. The shock accelerated driven gas becomes the test gas, which flows through the test section. The secondary diaphragm is used to separate the driven gas from the initially evacuated test section.

The operation of these facilities is initiated by the rupture of the primary diaphragm releasing the high pressure gas into the low pressure gas. Once released, the driver gas expands into the driven gas. This sends a shock wave through the

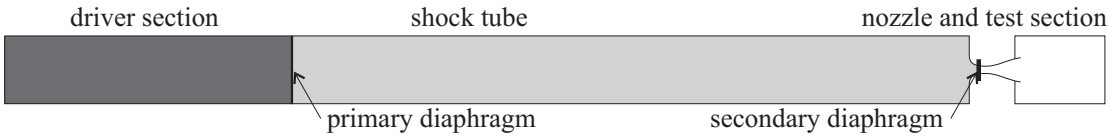

Figure 2.1: Initial conditions and layout for an idealised shock tunnel.

driven gas, compressing and heating it. The shock accelerates the driven gas towards the downstream end of the shock tube. Following the shock wave along the tube is the contact surface (or interface) between the driver and driven gases. This state is shown in Figure 2.2, in which the incident shock and the contact surface are moving along the tube.

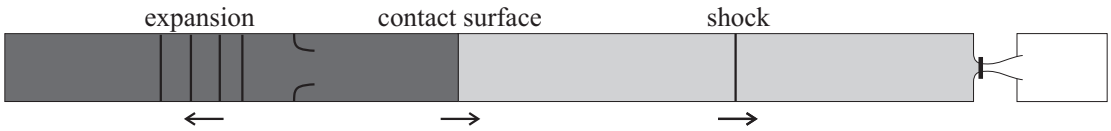

Figure 2.2: Idealised shock tunnel with the incident shock and contact surface travelling downstream. The unsteady expansion is seen eminating from the primary diaphragm location.

Under ideal conditions, such as flow without wall boundary layers or real gas effects, the shock wave and contact surface would be planar, moving at constant velocity, and the flow between the two waves would be uniform. Without the influence of boundary layers, the flow is one-dimensional.

Once the shock reaches the end of the tube it reflects off the end wall and travels back upstream through the tube and into the oncoming test gas; this reflected shock brings the test gas to rest. This stationary, high temperature, high pressure gas is then expanded through a converging-diverging nozzle into the test section, providing a high velocity, high Mach number test flow. This state is shown in Figure 2.3, in which the test gas supply is stagnated at the end of the tube and is being expanded through the nozzle as the test flow. Ideally the test time is limited by the finite mass of the test gas, lasting until the driven gas is exhausted.

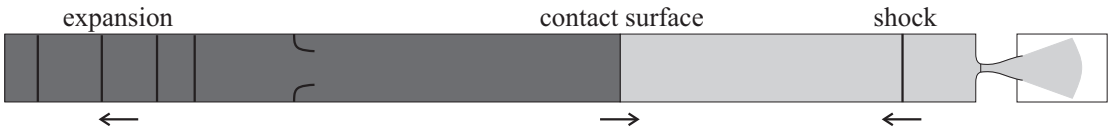

Figure 2.3: Idealised shock tunnel with the incident contact surface, the reflected shock travelling back upstream and the test gas being expanded through the nozzle into the test section.

At a later time the reflected shock crosses the contact surface. This is an important interaction in the operation of the shock tunnel. Depending on the initial conditions, additional waves may be generated by this interaction, known as tailoring waves, which will be described in Section 2.1.2.

In the x-t diagram, the upstream end of the unsteady expansion of the driver gas can be seen to reflect from the upstream wall of the driver section and travel rapidly down the tube as a u+a wave. These waves can arrive at the stagnated test gas early enough to end the test time for some operating conditions.

In an idealised shock tunnel, the test flow is steady and lasts from the time that the shock reflects from the end wall of the tube until either [248]:

1. all of the stagnated test gas has expanded through the nozzle
2. tailoring waves propagate into the test flow
3. the reflection of the unsteady expansion reaches the test flow

2.1.2 Tailoring

The use of tailored conditions in a shock tunnel was first described by Wittliff [248]. The conditions of the driver and driven gases are said to be tailored if no additional waves are created by the interaction of the reflected shock with the incident contact surface. Tailored conditions are characterised by the ability to produce a steady nozzle supply pressure and an interface that is stationary following the passage of the reflected shock. If the reflected shock is to pass through the interface without generating any additional waves, the driver and driven gases must be at equal pressure and velocity behind the reflected shock [27]. This means that the condition for tailoring is that the pressure and velocity change across the reflected shock should be the same in the two gases [248].

When the reflected shock reaches the oncoming contact surface, at tailored conditions the momentum change imparted by the shock on both the driver gas and the driven gas is the right amount to stop both of them. With over-tailored operation, the driver gas has too much momentum after the interaction and continues forward into the stagnated driven gas. With under-tailored operation, the driver gas does not have enough momentum to come to rest and is pushed back, ending up with backward velocity after the passage of the shock. Figure 2.4 shows x-t diagrams for the three tailoring cases.

In the over-tailored mode of operation, a series of reflected shocks are produced which move into the nozzle supply region. This mode of operation allows the contact surface to continue to move towards the nozzle, increasing the likelihood of

contamination. In the over-tailored case, the reflected waves quickly propagate into the stagnated test gas, and end the test time early.

In the under-tailored mode of operation, the interaction results in expansion waves being propagated into the nozzle supply region. This causes the nozzle supply pressure to decrease, but does cause the contact surface to move back upstream, potentially reducing the chance of driver gas reaching the test flow.

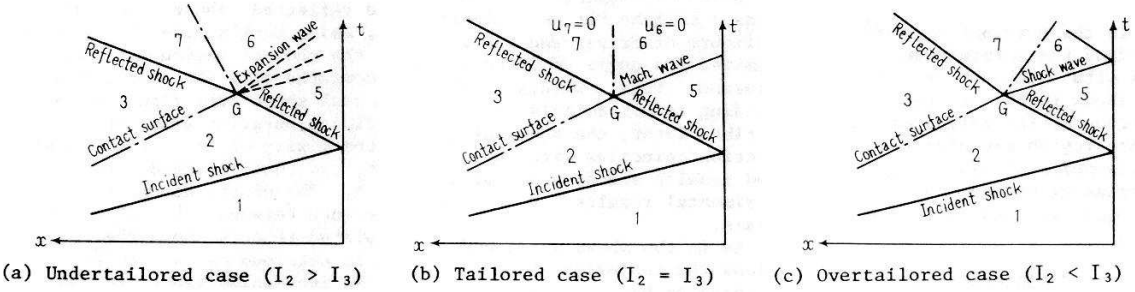

Figure 2.4: x-t diagrams showing under-tailored, tailored and over-tailored conditions at the reflected shock, contact surface interaction. Reproduced from Matsuo [144].

Until around 1992 it was believed that running a shock tube with the driver undertailored would help to prevent driver gas contamination. Causing the contact surface to receed after the passage of the reflected shock was thought to prevent the driver gas from being able to penetrate the bifurcated reflected shock and contaminate the test flow. This is now not believed to help the problem significantly and most shock tunnels are run with tailored conditions.

2.1.3 Non-ideal Operation

The test times produced by real shock tunnel facilities are a fraction of those predicted by idealised models of their operation. This reduction in test time is caused by non-ideal processes that occur during the operation of a shock tunnel. These processes are predominantly driven by the formation of viscous boundary layers on the walls of the shock tube. These boundary layers are formed as the shock propagates along the tube.

The theory of Mirels [148] has been used extensively to estimate the effect that viscous losses have on shock tunnel operation [206, 240]. This theory estimates the amount of test gas lost to the boundary layer as a function of the shock Mach number, the test gas density and the tube geometry. The theory of Mirels is supported by the experimental observations of Glass and Patterson [79].

The contact surface can be thought of as a piston driving the shockwave. In this analogy, the contact surface would be a leaky piston, in that mass is being

removed into the boundary layer at the contact surface [148, 138]. The boundary layer material moves more slowly than the contact surface and so, this driven gas is overtaken by the driver gas. This flow of material from the test gas, around the contact surface and into the driven gas causes the contact surface to accelerate [255].

The *separation distance* is the distance between the shock and contact surface. In an ideal shock tunnel, the separation distance increases as a linear function of the distance from the wall. The effect of the viscous boundary layers, acts to reduce the separation distance through the attenuation of the shock and the acceleration of the contact surface. Mirels [147] showed that, at some limiting distance, the test mass flowing into the boundary layer would be equal to the test gas flowing across the shock, resulting in a steady separation distance. As the length to diameter ratio of the shock tube is increased and as the initial pressure in the shock tube is reduced, the boundary layer effects become more pronounced [65, 147].

This attenuation results in a loss of test time as the separation distance is decreased. Hooker [104] provides that, as a general rule, the actual test flow is about one-third to one-quarter of the theoretical prediction. It also causes total enthalpy to be lost and the flow conditions in the test section to become unsteady [248]. The growth of this boundary layer, in the region between the shock and the contact surface, results in a slight increase in the pressure through this region as a function of the distance from the shock [198].

Skinner [212] states that the loss of test gas is predominantly from near the upstream end of the test gas slug. As a result of this Skinner believed that the test gas lost to the boundary layer is likely to be unusable in the test section.

The non-ideal operation is depicted in the x-t diagram in Figure 2.5. Like the idealised x-t diagram, the motion of the shock and the contact surface can be seen; however, in this figure the shock is shown to decelerate and the contact surface is shown to accelerate and mix as the flow propagates along the tube. The resulting decrease in the separation distance, and therefore test time, is evident.

Turbulence in the Boundary Layers

The boundary layers in the Drummond shock tunnel resulting from the flow conditions studied in this thesis are almost completely turbulent. Assuming a transition Reynolds number of 1×10^6, the boundary layers undergo transition at a distance of 15 mm from the incident shock position. Turbulence in the boundary layers along the shock tube have a significant effect on the operation of the facility.

The effect of turbulent boundary layers on shock tunnel flows was investigated by Mirels [148]. Turbulence in the boundary layer increases transfer rates to the

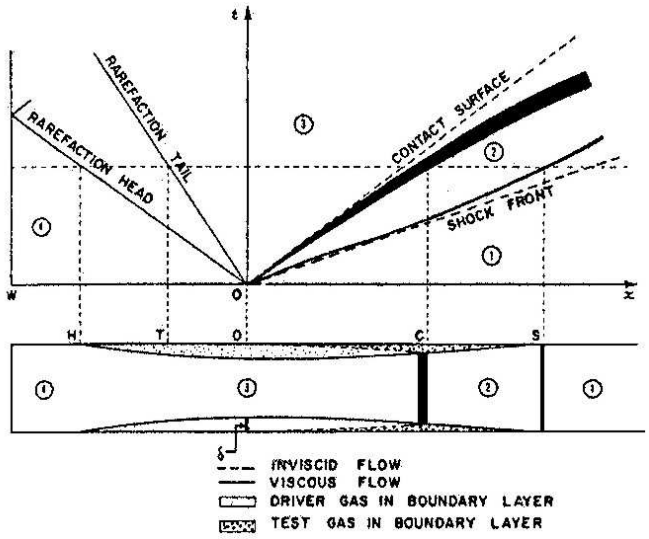

Figure 2.5: x-t diagram of a non-idealised shock tunnel, showing the attenuation of the shock and acceleration of the contact surface. The mixing at the contact surface is also shown. Reproduced from Dewey and Anson [57].

boundary layer increasing these effects. Mirels demonstrated cases in which the influence of turbulent boundary layers reduced the test time flow duration achieved to a fraction between 0.1 and 0.5 of the theoretical value.

The theory of Mirels [148], commonly applied to laminar boundary layers, can be applied to turbulent boundary layers. A later study by Fuehrer [72] measured the test time obtained in both hydrogen and air in a high pressure shock tunnel to be considerably less than that predicted by Mirels' turbulent boundary layer theory. This conclusion was also reached by the experimental study of Jacey, Jr. [117]. Fuehrer [72] proposed modifications to the theory of Mirels to account for this difference. Other studies of the effect of turbulent boundary layers were conducted by Bazhenova et al. [12] and Dumitrescu, Brun and Sides [66].

A laser-induced fluorescence image of an incompressible turbulent boundary layer is shown in Figure 2.6. This figure qualitatively demonstrates the significant differences between a turbulent boundary layer and the laminar boundary layer.

2.1.4 Diaphragm Rupture

In almost all shock tunnels and expansion tubes, the diaphragm initially separating the driver gas from the driven gas is made of metal. The use of a metal diaphragm means that the initial conditions in these facilities are very different to an instantaneous removal of the separation between the two gases. The actual opening process is gradual and is dependent on the complex manner in which the metallic diaphragm

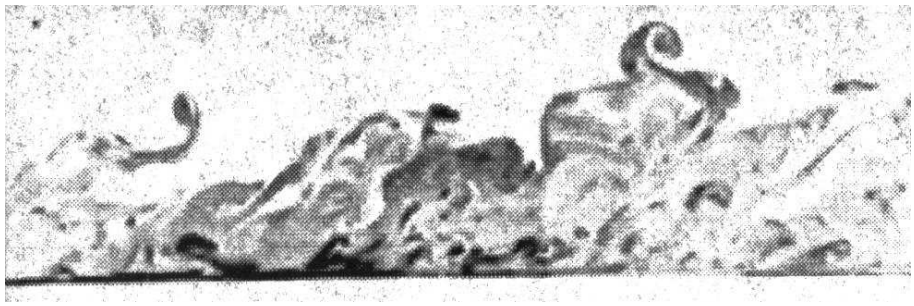

Figure 2.6: A laser-induced fluorescence image of an incompressible turbulent boundary layer. The flow is from left to right. The Reynolds number based on momentum thickness is 700. Reproduced from Wilcox [244].

deforms and ruptures.

It is a common practice to score the diaphragm material. In the experiments used in this study, the rupture of the diaphragm was initiated with the piercing of the diaphragm with an acuated spike. Scoring and piercing are used in order to produce a more repeatable and symmetrical rupture process. The diaphragm eventually tears along lines of natural or enforced weakness, folding outwards as petals of diaphragm as the driver gas spills through it.

The rupture of the primary diaphragm has a significant effect on the resulting flow. For this reason, simulations of the complete shock tunnel facility should include a model that accounts for the effect of the rupture process.

The diaphragms that were used in this study are 1 mm thick aluminium, separating a 59 mm driver section from a 62.2 mm shock tube and are pierced to initiate the rupture process. The literature review is limited to the characteristics of diaphragms similar to these. Used diaphragms from the Drummond Tunnel were investigated, indicating that the rupture process is roughly symmetric, resulting in petals that are roughly the same size and the opening process leaves a circular cross-section averaging 57 mm in diameter.

With almost all shock tube flows being initiated with the rupture of a metal diaphragm, and little being known about, not only the mechanics of the process, but its effect on the resulting shock tube operation, experimental studies aimed to provide an understanding of the process. Experimental studies attempting to understand, and quantify, the process, included: White [243]; Campbell, Kimber and Napier [36]; Simpson, Chandler and Bridgman [209]; Rothkopf and Low [197]; and Hickman, Farrar and Kyser [99]. An aim of these studies was the development of simplified analytical models of the effect of the rupture process on the flow. These models were often inaccurate when applied across a variety of diaphragm characteristics and flow conditions.

In the most relevant study, Rothkopf and Low [197] performed an experimental study of the opening process of various types of diaphragms. Their study included a qualitative analysis of how diaphragms with different characteristics ruptured and a quantitative study of the change in projected area through the open aperture as a function of opening time. The projected area was measured by continuously measuring the amount of light passing through the diaphragm using a photomultiplier. Their study considered the rupture of various thicknesses of diaphragms made of aluminium, copper and brass, using a 54mm square shock tube section. Figure 2.7 shows the profiles of diaphragm opening times that were obtained by Rothkopf and Low [197]. In the figures, both the areas and times are normalised by their final values. The similarity of the effect of the different diaphragms is evident in this figure, as all of the profiles have the same general shape: at first there is a slow increase in the open area, but after the area has opened to around 20% of the final area, the rate of opening is roughly linear. The rate at which the opening occurs varies between diaphragm materials, thicknesses, and the shape of the aperture. For a given size, the rupture process is similar, except that the process is slower than for an equivalent square cross section. Towards the end of the opening process, the rate of opening again slows. Of particular interest to this study is the profile of ruptured area versus time of the aluminium diaphragms observed in their study. This profile is shown in Figure 2.8, along with the profile for the copper diaphragm.

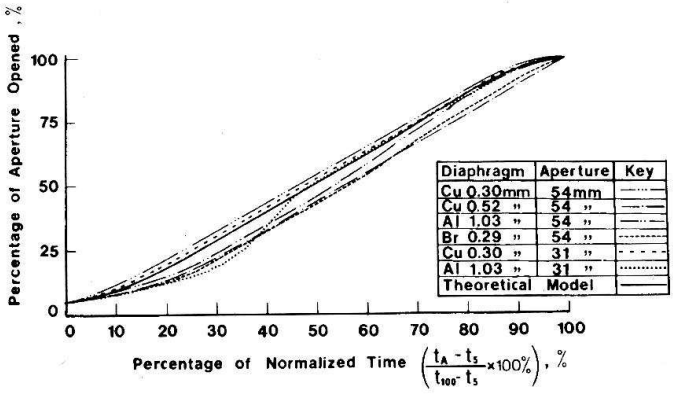

Figure 2.7: Normalised opening time curves for square aluminium, copper and brass diaphragms of varying thicknesses and apertures. The graph shows the similarity of the profile of open area versus time for diaphragms with different characteristics. Reproduced from Rothkopf and Low [197].

The results of Rothkopf and Low [197] indicate that, being relatively brittle, very little bulging of the diaphragm is observed with the aluminium diaphragms. The behaviour of copper and brass diaphragms is very different and significant bulging of these diaphragms was observed. The thickness of the diaphragm determined

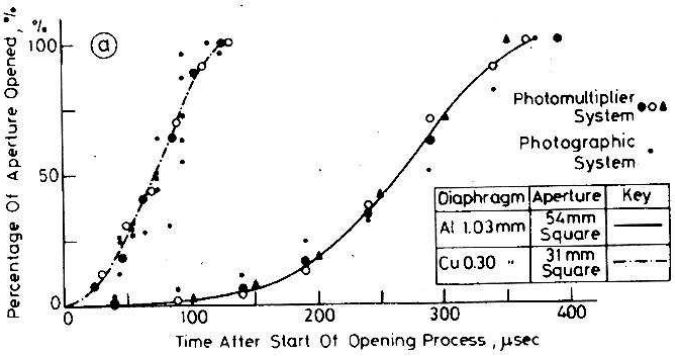

Figure 2.8: Normalised profile of open area versus time for aluminium diaphragms. Reproduced from Rothkopf and Low [197].

the pressure at which the diaphragm ruptured, but did not have an affect on the amount of bulging observed. The aluminium diaphragm, unlike the other metals, was observed to rupture in a symmetric manner.

In a later paper Rothkopf and Low [196] examined the period of shock formation and the resulting initial shock speeds in detail. The initial shock speeds were low and approached the ideally predicted level over a distance quoted as the shock formation distance. This distance was found to be proportional to the effective opening time of the diaphragm and inversely proportional to the average of the sound speeds of the driver and driven gases.

In a real shock tube, as with the ideal representation, the primary shock becomes planar very quickly. This distance is a function of the opening time [209], but is usually within two tube diameters of the diaphragm [35]. The development of the primary shock following the rupture of the diaphragm has been described by Petrie-Repar [176].

The contact surface is also initially distorted due to the way the gas spills through the slow opening diaphragm. The majority of the distortion occurs as the driven gas and the driver gas spills through the diaphragm opening. The contact surface is also affected by its interaction with the spherical parts of the shock, which reflect back and forth across the tube. The dynamics of the metal material fragments broken from the diaphragm also cause distortion of the contact surface. Unlike the shock, the contact surface is not stable and does not inherently become planar with time. The development of the contact surface as it progresses along the shock tube is discussed in Section 7.3.6.

In addition to the experimental studies, further insight into diaphragm rupture mechanics was sought in numerical simulations. Early simulations were performed by Satofuka [201] and Outa, Tajima and Hayakawa [169], who modelled the opening

diaphragm as an iris.

Later, Cambier, Tokarcik and Prabhu [35] performed axisymmetric simulations of the flow resulting from a gradually opening diaphragm. Their simulations demonstrated that: the primary shock becomes planar very quickly; the contact surface forms a complex shape distorted by the rupture process; and showed the complicated wave structure behind the contact surface dominated by a Mach disk. It was noted that that contact surface did not become planar with time, but would continue to evolve along the shock tube.

Petrie-Repar [176], and Petrie-Repar and Jacobs [177], solved the mechanics of the rupturing diaphragm using an unstructured, finite-volume code. The simulations achieved a high resolution through the use of an adaptive mesh. The simulations were inviscid and were only concerned with the early development of the flow following the rupture of the primary diaphragm. This study investigated the effect of varying the initial pressure ratios and the diaphragm opening time. A sequence from a simulations is shown in Figure 2.9.

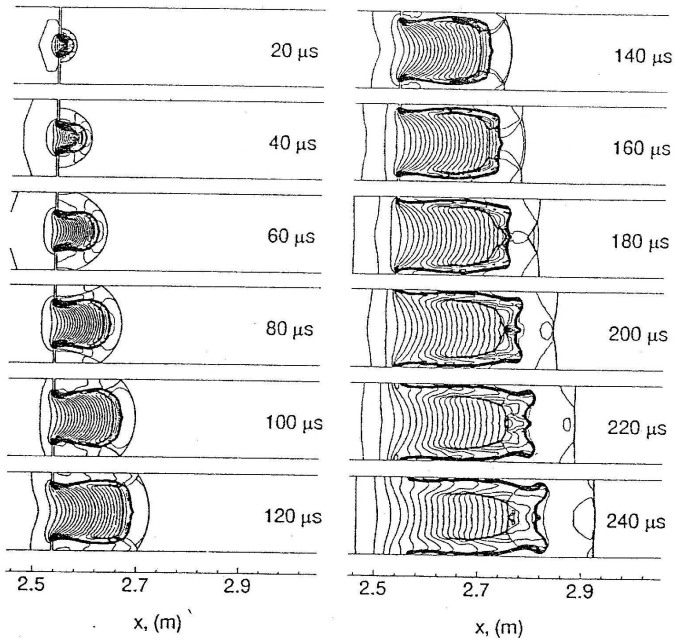

Figure 2.9: Figure reproduced from Petrie-Repar [176] demonstrating the flow resulting from the iris based diaphragm rupture model used.

Additional simulations were performed in Petrie-Repar [176]; however, these simulations were based on the rupture of the thin secondary diaphragms made from Mylor. These simulations are made around two assumptions: the first model assumes that the diaphragm vaporises immediately after the arrival of the incident shock, and the second model assumes that the diaphragm shatters into a number of pieces which can be treated as rigid bodies.

Rothkopf and Low [197] made experimental observations using similar diaphragms to the ones used in the Drummond Tunnel (aluminium material; 1mm thickness; 54mm square section instead of a 62.2mm circular section; pre-scored instead of pierced). It is thought, based on their observations that the characteristics of the rupturing process can be captured relatively well with an iris based model. The rupture is relatively symmetric, does not involve a significant bulging of the diaphragm, and with this small amount of deformation, a small amount of energy taken out of the flow. In effect the diaphragm folds outward gradually increasing the cross sectional area through which driver gas can flow through, much like the iris model predicts.

Zeitoun, Brun and Valetta [254] account for the effect of the rupture of the primary diaphragm in their one dimensional model, stating that it results in a strong acceleration of the gas, followed by a slow deceleration.

The implementation of the iris based model is described in Section 7.1.4 and the flow resulting from the used of this model is described in Section 7.3.1.

2.1.5 Shock Reflection

When the shock reaches the end of the shock tube it reflects from the end of the tube back into the test gas. This means that the shock must reflect from the upstream end of the nozzle. Some facilities, such as the Drummond Tunnel, have a smooth convergent upstream section to the nozzle, while others have a flat end wall. The requirement for the nozzle on the end of the shock tube means that the shock reflection process is a complex process; however, the inherent stability of the shock means that it will rapidly coalesce and become planar.

Many numerical studies examining the shock reflection process have been conducted, including Lee [129], Cambier et al. [35], Jacobs [111], Craddock [53] and Nishida and Kishige [164]. The details of the shock reflection process are discussed in these studies. These studies have achieved agreement with experimental photographs.

The reflection process has been observed to produce a vortex in the stagnated flow near the centreline [35, 111]. This vortex has been observed to form during the reflection from different types of nozzles. He [127] observed the production of significant levels of noise in the stagnated gas resulting from this reflection process.

In most facilities, the ratio of the nozzle throat area to the shock tube cross sectional area is small enough that the effect of the air going into the nozzle is not significant and the shock is almost completely reflected, creating an almost stagnant, compressed gas at the end of the shock tube. The strength of the shock adjusts itself

to provide a constant pressure at the nozzle entrance, even though gas is flowing out of the shock tube [248].

Lee [129] simulated the process of shock reflection from Amann's reflection nozzle [4]. This nozzle is two dimensional and has a flat surface at the end of the shock tube, with a small anulus into the nozzle; this makes the reflection process more simple than in the Drummond Tunnel. These simulations only modelled the last 40 mm of the shock tunnel, which causes problems at later times as the shock reflected from the downstream wall, containing the nozzle anulus, reach the upstream inflow boundary. Before this time, the simulations compared well with the experimental shadowgraphs of Amann [4].

2.1.6 Interaction of the Reflected Shock with the Boundary Layer

As the incident shock moves along the tube it grows a boundary layer on the wall of the shock tube behind it. When the shock reflects from the nozzle at the end of the tube, it must travel back upstream through this boundary layer. Soon after reflection, the shock forms a Mach interaction near the wall of the tube. This process is caused by its interaction with the boundary layer.

The energy deficient boundary layer material does not have enough energy to cross the normal shock and instead builds up at the foot of the shock. The stagnation pressure of the boundary gas is less than that of the free stream gas. This causes the shock to bifurcate and the core flow to separate as it passes around the foot material and through the oblique shock structure that is built up. Due to the entropy difference between the gas that has passed through the normal shock and the gas that has passed through the oblique shocks, a shear layer is formed. This shear layer is unstable and has been shown to form discrete vortices [240].

This interaction phenomenon was first described by Mark [141]. Mark demonstrated that for incident shock Mach numbers between a certain range the stagnation pressure of the fluid in the boundary layer is exceeded by the pressure behind the reflected shock. For the experimental conditions investigated by Mark, the range within which the material would build up at the foot of the shock was incident shock Mach numbers of 1.3 to 6.4. Mark also showed that the gas emerging from the two oblique shocks has a higher velocity than the free stream gas.

Mark developed a generalised model of the phenomena, capturing what the author thought were the important features of the flow. This model was developed further by other authors, including Sanderson [200]. Figure 2.10 shows a schematic of the interaction process used in Sanderson.

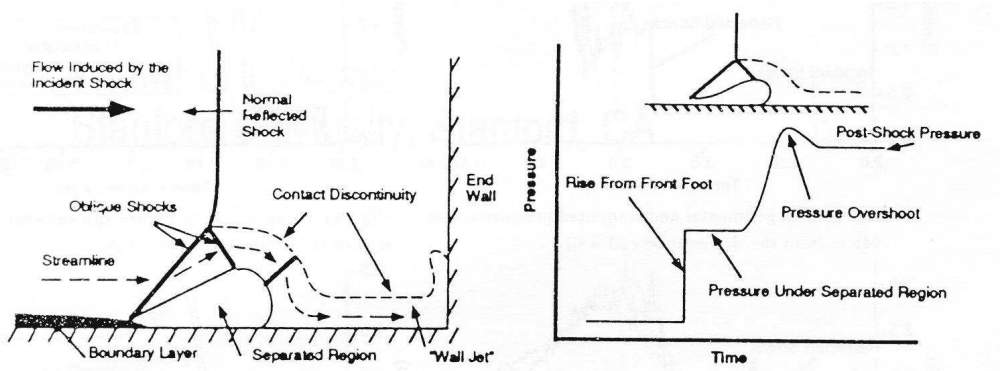

Fig. 2 Schematic diagram of the interaction of the reflected shock with the boundary Layer.

Fig. 3 Schematic diagram of the pressure trace under a bifurcated shock.

Figure 2.10: A schematic of the modified version of Mark's [141] representation of the shock boundary layer interaction used in Wilson, Sharma and Gillespie [247].

Wall jetting of the driver gas leading to early contamination of the stagnated region and therefore of the test gas. Also the driver gas is relatively cold and therefore, as well as contamination of the test gas, the driver gas cools the test gas prematurely. While the gas being jetted through the shock is also driven gas, the heat added to the gas from its interaction with the two oblique shocks is negligibly smaller than the heat added through the normal shock [27]. This means that this process, although imparting velocity on the gas jetted along the walls, does not introduce significant temperature fluctuations in the region behind the reflected shock; however, when the reflected shock reaches the contact surface, cold driver gas material can be jetted through the shock foot into the hot test gas.

Another important contribution by Mark [141] was the identification of a limit on Reynolds number for the effect to be recorded experimentally. Mark stated that the picture gradually changed as the Reynolds number was increased until the effect disappeared almost completely. This was presumed to correspond to the boundary layer becoming turbulent. A Reynolds number of 900,000 was identified in two sets of experiments as the limit for the observation of the bifurcated shock foot. This Reynolds number was stated as the transition Reynolds number for the flows used in the sets of experiments. With the turbulent boundary layers, experimental results were reported to have shown a normal shock propagating along the tube, with slight forward concavities at the walls of the tube. It was believed that the same shock bifurcation was occurring as with the laminar boundary layer, but on a smaller scale so as not to show up on the photograph. Rudinger [198] also noted that the bifurcation of the foot of the reflected shock does not occur for turbulent incident boundary layers, with the reflected shock tending to remain as a planar shock front. Davies and Wilson [55] extended Mark's model by accounting for the effect of the

growth of the interaction region.

Bull and Edwards [27] investigated the interaction processes in a blanked end shock tube by tagging driver gas, by making it infrared active, in order to obtain direct evidence of its arrival at the end wall. A H_2 driver gas was tagged by mixing it with small quantities of CO_2, which is an infrared active permanent gas. Various test gases were used. The results obtained evidence of the premature arrival of driver gas at the end wall following the reflection of incident shock waves in the Mach number range from 2 to 6, in a Nitrogen test gas. These results pointed to a strong shock bifurcation process in this gas. The durations before the arrival of driver gas in the stagnated test gas were in agreement with theoretical analyses based on Mark's model. They showed that for a monotomic test gas, such as with Argon used in their study, little or no shock bifurcation occurs, and the period before the arrival of the test gas was considerably longer than those observed in Nitrogen. The arrival times in Argon corresponded with the predicted arrival times of the reflected expansion from the driver end of the shock tube. Bull and Edwards believed that their results gave support for the primary cause of contamination being the shock bifurcation and jetting, rather than contact surface instability or the mechanics of the primary diaphragm opening.

Taylor and Hornung [233] investigated the effects of real gas effects and shock tube wall roughness on the interaction process. The experiments were conducted at sufficiently high shock speed to produce vibrational excitation and dissociation of the Nitrogen and Carbon Dioxide test gases. The experimental results were shown to be in reasonable agreement with a modified version of Mark's model for the Nitrogen experiments; however, there were large discrepancies for the Argon and Carbon-Dioxide gases.

Numerical simulations of the shock boundary layer interaction process occurring in a shock tunnel will be described in Section 3.3.

2.1.7 Development of the Contact Surface

The contact surface is heavily dependent on the mechanics of the rupturing diaphragm [176]. It is also dependent on its stability properties as it moves along the shock tube [20]. Despite the importance of the contact surface shape and characteristics on the resulting flow, the difficulty in obtaining experimental data on the contact surface has meant that publications in this area are limited in comparison to other areas.

Experiments on the contact surface were performed by Hooker [104] in a low pressure, small diameter shock tube. This combination of low pressure and small

diameter increases the importance of viscous effects [65]. A rake of heat flux probes inserted into the flow was used to determine the shape of the contact surface, as was done in this study. These measurements indicated that the contact surface was turbulent, but had an essentially planar profile. This planar profile was observed at 1.05 m up to at least a distance of 4.7 m. The contact surface was described as a region of widely varying composition and temperature, the extent of which is almost solely a function of the tube geometry. These observations are believed to be influenced by the low pressure, small diameter conditions studied. They also neglect the stability of the contact surface and, therefore, the operating conditions as a factor.

Experiments show that by the time the shock has made it to the end of the shock tube and the test flow begins, the interface between the driver gas and the driven gas is turbulent. This turbulent region, along with boundary-layer effects and other effects, usually engulfs a significant part of the heated driven gas; in some cases up to half of the total driven gas [206]. Cambier, Tokarcik and Prabhu [35] discussed the effects that the Rayleigh-Taylor instabilities may have on the development of the actual contact surface in a shock tunnel. Cambier et al. took into account viscous effects which slowed down the contact surface at the walls, but the jetting of the contact surface near the walls relative to the centre of the tube was evident [176].

Zuev, Vasilieva and Mirshanov [255] studied the contact surface in a shock tube over a range of Mach numbers from 3.5 to 13.5 using X-ray diagnostic techniques. The authors stated, based on their experimental observations that the loss of test gas to the mixing region at the contact surface is more significant than the loss to the boundary layer as observed by Mirels [148].

Houwing et al. [106] observed the contact surface in the T3 shock tunnel using differential interferometry. The instability of the contact surface was observed and, for regions in which the contact surface was decelerating, long tongues of gas were measured to penetrate from the heavy driver gas into the light test gas.

2.1.8 Interaction of the Reflected Shock with the Contact Surface

The interaction of the reflected shock with the contact surface occurs between the reflected shock, following its interaction with the boundary layer, and the contact surface that has evolved along the length of the shock tube. Significant amounts of vorticity can be generated at this interaction process, as a product of the Richtmyer-Meshkov instability [26]. The interaction of the reflected shock with the contact surface has received less attention that the interaction with the boundary layer. The

Richtmyer-Meshkov instability was described and demonstrated, for the interaction of a planar shock wave with a cylindrical bubble, in Chapter 6.

Dumitrescu, Popescu and Brun [67] discussed the interaction of the reflected shock with the contact surface, following its interaction with the boundary layer. In order to investigate the effect of the reflected shock interaction processes on the nozzle supply region they conducted a study of the shape of the incident contact surface and the influence of the shape of the boundary layer on it. A hot-wire probe was used, which was placed at various distances from the diaphragm, to detect the arrival of the contact surface. The boundary layer remain laminar through their study. The experimental studies conducted in two facilities, using various combinations of driver and driven gases, operating with ranges of pressures. These facilities are blanked end shock tubes, with sensors places at locations on the tube walls and the end plate.

Dumitrescu, Popescu and Brun were unable to provide detailed explanations of the flow features indicated in their results and it was concluded that detailed visualisation of the flow in the region between interaction process and the end wall would be useful. It was demonstrated that, even when the interaction process was occurring on the shock tube wall, no disturbances reached the centre of the end plate. This conclusion could indicate that these disturbances would not reach a nozzle entrance in an equivalent shock tunnel; however, with the stagnated gas flowing out through a nozzle, this flow would likely promote these disturbances in reaching the test flow.

The authors conclude that the temperature fluctuations detected at the end wall, soon after the reflection of the shock wave, are due to the jetting of gas through the bifurcated foot of the shock. In certain cases, and in contrast to the pressure measurements, temperature rises are detected for a range of tailoring conditions and was believed to be due to the instability of the contact surface. The degree of stability of the contact surface, and therefore the susceptibility to this effect depended on the gases used in the operation while at tailored conditions. A pressure gauge on the end wall detected the arrival of tailoring waves from the interaction of the reflected shock with the contact surface. A heat transfer gauge showed a disturbance corresponding to a increase in wall temperature either below of above tailoring; this disturbance was even shown to occur when the tube was operated at roughly tailored operating conditions (with a shock Mach number of 3.65). The end wall pressure was shown to be practically constant, as would be expected.

The possible explanation provided by the authors for this effect is the arrival of cold driver gas at the end plate, arriving there due to the instability at the interaction with the reflected shock. This gas has a high density and a residual

velocity at the end plate and may have resulted in the temperature rise. Gauges near the wall of the tube at the end plate showed that this disturbance reaches the centre of the end plate significantly before (about 3 ms) the outer part of the end plate. This indicates that the disturbance moves along the centre of the tube. Above tailoring, the duration of the undisturbed flow at the end plate coincides for the heat transfer gauges at the centreline and end plate edge, and for the pressure sensor. This indicates that for the over-tailored conditions, the disturbance is the tailoring wave, rather that the unstable driver gas.

2.1.9 Driver Gas Contamination

In a real shock tunnel only a fraction of the test gas is used before waves from other sources enter the test flow. As a result of this, the period of steady pressure in the test flow has been used as the measure of test time [12]. Stalker and Crane [221] showed that the arrival of driver gas in the test flow, and with it the end of the test time, can occur earlier than the loss of steady pressure. The arrival of driver gas in the test flow causes the end of the test time as it renders the test flow unusable for testing. Even if the driver gas is the same type of gas as the test gas, the driver gas is significantly colder than the test gas and, therefore, the driver gas cools the test gas prematurely, affecting the expanded test flow properties [228].

Under certain conditions, driver gas is projected forward through the test gas by processes occurring at the end of the shock tube, entering the test flow well before the supply of test gas is exhausted. This process is known as driver gas contamination. It results from many complex flow phenomena occurring in the shock tube, including interactions between the reflected shock and the boundary layer and contact surface.

The predominant approach that has been taken to the investigation of driver gas contamination is the detection of its driver gas in the test flow. This data can be used for either the investigation of trends in the data, or to identify the degree of contamination for the particular conditions used in the experiments.

Skinner [212] designed and constructed a time of flight mass spectrometer. This device was used in the test section of the T4 shock tunnel to record time histories of all species concentrations in the test flow. The detection of the premature arrival of driver gas in the test flow was the primary concern. The results shows that contamination of the test flow occurred for nominal enthalpies above 10 MJ/kg, with the degree of contamination becoming progressively larger for higher enthalpies. Skinner discussed many mechanisms which could have lead to the contamination observed; however, no certain conclusion as to the cause of the contamination was reached in the study.

Jenkins, Stalker and Morrison [115] provided one of the first trends that could be used in the prediction of contamination. A qualitative relationship between the constancy of the stagnation pressure and the arrival of the driver gas was suggested.

A gasdynamical detector, consisting of a small duct and a wedge was constructed by Paull and King [174] and Paull [173]. The duct and wedge were arranged so that the duct would choke for any test flow in which the contamination reached a certain level. This device was used in the T4 shock tunnel and could be used in conjunction with an experimental model in the test section. Sudani and Hornung [227] designed similar detectors based on this concept, with modifications that provided greater sensitivity to the degree of contamination. A two dimensional duct detector was designed which allowed for visualisation of the flow through the duct.

Methods of extending the test time by postponing, or preventing, the driver gas from reaching the nozzle throat have been proposed. The utility of these devices generally assumes that the driver gas is jetted along the wall. Dumitrescu [68] proposed a device with a slit opening at the corner of the end plate. Chue and Dumitrescu [44] showed, both numerically and experimentally, that this device had no effect in preventing driver gas contamination. In addition to this, Sudani and Hornung [226] showed that such a device actually advances the arrival of driver gas in the test section. Cordoso et al. [37] proposed the same type of device, except positioning it further upstream. Numerical simulations indicated that this device would work; however, in experimental trials, it was ineffective.

Driver gas contamination is the primary cause of the end of the test time in high enthalpy operation in facilities such as T4 [221]. With the low enthalpy Drummond Tunnel, any driver gas detected in the test flow for the operating conditions used in this study is after the test flow is already ended; however, this facility can still be used to investigate the mechanisms leading to driver gas contamination.

2.1.10 Test Time and Flow Quality

A rigid definition of test time permits no variation of flow properties; however, noise from the development of the flow through the facility means that there will be fluctuations even during the test flow. The test time is usually defined as the duration for which the variation in the flow properties is within an acceptable tolerance [212].

Test time limits the size of models that can be used in these facilities due to the requirement on flow establishment time. Depending on the type of model being tested, the test flow length is required to be 3 times the length of the model, in order for a steady flow to be established during the test [212, 115]. The short test

times are also important for practical reasons, by preventing damage to the models or the facilities themselves, caused by high temperature gases being produced.

In an ideal shock tunnel the test time is ended by the exhaustion of the driver gas slug, or the arrival of tailoring waves or the reflected expansion in the test flow. In the operation of a real shock tunnel facility, the test flow duration is further limited by non-ideal effects, including:

1. the reduction of the separation distance caused by the viscous attenuation of the shock and acceleration of the contact surface.
2. transient waves as the nozzle test flow starts and break down
3. the premature arrival of the driver gas in the test flow (driver gas contamination)

Thermochemical Effects

High temperatures are caused by the stagnation of the test gas with the reflected shock. As a results of these high temperatures, the thermal energy of the gas can be sufficient to excite the rotational and vibrational states of the molecules, cause dissociation and ionization and promote chemical reactions.

This gas remains effectively frozen at non-equilibrium conditions during its expansion through the nozzle. This thermochemically modified gas then flows through the test section. Due to the relatively low stagnation temperatures experienced in the Drummond Tunnel (below 1600 K) these effects are not significant in this study. This type of effect must be given consideration in larger facilities such as the T4 free piston driven reflected shock tunnel, which experiences significantly higher stagnation temperatures in its operation. These issues are not addressed in this thesis.

Noise in the Test Flow

Even during the time period that is relatively steady and is used as the test time, significant fluctuations in flow properties are still present in the test flow. Noise levels in shock tunnels are significantly higher than those in free flight. These fluctuations can impact on results by affecting, for example, boundary layer transition on the experiment [127, 202]. In a reflected shock tunnel, noise would be expected to originate through the non-ideal processes that occur throughout the flow development.

Paull and Stalker [175] studied the generation of noise and and its effects in an expansion tube flow. Paull observed that noise first appeared in the driver gas and

was then transmitted into the test gas, as the two gases propagate along the tube. The majority of this noise generated in the driver gas originates from the oblique shocks that form due to the finite rate of the primary diaphragm opening. The noise that was transmitted into the test gas was shown to remain in the test gas as it entered the test section. It was shown that only the high frequency components of the noise in the driver gas gas could penetrate the interface between the driver gas and the test gas. The penetration of the noise into the test gas was found to be limited by a sufficient increase in sound speed across the interface from the driver gas to the test gas. These observations are also applicable to noise in a reflected shock tunnel and the contact surface separating the driver gas from the driven gas in these facilities.

In addition to the transmission of noise from the driver gas, significant levels of noise are generated directly in the test flow gas. Noise is generated by the growing boundary layer in the test gas. The reflection of the shock from the end of the shock tube introduces significant noise into the stagnated test gas. He [127] recorded pressure traces in the stagnated test gas in the T4 shock tunnel. Large acoustic waves were observed in these traces, which were believed to result from the shock reflection process. As the shock travels back upstream it causes noise to be generated through its interaction with the boundary layer and the contact surface.

Noise is also introduced directly into the test flow through turbulence in the nozzle boundary layers [202]. The rupture of the secondary diaphragm should not have a significant effect on noise levels in the test flow as the affected flow is dominated by the nozzle startup waves.

Although a number of transition experiments have been performed in these facilities [93, 76, 1], little data is available on the effect of free stream noise in the test flow [202].

To address the issue of noise levels affecting the test flow, a number of 'quiet' supersonic and hypersonic wind tunnels have been developed [15]; however, it is also known that enthalpy can have a significant effect on boundary layer transition [1] and that these quiet facilities do not correctly simulate the high enthalpy associated with hypersonic flight.

2.2 The Experimental Study

Over the period from 1998 to 2000, Dr. David Buttsworth performed a series of experiments in the a shock tunnel located at the University of Queensland. These experiments were designed to investigate the flow development through a shock tunnel. The results from these experiments provide experimental data with which numerical simulations can be compared, ensuring that the numerical techniques and assumptions are appropriate. No test article was placed in the test section in these experiments (apart from the probes) as they were designed specifically with the aim of investigating the flow development within the shock tunnel. This section outlines the equipment used and the data obtained in these experiments.

2.2.1 The Drummond Tunnel

The experiments were conducted in the Drummond Tunnel facility, also known as the Small Shock Tunnel (SST) Facility, which is operated within the Centre for Hypersonics at the University of Queensland. It is a relatively low enthalpy shock tunnel, constructed by the Department of Defence in the 1970s in order to study chemical reactions occurring in shock heated gases [102]. It was originally operated in a straight-through configuration, but was re-developed into a reflected shock tunnel.

The details of the design and its operation as a reflected shock tunnel are described in two Department of Mechanical Engineering Reports: Austin et al. [7] and Craddock et al. [54]. The layout of the Drummond Tunnel is shown in Figure 2.11 and two photographs of the facility are shown in Figure 2.12. The whole tunnel is shown on the left, with the facility running from the bottom left of the picture to the test section and dumptank at centre right. A close-up of the nozzle and test section shown on the right from the same angle.

The facility consists of a cylindrical tube made up of a driver section, a shock tube section and an exchangable nozzle. The test section and dump tank enclose the downstream end of the nozzle. The driver section is 59 mm in diameter for 770 mm of length and is 74 mm in diameter for a further 230 mm. It is initially separated from the shock tube by a 1mm thick Aluminium diaphragm. The driver section contains a pneumatic cylinder with a shaft running along the centre-line of the driver which has a spike on the end. The shaft and spike is actuated with the pneumatic cylinder and is used to pierce the diaphragm, initiating the rupturing process and starting the experimental shot. The pneumatic cylinder fits inside 74 mm diameter part of the driver section and is surrounded by a volume occupied by additional driver gas. The shock tube is 62.2 mm diameter and is 3013 mm in length. Three nozzles are

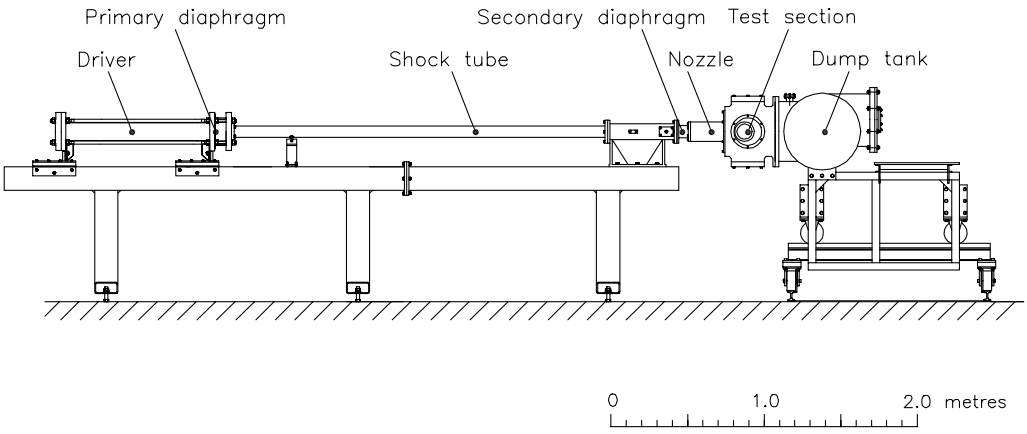

Figure 2.11: The layout of the Drummond Tunnel facility. Reproduced from Craddock et al. [54].

Figure 2.12: Photographs of the Drummond Tunnel facility taken during July 2002.

used in the facility, a conical Mach 4 nozzle, a conical Mach 7 nozzle and a contoured Mach 7 nozzle. The test section has optical access through four 100 mm diameter quartz windows. A dimensioned cross-sectional view of the Drummond Tunnel, with the detail of the driver section and spike, is shown in Figure 2.13.

The driver section is filled from bottled Helium or Nitrogen to a maximum absolute pressure of 6 MPa. The shock tube is filled with the test gas, which is usually Nitrogen or air. The test section and dump tank, initially containing atmospheric air, are evacuated before a test. A secondary diaphragm, made of cellophane or thin plastic, is used in the nozzle throat to separate the test gas from the shock tube. Under typical operating conditions, with He at 6 MPa driving Air at 20 kPa, the tunnel produces a supply enthalpy of 2 MJ/kg and a supply pressure of 2.2 MPa [242].

The Drummond tunnel provides a useful test-bed for investigating the flow development in a shock tunnel facility that:

1. is relatively simple, using a single diaphragm which is mechanically pierced,

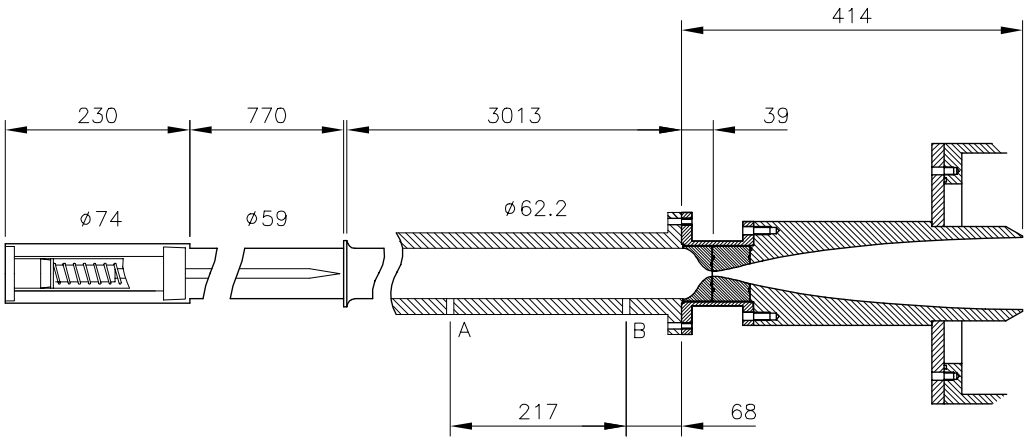

Figure 2.13: Cross section of the Drummond Tunnel facility with dimensions shown. The Mach 7 nozzle and the details of the driver section and piercer are shown. The Mach 4 nozzle used in the experiments is exchangable with this nozzle. Reproduced from Craddock et al. [54].

making the firing predictable, and its geometry is fully documented and accessible (including the nozzle profiles). It does not have a free-piston driver such as the larger T4.

2. is small in size, meaning that fine resolution of flow features can be achieved using a mesh with much fewer cells than would be required to model a large facility such as T4; mesh resolution is critical to accurate modelling of the boundary layers and shock interaction processes.

3. has accurate and reproducible experimental results that are available in this study. The experiments were conducted with a range of operating conditions, utilising an array of data acquisition equipment in the shock tunnel.

4. operates in a regime in which the limited stagnation temperatures reduce the dependency on other forms of molecular excitation, ionization, dissociation and other thermochemical effects [242]. Temperatures in the nozzle supply region are limited to below 2000 K. Although the numerical code used in this thesis is capable of modelling high temperature flows, the dependency on implementing finite rate chemistry models are removed. Modelling is also not hindered by the extreme pressure ratios found in some facilities, such as super-orbital expansion tubes.

5. is of interest in other research. For example, it is being used to develop new optical techniques and in rarefied flow research [242]). Being able to simulate this facility accurately will be of benefit to these projects.

The modelling techniques developed in this study can be extended to larger, more complicated reflected shock tunnel facilities, such as T4, and expansion tubes. It is also believed that the conclusions reached through simulations of this relatively low enthalpy facility are applicable to the flows in high enthaply facilities.

2.2.2 Operating Conditions

Three sets of experiments were performed using producing two different levels of tailoring: one resulting in over-tailored operation and the other in roughly tailored operation. Nitrogen was used as the test gas for all experiments. The over-tailored case used Nitrogen as the driver gas and the tailored case used Helium as the driver gas. Nitrogen driver experiments were conducted using a conical Mach 4 nozzle and also with the end of the shock tube blanked off. The Helium driver experiments were performed using a conical Mach 4 nozzle attached to the end of the shock tube. The three operating conditions are listed in Table 2.1.

Table 2.1: Drummond tunnel operating conditions.

Driver gas	Driven gas	Test section	Condition	Test attachment
3.25 MPa N_2	30.0 kPa N_2	0.4 kPa air	over-tailored	Mach 4 conical
3.20 MPa N_2	n/a	0.4 kPa air	over-tailored	blanked end
5.60 MPa He	61.4 kPa N_2	0.4 kPa air	approx. tailored	Mach 4 conical

The use of Nitrogen simplifies the modelling by remaining closer to the calorically perfect gas approximation over the range of temperatures used in these experiments. The use of Helium as a driver gas increases the driver to driven gas sound speed ratio. This is done in order to maximise the shock strength in the shock tube.

The ambient temperature during the experiments was usually around 23^{o}C, which was assumed to be a constant through all of the experiments; however, the actual driver gas and test gas temperatures in the experiments were not measured at the time. The driver and test gases are filled from high pressure gas bottles. The driver section is filled to high pressures and, due to throttling and heat transfer processes in the filling pipes, this process can result in significant temperature rise in the driven gas. A series of filling trials were used to investigate the temperature of the driver gas that was used in the tailored experiments. The driver section was filled with Helium, varying the speed at which the section was filled from the gas bottle. It was found that the process of filling the driver section with Helium to 5.60 MPa caused the driver gas to reach a peak temperature of around 44^{o}C at the

completion of filling. There was a delay between the end of the filling process and when the shot was fired and during this time heat was conducted away from the hot driver gas to the ambient driver tube walls, decreasing the temperature of the driver gas over time. The driver temperatures decreased to about 37^oC during this delay.

The temperatures used in the experiment were not known exactly and the simulated pressures were varied, for both driver gas conditions, over the range from 30^oC to 40^oC to obtain the temperatures ultimately used in the simulations. This selection procedure is described in Section 7.2.

The Nitrogen test gas in the shock tube was filled slowly and to only modest pressures and so any temperature rise would be negligible. Additionally, the driven section was filled before the driver meaning a substantial delay before the experiment took place. In the simulations, this gas was assumed to be at ambient conditions.

2.2.3 Experimental Data Acquisition

For the experiments conducted in this study, the Drummond Tunnel was operated with no test article in the test section; however, operational and test flow data was recorded with the aim of investigating the flow development process. More details of the experimental data acquisition procedures are provided in Buttsworth and Jacobs [31, 32].

Schematics of the end region of the shock tunnel, with the locations of the data acquisition equipment, are shown in Figure 2.14.

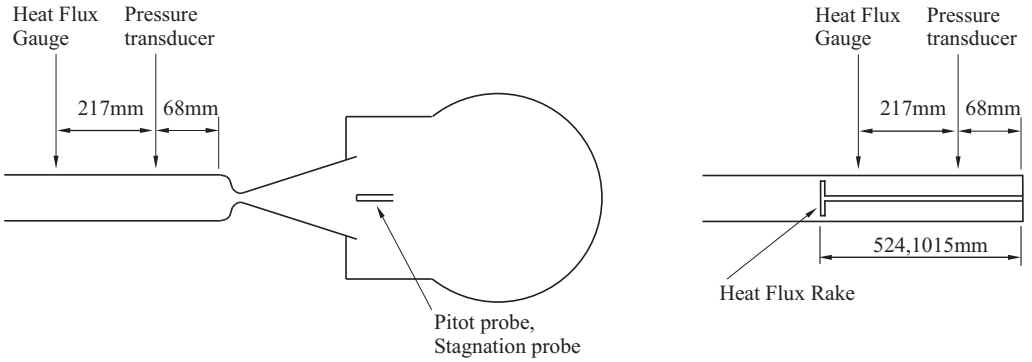

Figure 2.14: Schematic of the end region of the shock tunnel with the locations of the data acquitition equipment : Mach 4 nozzle attached (left) and blanked shock tube end (right). Reproduced from Buttsworth and Jacobs [32].

For all experiments, data was recorded at two positions on the wall of the shock tube:

1. a piezoelectric transducer located 68 mm upstream from the nozzle attachment. This transducer provides a time accurate record of pressures in the nozzle supply region. The incident shock and reflected shock (stagnation) pressures are recorded, as well as the times of arrival of other waves, such as the reflected expansion and tailoring waves.

2. a thin film transient heat flux gauge located 285mm from the nozzle attachment. This gauge records a detailed history of the characteristics of the boundary layer and can be used to identify the arrival of the incident shock, the contact surface and the passage of the reflected shock after its interaction with the contact surface.

The time of arrival of the incident shock at the two positions is used to determine the shock speed as it approaches the end of the shock tube. This is an important factor for comparison with the simulations as it demonstrates the ability of the simulations to reproduce the level of viscous attenuation of the shock as it progresses along the tube.

For the experiments conducted with the end of the tube blanked off, a rake of heat flux probes on a sting was inserted into the tube through the blanked end. This sting and rake are shown in Figure 2.15. This rake can be positioned at various axial positions along the tube for different experimental shots. These heat flux probes were used to detect the arrival of the driver gas, through the change in stagnation point heat flux, at points across the diameter of the tube. Geometrically, this rake represents a ten percent blockage of the shock tube flow. This rake allowed measurement of the shape of the contact surface at these axial and radial positions, and assuming the reproducibility of the shots, was used to quantify the evolution of the contact surface along the tube length. The speed of the contact surface as it moves along the tube was also found from its time of flight between these stations.

For the experiments with the nozzle attached, data was recorded in the test flow using:

1. a piezoelectric pressure transducer (of a similar type to the supply pressure transducer) mounted on the centreline of the nozzle exit. This probe measured the pitot pressure on the centre-line of the test flow for the duration of the test. For the experiments with the Nitrogen driver, the pitot probe utilised a pneumatic cavity to shield the pressure transducer from the shock tunnel flow, whereas for the experiments with the Helium driver the pitot pressure transducer was protected from the flow using a thin layer of RTV rubber [32]. The Helium driver pitot probe configuration yielded pitot pressure measure-

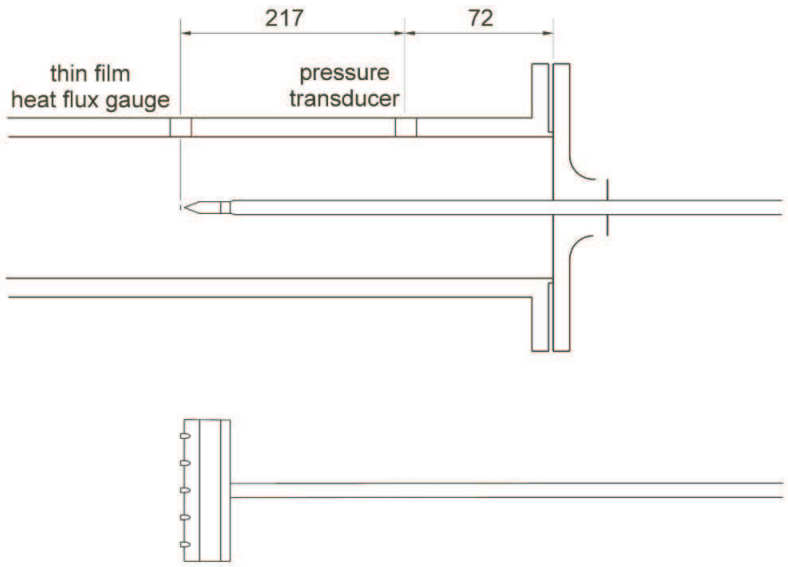

Figure 2.15: Diagram of the rake of heat flux gauges inserted on a sting through the blanked end of the tube. Reproduced from Buttsworth [30].

ments to a much higher bandwidth than the Nitrogen driver and with resonant frequency of 230 kHz.

2. a rake of five thermocouple probes (three of which produced meaningful data) across the radius of the nozzle exit flow. This rake was used to measure details about the test flow, and its variation, across the radius test flow.

3. a stagnation temperature probe at the nozzle exit for the Nitrogen driver case. This probe was mounted 14 mm from the centreline and was simultaneously with the pitot probe. which is interchangable with the pitot probe and was used for Nitrogen driving Nitrogen experiments.

This arrangement is shown in Figure 2.16. For the Helium driver case, these probes were axially located on the plane of the nozzle exit and for the Nitrogen driver case, were located 14mm downstream from the nozzle exit plane.

2.2.4 Experimental Results

The results obtained from these experiments are documented and described here, and are used in Section 7.2 for validation of the numerical simulations. The times for the experimental and simulated traces were all set at zero at the point in time of the arrival of the incident shock at the supply pressure transducer.

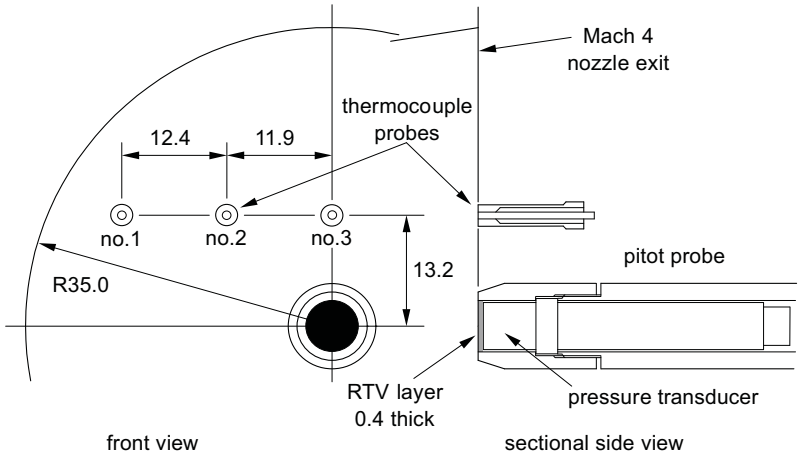

Figure 2.16: Centre-line pitot probe and thermocouple rake which was located in the nozzle test flow. Reproduced from Buttsworth and Jacobs [32].

Supply Pressure Transducer

This transducer measures the pressure in the region which acts as the reservoir for the test flow and so correctly predicting the properties of this region is vital to the model.

1. Nitrogen driving Nitrogen. Blanked end (Figure 2.17)
The pressure is initially at the driven gas fill pressure. The pressure rises rapidly due to the passage of the incident shock and rises straight to the post shock pressure. The pressure behind this shock is steady until the reflected shock arrives, travelling upstream. The pressure rise due to this shock has a slight kink in it. This is due to the passage of the bifurcated foot of the reflected shock over the transducer. For 850 μs, the test gas at the transducer is largely stagnated; however, during this time there is a slight steady rise in pressure caused by the interaction between the reflected shock and the boundary layer. Noise is evident in the trace during this region, at a higher level to that behind the incident shock.

In this mode of operation, the shock tunnel is over tailored and the arrival of the tailoring waves resulting from the interaction of the reflected shock with the contact surface appear as the steady increase in pressure at the end of the steady period. Two distinct waves are evident, with a small plateau between ; these are the tailoring wave, and its reflection from the blanked end. This reflected tailoring wave further increases the pressure. With this over-tailored mode of operation, the driver gas continues to move downstream after its interaction with the reflected shock. This means that the driver gas passes this transducer, appearing on the trace as a significant increase in noise around 1.7 ms after the incident shock. The expansion reflected from the upstream end of the driver section is seen as the final gradual

decrease in the gradient in pressure, arriving around 4 ms after the incident shock.

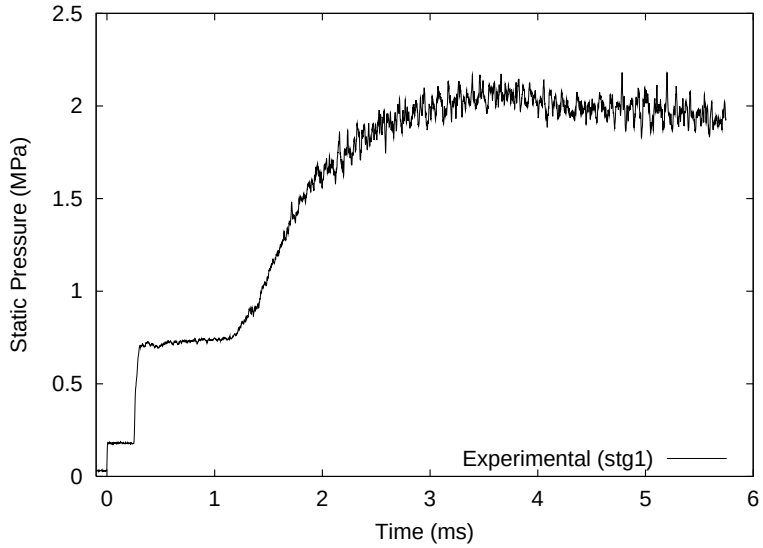

Figure 2.17: Experimentally measured trace from the supply pressure PCB transducer for the Nitrogen driving Nitrogen case with the blanked shock tube end.

2. Nitrogen driving Nitrogen. Mach 4 Nozzle (Figure 2.18)

The operating conditions for this case are approximately the same as for the blanked end case. The difference in this case is that the Mach 4 nozzle is attached to the end of the shock tube. The reflection from the throat of the nozzle results in a more complicated reflection process than from the blanked wall, and once the secondary diaphragm ruptures, the test gas is drained through the nozzle from the stagnated region. This case shows a significantly more defined kink in the pressure rise and a step at the arrival of the reflected shock. For 350 μs, the test gas at the transducer is stagnated, which is a much shorter period than for the blanked end case. During this time there is also a steady rise in pressure. The plateau between the tailoring wave and its reflection is more defined and contains a dip in pressure. No significant amount of the reflected expansion had arrived by the end of the recorded time.

3. Helium driving Nitrogen. Mach 4 Nozzle (Figure 2.19)

The pressure is initially at the driven gas filling pressure. The pressure rises rapidly due to the passage of the incident shock to the post shock pressure. There is an early dip in pressure and noise evident in the flow behind the shock. The pressure behind this shock is steady until the reflected shock arrives upstream. There is a noticeable kink followed by a step in the pressure trace from this shock, indicating the passage of the bifurcated foot of the reflected shock over the transducer. For 340 μs, the test gas at the transducer is stagnated. In this tailored mode of operation, the majority of the driver gas is stopped by the reflected shock before this transducer.

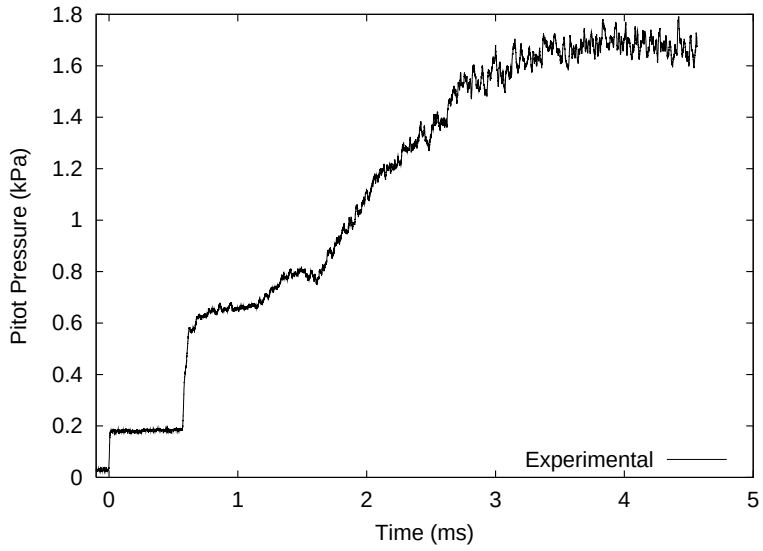

Figure 2.18: Experimentally measured trace from the supply pressure PCB transducer for the Nitrogen driving Nitrogen case with the Mach 4 nozzle.

The tailoring waves are weak and do not appear on this trace. This steady period is ended by the arrival of the reflected expansion from the upstream end of the driver section. This expansion arrives in the stagnated gas as a gradual decrease in pressure.

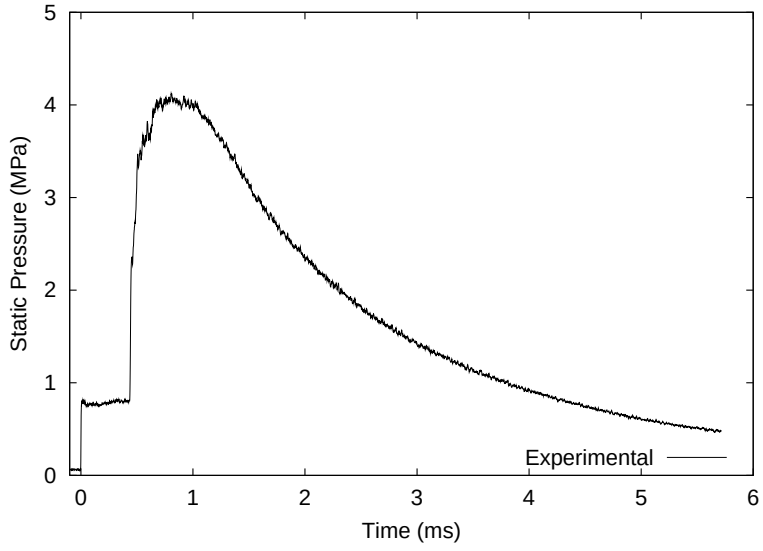

Figure 2.19: Experimentally measured trace from the supply pressure PCB transducer for the Helium driving Nitrogen case with the Mach 4 nozzle.

Heat Flux Gauge

This gauge measured the amount of heat transferred from the gas to the wall, through the boundary layer. The gauge was not calibrated and so the heat flux has an arbitrary scale.

1. Nitrogen driving Nitrogen. Blanked end (Figure 2.20)
Both the gas in the driven section and the walls are initially at ambient conditions and, therefore, there is initially zero heat flux at the gauge. This gauge is upstream of the pressure transducer and, with the zero time aligned with the arrival of the incident shock at the pressure transducer, it can be seen to arrive before the zero time. The arrival of the incident shock is seen as a sharp increase in heat flux. The heat flux from the gas behind the incident shock is roughly steady. This steady heat flux is ended by the arrival of the contact surface travelling behind the shock. The driver gas behind the contact surface is colder than the driven gas and so as the contact surface arrives, the heat flux deceases gradually as the concentration of driver gas increases. As the contact surface is still passing over the heat flux gauge, the reflected shock arrives at the gauge. This shock is responsible for the sharp jump in heat flux, after which the heat flux continues to decrease, as before, with more of the contact surface arriving. The tailoring waves, resulting from the interaction of the reflected shock with the contact surface, can be seen as the waves following the arrival of the contact surface. Being overtailored, the tailoring waves are quite strong and result in the significant fluctuations seen in this trace for the remainder of the time. The heat flux returns towards zero as the gases in the tube approach ambient temperature.

2. Nitrogen driving Nitrogen. Mach 4 Nozzle
The heat flux trace from this case was unusable due to excessive experimental noise.

3. Helium driving Nitrogen. Mach 4 Nozzle (Figure 2.21)
Both the gas in the driven section and the walls are initially at ambient conditions and, therefore, there is initially zero heat flux at the gauge. The arrival of the incident shock is seen as a sharp increase in heat flux; this shock arrives before the zero time. The heat flux from the gas behind the incident shock is roughly steady. This steady heat flux is ended by the arrival of the contact surface travelling behind the shock. The driver gas behind the contact surface is colder than the driven gas and so as the contact surface arrives, the heat flux deceases gradually as the concentration of driver gas increases. As the contact surface is still passing over the heat flux gauge, the reflected shock arrives at the gauge. This shock is responsible for the sharp jump in heat flux, after which the heat flux continues to decrease, as before, with more of the contact surface arriving. The tailoring waves can be seen

as the peaks in heat flux following this. These wave have much less energy than the reflected shock and so these peaks are small. As the gas in the shock tube settles and the remaining test gas mixes into the cold driver gas, the heat flux tapers away to zero.

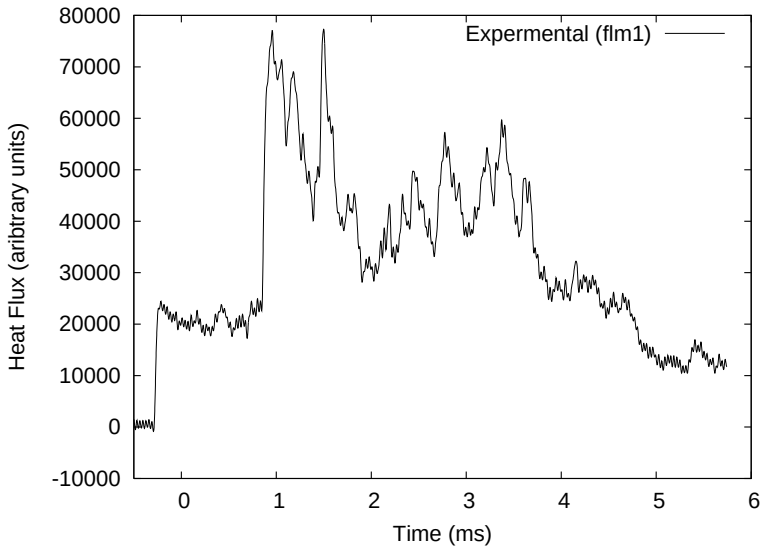

Figure 2.20: Experimentally measured heat flux from the heat flux gauge for the Nitrogen driving Nitrogen case with the blanked shock tube end.

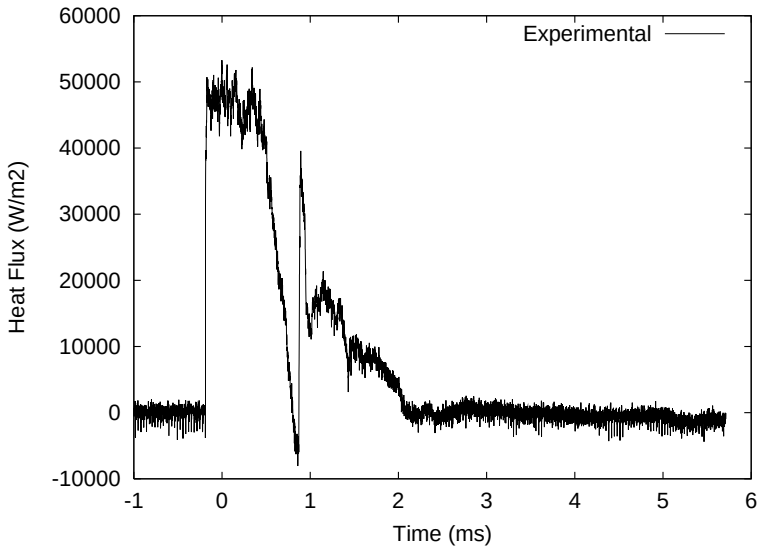

Figure 2.21: Experimentally measured heat flux from the heat flux gauge for the Helium driving Nitrogen case with the Mach 4 nozzle.

Test Flow Pitot Probe

The pitot pressure probe which was positioned on the centreline of the test flow provides valuable information about the test flow. As well as allowing identification of the test time and the mean flow pitot pressure, this probe is also used to record the fluctuations in the flow. In the Helium driving Nitrogen case the high bandwidth configuration was used. The experimental traces were filtered using a moving average.

1. Nitrogen driving Nitrogen. Mach 4 Nozzle (Figure 2.22)
With no flow through the nozzle, the pitot pressure is initially zero. The nozzle flow starting process begins with the rupture of the secondary diaphragm, resulting in an initial flow of the test gas through a series of transient shocks and a contact surface that moves through the nozzle. The nozzle startup waves arrive at the transducer, recorded by the probe as they focus on the centreline. These waves took 368μs to pass the transducer. After the startup waves have passed, the test time begins, with the steady expansion of the stagnated test gas through the nozzle. The test time was measured to have a duration of $724\,\mu$s. The tailoring waves, from the interaction of the reflected shock with the contact surface, arrive in the nozzle reservoir region, as recorded in the supply pressure transducer. These compression waves travel through the nozzle and result in the pressure increase which ends the test time. There is a dip in the pitot pressure, followed by a rapid rise to a higher level as the driver gas arrives. After this time, the pitot pressure remains constant. The arrival of the driver gas in the test flow is indicated by a significant increase in the noise level in the test flow.

2. Helium driving Nitrogen. Mach 4 Nozzle (Figure 2.23)
The pitot pressure is initially zero. The nozzle startup waves can be seen to arrive at the transducer, taking $378\,\mu$s to pass the transducer. The test time was measured to have a duration of $356\,\mu$s and begins as the startup waves gradually taper off. The test time is ended by the arrival of the reflected expansion from the driver section into the test flow. This expansion can be seen as the gradual decrease in pitot pressure as the pressure in the nozzle supply region decreases. The arrival of driver gas seems to be indicated by a decrease in the noise level, unlike the Nitrogen driver case.

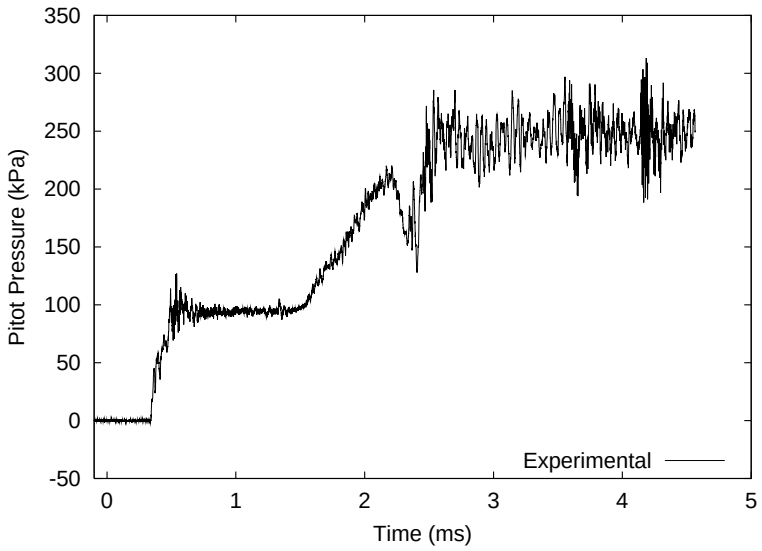

Figure 2.22: Experimentally measured trace from the nozzle exit centreline pitot probe for the Nitrogen driving Nitrogen case with the Mach 4 nozzle.

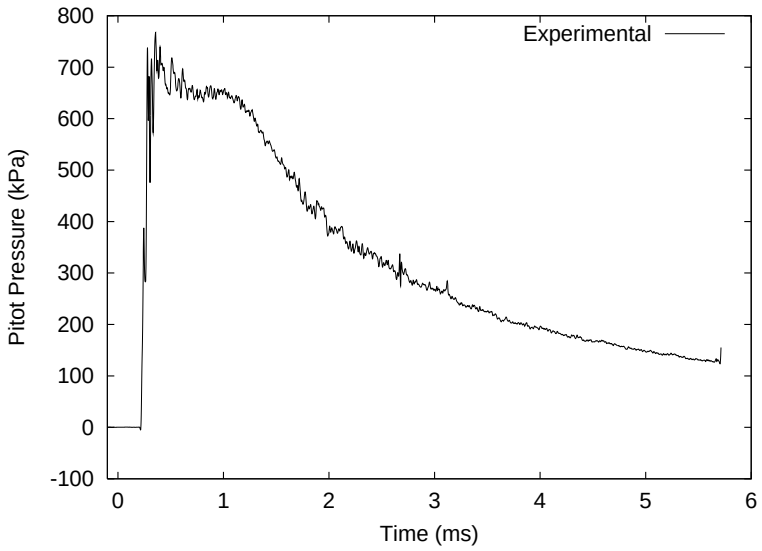

Figure 2.23: Experimentally measured trace from the nozzle exit centreline pitot probe for the Helium driving Nitrogen case with the Mach 4 nozzle.

Shock Tube Heat Flux Rake

The heat flux rake on a sting was inserted through the blanked end of the shock tube and, therefore, recorded data for the Nitrogen driving Nitrogen case. The rake records the stagnation point heat flux at five positions across the tube. Measurements were obtained at four axial stations; however, only the two upstream stations provided data on the contact surface because the reflected shock interferred with measurements at the two downstream stations. An example trace, from the most upstream position, is shown in Figure 2.24. In this plot, the arrival of the incident shock can be seen as the jump in heat flux. The heat flux from the flow behind the shock gradually increases in the time before the contact surface arrives. The contact surface arrival can be seen as the decrease in heat flux resulting from the colder driver gas. The noise increase associated with the arrival of driver gas can be seen during this time. The heat flux decreases back to approximately zero with the arrival of the cold driver gas.

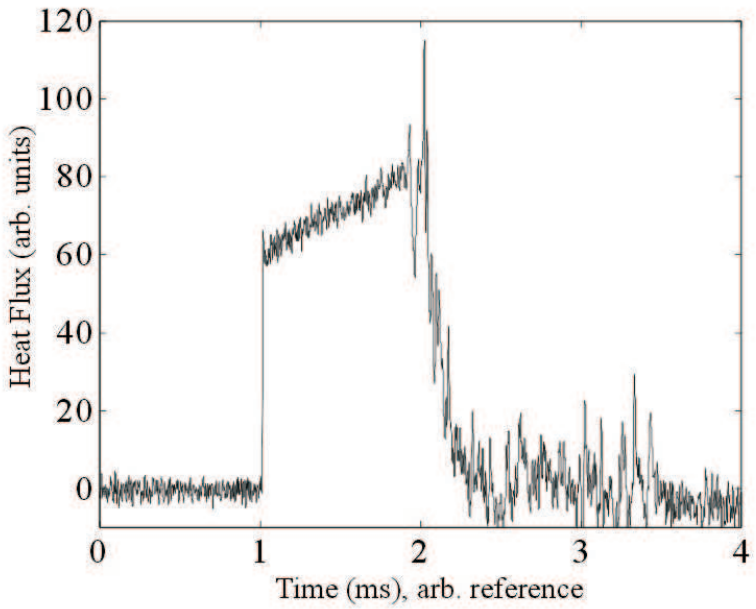

Figure 2.24: Heat Flux data for the contact surface arrival at the 524 mm position.

This data is used to find the 10% and 90% values for the decrease in stagnation point heat flux associated with the contact surface arrival. These values are used for the arrival times of the start and the end of the contact surface at their relative axial and radial positions. This data not only provides a record of the arrival time of the contact surface along with its shape, but also a measure of the amount of diffusion that has occurred at the interface along the tube. The data for the contact surface arrival at the 524 mm position is shown in Figure 2.25 and the 1015 mm position in

Figure 2.26. Points are shown for the experimental data, and smooth bezier curves have been fitted through each of the traces to provide a better indication of the shape.

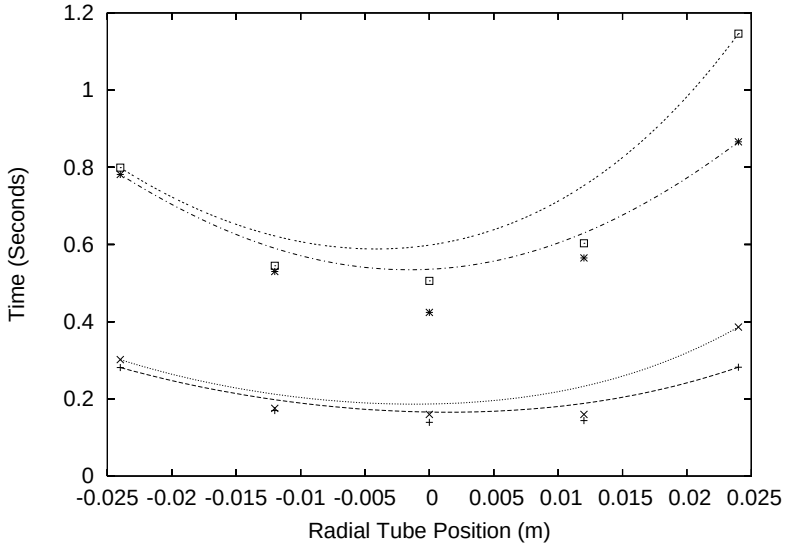

Figure 2.25: Contact surface shape for the 524 mm axial position.

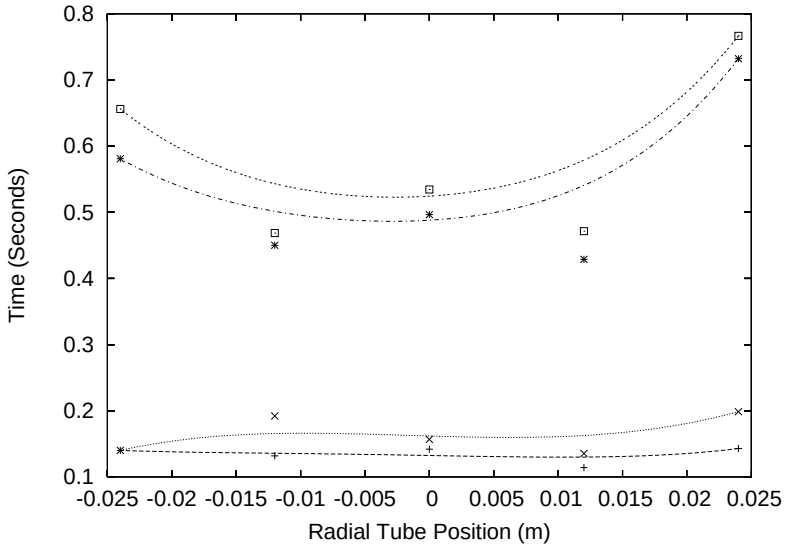

Figure 2.26: Contact surface shape for the 1015 mm axial position.

Test Flow Stagnation Probe

Figure 2.27 shows the stagnation temperature recorded by the stagnation probe located on the nozzle exit plane, 14 mm radially from the centreline. This trace was recorded for the Nitrogen driving Nitrogen case. This trace is useful in that the large change in temperature recorded between 2 ms and 2.5 ms indicates a change in gas entropy associated with the arrival of driver gas in the flow. The segment of the recorded trace that is associated with the driver gas arrival is shown in the figure.

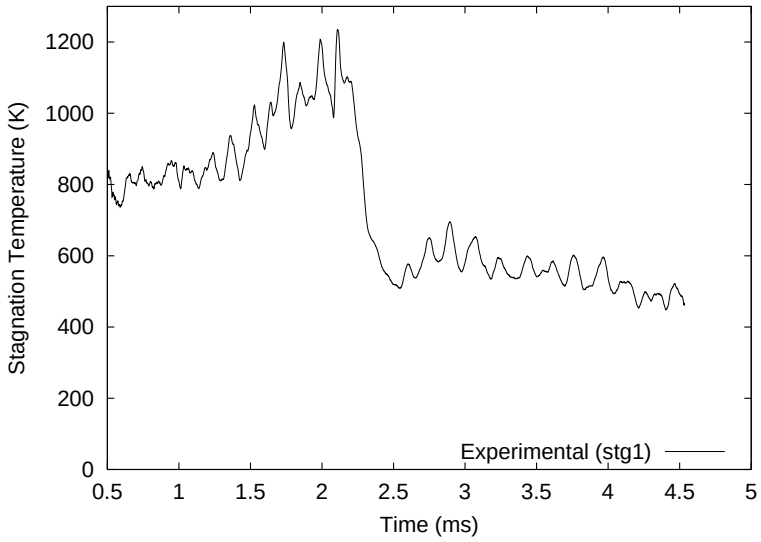

Figure 2.27: Experimentally measured trace from the nozzle exit centreline stagnation heat flux gauge for the Nitrogen driving Nitrogen case with the Mach 4 nozzle.

CHAPTER 3

Numerical Formulation for Compressible Flow Simulation

Computational Fluid Dynamics (CFD) is based on the numerical solution of the continuity equation (describing the conservation of mass), the momentum equation (describing Newton's Second Law) and the energy equation (describing the conservation of energy). These governing equations can be obtained in various different forms [116].

The analytical study of fluid mechanics often uses two descriptions of fluid flow: the Eulerian description and the Lagrangian description. Methods based on the Eulerian description discretise the geometry of the flow domain into stationary computational elements through which the fluid flows. Methods based on the Lagrangian description discretise the fluid itself. This means that the computational elements move with the fluid through the domain. The majority of computational methods that have been used for multi-dimensional modelling of impulse facilities are Finite Volume methods based on the Eulerian description; however, the Lagrangian description could provide certain advantages in the representation of the contact surface. Figure 3.1 shows an example problem of a circle of one fluid entering another fluid. The Eulerian representation of this scenario is shown on the left and the Lagrangian representation is shown on the right.

Numerical methods based on the Eulerian description may be susceptible to numerical forms of diffusion, especially noticeable in regions with large gradients. When a flux of material enters a cell, its characteristics are uniformly mixed with those of the cell that it enters [165]. When a fluid interface crosses through the interior of cell, the properties of the gases on either side of the interface occupy the same cell. Numerical diffusion in Eulerian methods can be reduced through the use of techniques such as Flux Corrected Transport (FCT) [22] and is minimised in modern high resolution schemes. On the other hand, numerical methods based on the Lagrangian description may provide advantages in modelling flows in which interfaces are important by providing a more natural representation of these inter-

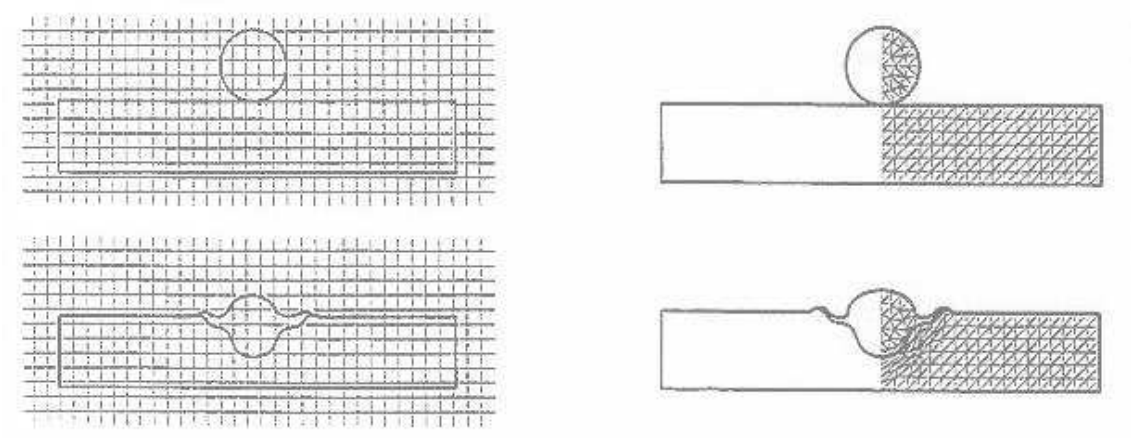

Figure 3.1: A comparison of an Eulerian mesh and a Lagrangian mesh for an example of a circle of fluid entering another fluid. Reproduced from Zukas [256].

faces. In the Lagrangian description, elements of fluid on one side of an interface remain on that side of the interface as the fluid moves.

This chapter will discuss the implementation of CFD in two numerical techniques. The CFD code MB_CNS [114] is based on a finite volume formulation of the Navier-Stokes equations. These equations are in the form based on the Eulerian description of fluid motion. As an alternative technique, the Smoothed Particle Hydrodynamics (SPH) [78, 137] will be investigated. This numerical technique uses a particle based approach to modelling the Lagrangian form of the governing equations. Despite the fact that the two descriptions and their implementations are very different, they are both still modelling the same equations and in doing so performing the same role. It is shown in Anderson [116] that the two methods solve different forms of the same equations.

3.1 Finite-Volume Eulerian Methods

cThe discretisation of the flow using the Eulerian description mcns that the computational mesh must be fitted to the geometry containing the fluid. In the numerical techniques used in this section, the flow is discretised using a body-fitted computational mesh.

3.1.1 Multi-Block Compressible Navier-Stokes Solver

This section is based on the Department of Mechanical Engineering Technical Report 10/96 [114]. The program *MB_CNS* is a CFD tool for the simulation of transient compressible flow in two-dimensional (planar or axisymmetric) geometries. The code is intended primarily for the simulation of the transient flows experienced in shock tunnels and expansion tubes and many features are included in the code specifically for this aim.

The present code is a development of the single-block Navier-Stokes integrator *CNS4U* [110] with the primary difference being the ability to handle a relatively complex flow geometry by decomposing it into several non-overlapping blocks. The name MB_CNS is an acronym for Multiple-Block Compressible Navier-Stokes solver. Further details on the code are available in Jacobs [114].

MB_CNS is based on a cell-centred finite-volume formulation of the Navier-Stokes equations and has a shock-capturing capability through the use of a limited reconstruction scheme and an upwind-biased flux calculator. The governing equations are expressed in integral form over arbitrary quadrilateral cells with the time rate of change of conserved quantities in each cell specified as a summation of the fluxes (of mass, momentum and energy) through the cell interfaces. Subject to grid resolution and numerical diffusion issues, the code is capable of modelling flows that include shocks, expansions, shear layers and boundary layers.

The code is written in C and uses data structures such that entire blocks of cells are packaged in single data structure. The functions were then written so that they operated on the flow data contained within that data structure without the need for other information. The multi-block formulation of MB_CNS allows solution of the flow field in separate blocks in parallel. Parallelisation in MB_CNS is described in Chapter 5.2.2. The flow domain is specified as a Bezier polyline description of the block boundaries.

In MB_CNS, simulations are assigned a case specific identification number, which allows code to be included that performs tasks specific to that simulation. This code is included in the files `mb_special_init.inc` and `mb_special_step.inc`, which are

included in the `mb_cns.c`. Code used to set up special conditions specific to a case identification number is written in the file `mb_special_init.inc` and code that is used at the start of each time step is written in the file `mb_special_step.inc`. This files are provided in Appendix C.

3.1.2 Formulation

The formulation of the governing equations, the flow field discretisation as a single block of cells and the time-stepping scheme used to integrate the discrete equations is described in Jacobs [110]. Some details of the numerical techniques used in MB_CNS are discussed in this section.

Governing Equations

The starting point for the governing equations encoded within MB_CNS is the set of Navier-Stokes equations which, in integral form, can be expressed as:

$$\frac{\partial}{\partial t}\int_V U dV = -\int_S (\overline{F}_i - \overline{F}_v)\cdot \hat{n}\; dA + \int_V Q\; dV \quad , \tag{3.1}$$

where V is the cell's volume, S is the bounding (control) surface and $\hat{n}$ is the outward-facing unit normal of the control surface. For two-dimensional flow, V is the volume per unit depth in the z-direction and A is the area of the cell boundary per unit depth in z. The array of conserved quantities (per unit volume) is:

$$U = \begin{bmatrix} \rho \\ \rho u_x \\ \rho u_y \\ \rho E \\ \rho f_{is} \end{bmatrix} \quad . \tag{3.2}$$

These elements represent mass density, x-momentum per volume, y-momentum per volume, total energy per volume and mass density of species is. The flux vector is divided into inviscid and viscous components. The inviscid component, in two dimensions, is:

$$\overline{F}_i = \begin{bmatrix} \rho u_x \\ \rho u_x^2 + p \\ \rho u_y u_x \\ \rho E u_x + p u_x \\ \rho f_{is} u_x \end{bmatrix} \hat{i} + \begin{bmatrix} \rho u_y \\ \rho u_x u_y \\ \rho u_y^2 + p \\ \rho E u_y + p u_y \\ \rho f_{is} u_y \end{bmatrix} \hat{j} \quad . \tag{3.3}$$

The viscous component is:

$$\overline{F}_v = \begin{bmatrix} 0 \\ \tau_{xx} \\ \tau_{yx} \\ \tau_{xx}u_x + \tau_{yx}u_y + q_x \\ \rho f_{is}\mu_{x,is} \end{bmatrix} \hat{i} + \begin{bmatrix} 0 \\ \tau_{xy} \\ \tau_{yy} \\ \tau_{xy}u_x + \tau_{yy}u_y + q_y \\ \rho f_{is}\mu_{y,is} \end{bmatrix} \hat{j} \quad , \tag{3.4}$$

where the viscous stresses are:

$$\begin{aligned} \tau_{xx} &= 2\mu\frac{\partial u_x}{\partial x} + \lambda\left(\frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y}\right) \quad , \\ \tau_{yy} &= 2\mu\frac{\partial u_y}{\partial y} + \lambda\left(\frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y}\right) \quad , \\ \tau_{xy} = \tau_{yx} &= \mu\left(\frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x}\right) \quad , \end{aligned} \tag{3.5}$$

and the viscous heat fluxes are:

$$\begin{aligned} q_x &= k\frac{\partial T}{\partial x} + \rho\sum h_{is} f_{is}\mu_{x,is} \quad , \\ q_y &= k\frac{\partial T}{\partial y} + \rho\sum h_{is} f_{is}\mu_{y,is} \quad . \end{aligned} \tag{3.6}$$

Currently, the code convects species without considering their diffusion (*i.e.* $\mu_{x,is} = 0$, $\mu_{y,is} = 0$). For flow without heat sources or chemical effects, the source terms in Q are set to zero.

The conservation equations are supplemented by the equation of state giving pressure as a function of density, specific internal energy and species mass fractions:

$$p = p(\rho, e, f_{is}) \quad . \tag{3.7}$$

The coefficients of viscosity μ, λ and heat conduction k are also allowed to vary with the fluid state.

Axisymmetric Geometries

The shock tunnel simulations used in Chapter 7 use an axisymmetric representation of the geometry of the shock tunnel. For axisymmetric flow, the geometry is defined such that x-axis is the axis of symmetry and y is the radial coordinate. The governing equations are modified such that:

- dA is now computed as interface area per radian;

- dV is now cell volume per radian;
- The shear stresses τ_{xx}, τ_{yy} have a extra term so that:

$$\begin{aligned} \tau_{xx} &= 2\mu\frac{\partial u_x}{\partial x} + \lambda\left(\frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} + \frac{u_y}{y}\right) \quad , \\ \tau_{yy} &= 2\mu\frac{\partial u_y}{\partial y} + \lambda\left(\frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} + \frac{u_y}{y}\right) \quad , \end{aligned} \tag{3.8}$$

- and there is a pressure and shear-stress contribution to the radial momentum equation which can be expressed as an effective source term:

$$Q = \begin{bmatrix} 0 \\ 0 \\ (p - \tau_{\theta\theta})A_{xy}/V \\ 0 \\ 0 \end{bmatrix} \quad , \tag{3.9}$$

where A_{xy} is the projected area of the cell in the (x, y)-plane and:

$$\tau_{\theta\theta} = 2\mu\frac{u_y}{y} + \lambda\left(\frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} + \frac{u_y}{y}\right) \quad . \tag{3.10}$$

Discretised Equations and Flux Calculation

The conservation equations are applied to each finite-volume cell for which the boundary, projected onto the (x, y)-plane, consists of four straight lines. These lines (or cell interfaces) are labelled North, East, South and West and the integral equation is approximated as the algebraic expression:

$$\frac{dU}{dt} = -\frac{1}{V}\sum_{NESW}(\overline{F}_i - \overline{F}_v)\cdot\hat{n}\ dA + Q \quad , \tag{3.11}$$

where U and Q now represent cell-averaged values. The code updates the cell-average flow quantities each time step by:

1. applying inviscid boundary conditions or exchanging data at boundaries of each block as appropriate;
2. reconstructing (or interpolating) the flow field state to both sides of each interface;
3. computing the inviscid fluxes at interfaces as $(\overline{F}_i \cdot \hat{n})$using a one-dimensional flux calculator such as a Riemann solver [112], the equilibrium-flux method (EFM) [180, 139] or AUSM [135];

4. applying viscous boundary conditions at solid walls;

5. computing the viscous contribution to the fluxes as $(\overline{F}_v \cdot \hat{n})$; and finally

6. updating the cell-average values using equation (3.11).

This whole process is applied in two stages if predictor-corrector time stepping is specified.

When computing the inviscid fluxes at each interface, the velocity field is rotated into a local (n, p)-coordinate system with unit vectors:

$$\begin{aligned} \hat{n} &= n_x \, \hat{i} + n_y \, \hat{j} \quad , \\ \hat{p} &= p_x \, \hat{i} + p_y \, \hat{j} \quad , \end{aligned} \tag{3.12}$$

normal and tangental to the cell interface respectively. We have chosen the tangential direction $p_x = -n_y$ and $p_y = n_x$. The normal and tangential velocity components:

$$\begin{aligned} u_n &= n_x \, u_x + n_y \, u_y \quad , \\ u_p &= p_x \, u_x + p_y \, u_y \quad , \end{aligned} \tag{3.13}$$

are then used, together with the other flow properties either side of the interface, to compute the fluxes:

$$\begin{bmatrix} F_{mass} \\ F_{n-momentum} \\ F_{p-momentum} \\ F_{energy} \\ F_{species-is} \end{bmatrix} = \begin{bmatrix} \rho u_n \\ \rho u_n u_n + p \\ \rho u_n u_p \\ \rho u_n E + p u_n \\ \rho u_n f_{is} \end{bmatrix} \quad , \tag{3.14}$$

in the local reference frame. These are then transformed back to the (x, y)-plane as:

$$\overline{F} \cdot \hat{n} = \begin{bmatrix} F_{mass} \\ F_{x-momentum} \\ F_{y-momentum} \\ F_{energy} \\ F_{species-is} \end{bmatrix} = \begin{bmatrix} F_{mass} \\ F_{n-momentum} n_x + F_{p-momentum} p_x \\ F_{n-momentum} n_y + F_{p-momentum} p_y \\ F_{energy} \\ F_{species-is} \end{bmatrix} \quad . \tag{3.15}$$

Flux Solvers

A range of inviscid flux calculators are available in MB_CNS; however, the `adaptive` flux calculator was used throughout the simulations in this thesis. This flux calcula-

tor uses a switching function to choose between using the AUSMDV flux calculator of Wada and Liuo [238, 135] and the Equilibrium Flux Method of Macrossan [139]. This switching function sets the flux calculator to EFM where large compressive gradients in velocity are detected, otherwise, the flux calculator defaults to using the AUSMDV flux calculator. This adaptive flux calculator uses the good stability properties of EFM in regions, near discontinuities in the flow, that can cause problems with some flux calculators, as described by Quirk [182, 183]

Gas Properties

The gases simulated in MB_CNS may consist of several components (or species). Specific gas models can be included in MB_CNS through the use of the `gas.c` module. Access is provided by the function that calculates the equation of state, which uses values for ρ, e and f_{is} to compute the other thermodynamic properties T, p, a, and the viscous transport coefficients μ and k.

In chapter 6, the gas models used in MB_CNS are: an ideal gas mixture of Helium and air, and an ideal gas mixture of Refrigerant-22 and air. In chapter 7, the gas models used in MB_CNS are: an ideal mixture of Helium and Nitrogen, and a look-up table model of Nitrogen gas properties produced using the CEA program [39].

Ideal air is modelled as a perfect gas, having the equation of state:

$$p = \rho\, e\, (\gamma - 1) \qquad \text{Pa} \quad , \tag{3.16}$$

where $\gamma = C_p/C_v = 1.4$ is the ratio of specific heats, density ρ is given in kg/m^3 and specific internal energy e is given in J/kg. Temperature and speed of sound are also determined as:

$$T = \frac{e}{C_v} \quad {}^{o}\text{K} \quad , \tag{3.17}$$

$$a = (\gamma R T)^{\frac{1}{2}} \quad \text{m/s} \quad , \tag{3.18}$$

where:

$$\begin{aligned} R &= 287 \text{ J/(kg.K)}, \\ C_v &= \frac{R}{(\gamma - 1)} = 717.5 \text{ J/(kg.K)}, \text{ and} \\ C_p &= C_v + R = 1004.5 \text{ J/(kg.K)} \ . \end{aligned}$$

The viscous transport coefficients are computed as:

$$\mu = 1.458 \times 10^{-6} \frac{T^{3/2}}{(T + 110.4)} \quad \text{Pa.s} \quad , \tag{3.19}$$

$$\lambda = \frac{-2}{3}\mu \quad \text{Pa.s} \quad , \tag{3.20}$$

$$k = \frac{\mu C_p}{Pr} \quad \text{W/(m.K)} \quad , \tag{3.21}$$

where $Pr = 0.72$ is the Prandtl number. The properties of the other ideal gases used in these simulations, such as the Helium, Nitrogen and Refrigerant-22, can be specified and used with these equations. A finite-rate chemistry module [52] is being developed, which will extend the applicability of MB_CNS into thermochemically active regimes.

A Sutherland's law viscosity model [230] is used in calculating the viscous fluxes. For the Helium driving Nitrogen case, Wilke's law [245] is used to calculate the viscous fluxes where mixtures of different gas species are present. This model uses the relative mass fractions of each of the component gases in the computational cells.

Mixtures of Two Perfect Gases

The thermodynamic properties of a mixture of two perfect gases are computed using effective gas constants and specific heats, which are based on the mass fractions of the gases present in each cell:

$$R = f_a \, R_a + f_b \, R_b \quad , \tag{3.22}$$

$$C_v = f_a \, C_{va} + f_b \, C_{vb} \quad , \tag{3.23}$$

$$C_p = f_a \, C_{pa} + f_b \, C_{pb} \quad , \tag{3.24}$$

where f_a and f_b are the mass fractions of the gas components. The viscosity of the gas in the cell is estimated using Wilke's method, with the Herning-Zipperer approximation, as described in Reid, Prausnitz and Poling [189].

3.2 Lagrangian Particle Methods

This section will discuss numerical methods based on the Lagrangian description of fluid motion. These methods may provide advantages in simulating flows in which fluid interfaces are a key component. Flows in which interfaces are important include shock tunnel flows, in which the contact surface between the driver and driven gases plays an important role in many of the processes that occur.

A numerical method based on the Lagrangian description that relied on the use of a mesh would suffer problems caused by mesh distortion. As fluid elements moved around one another, the mesh elements would overlap and become tangled. This mesh distortion is evident in Figure 3.1. It is possible to continually remap the Lagrangian mesh, but this would introduce numerical diffusion and, therefore, would negate the advantage of being able to advect fluid interfaces with little diffusion.

The problem of mesh distortion can be avoided by not using meshes at all. What is required for this is a method of interpolating fluid properties without relying on a mesh joining computational points. One such method is the Smoothed Particle Hydrodynamics (SPH) method [78, 137].

Meshes are used to join computational points for the purpose of interpolating fluid properties. With the aid of a computational mesh, interpolating fluid properties is straight-forward; however, interpolation is more complicated without the aid of a mesh. In SPH, continuum fluid properties at a particular location are interpolated as weighted sums of the properties of surrounding particles in a process known as kernel interpolation.

Some aspects of the SPH technique will be described in this section. The parallel implementation of a code based on the SPH technique is described in Chapter 5. The parallel performance and efficiency of this code are discussed in Appendix A.

3.2.1 Smoothed Particle Hydrodynamics

Smoothed Particle Hydrodynamics (SPH) is a mesh-less, purely Lagrangian numerical method for the transient solution of the Euler equations. The Euler equations describe the motion of an inviscid, Newtonian fluid. The continuum fluid is represented by a collection of fluid pseudo-particles; however, these particles are more precisely thought of as merely interpolation points for fluid properties, rather than actual particles of fluid. The Lagrangian form of the conservation equations of fluid flow become the equations of motion of these interpolation points as they move with the flow. As a result the method is completely deterministic. These interpolation points move with the flow and as they do, they interact with one another and carry

with them all of the computational information about the fluid. Fluid properties are then interpolated between the particles by kernel interpolation, which will be described later.

The SPH technique is described in two major reviews: Benz [19] and Monaghan [156]. Another review on the subject of particle methods, which focuses on SPH, is Monaghan [151],

An actual fluid is made up of molecules with empty space between them. SPH is based on the continuum model of a fluid which assumes that the fluid is a continuous medium over all length scales. The fluid properties are known as a discrete, arbitrary set of points, that is the set of interpolation points. Through kernel approximation [73], the properties of the fluid at the discrete points are smoothed out over the flow domain, producing a smoothed field of fluid variables. The properties of the fluid at a particular point in space can then be calculated using the sum of all of the smoothed contributions from particles near the point.

The smoothing function applied to each of the particles is referred to as the *kernel*. These particles do not represent a singular point in space, but rather the space surrounding them according to the kernel function. Mass is maintained by particles, while momentum and energy are exchanged between them. The kernel function and a characteristic smoothing length parameter, referred to as the *smoothing length* define the domain over which the properties of a particle are smoothed.

Artificial viscosity is used to introduce dissipation into the technique in order to resolve shocks [159]. The use of switches to control the addition of this artificial viscosity have been discussed by Morris and Monaghan [160] and an improved form of the artificial viscosity was described by Watkins et al. [239].

Background of the Technique

Particle methods have been used since the 1960s in plasma physics and N-body problems. These N-body problems were largely self-gravitating astrophysical simulations. The first use of particle based methods for solving fluid dynamics problems was the Particle In Cell (PIC) method Harlow [91]. This method used particles to advect physical quantities and then used a grid to calculate spatial derivatives by finite differences. The PIC method used large amounts of computer memory due to its need to store two sets of data, from the particles and from the mesh. The main limitations of the PIC method were excessive numerical noise and inadequacies in the representation of viscosity and heat conduction [18].

The problems with the PIC method were mainly as a result of the requirement for a mesh as well as the particles. A method by which spatial derivatives could

be calculated over the particles would remove the requirement of a mesh. This was achieved with the SPH technique, which was first described in 1977, concurrently by Lucy (1977) [137] and Gingold and Monaghan (1977) [78]. The application that was described by both publications was the simulation of the fission of a rapidly rotating star. As with previous particle methods, particles carry physical quantities with them, but what was new with this method is that spatial derivatives are calculated analytically from approximate interpolation formulae which refer only to particle properties. A significant advantage of this is that removing the requirement for remapping onto a grid meant that SPH was significantly less diffusive than previous methods, such as the PIC method. The method was applied primarily to astrophysical and cosmological problems throughout the 1980s but, in recent times, other applications of the method have been realised.

Reviews of the applicability and suitability of SPH to particular problems have been provided by Steinmetz and Muller [222], Belytschko, Krongauz, Dolbow and Gerlach [16], and Hernquist [96]. Randles and Libersky [185] described the improvements that have been made to the SPH technique over the period from 1991 to 1996. Belytschko, Krongauz, Organ, Fleming and Krysl [17] provided a similar description of the developments over the same period, extending their discussion to other particle methods.

Modifications to the basic SPH technique are being developed, which make improvements to aspects of the technique. These modifications include the Corrective SPH (CSPH) technique, which was developed by Chen, Breaun and Carney [42], and the Normalised SPH technique described by Randles and Libersky [186]. The Moving Least-Squares Particle Hydrodynamics (MLSPH) technique has been described by Dilts [59, 60] and the Moving Least Squares Reproducing Kernel (MLSRK) technique has been described by Li and Liu [134]. These modifications to SPH demonstrate both the early stage of development that SPH has reached and the significant amount of research that is being conducted in improving the technique.

The Adaptive SPH technique has been described by Shapiro, Martel Villumsen and Owen [205, 170]. This modified technique provides many improved qualities over the traditional SPH technique; the most important of these is the use of a smoothing length that is solution adaptive.

The SPH technique exhibits a natural parallelism. This aspect of the technique will be discussed further in Chapter 5.

Interface Modelling with SPH

Provided that enough particles are used in a simulation, fluid interfaces should be maintained as with a grid-based Lagrangian method; that is, material from each side of the interface will remain on that side of the interface and will not cross over or mix at the interface. As a result of this fluid interfaces should remain sharply defined throughout a simulation.

Figure 3.2 compares the modelling of an fluid interface using a fixed Eulerian mesh and using moving computation points which are not joined using a mesh, such as with SPH. In the Eulerian mesh representation it can be seen how the fluid interface crosses through the centres of the cells. In any of these cells, the fluid properties from either side of the interface are mixed in these cells. This process continues through time, as the interface crosses more cells, increasing the mixing.

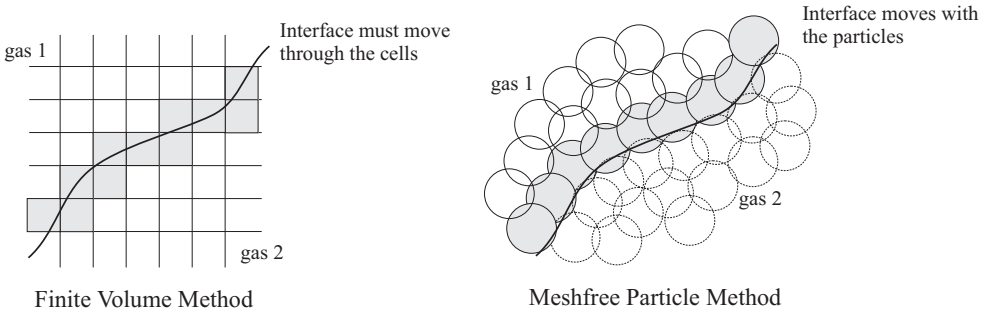

Figure 3.2: Interface modelling with finite volume and mesh-free particle methods

In modelling a fluid interface with SPH the sharp interface is smeared out by the interpolating function. This reduces the detail around the interface; however, since these smoothed computational elements move with the interface, an improved representation may still be obtained. A further problem, which will be discusses later, is the that of the unphysical penetration of particles through the interface [154].

Applications of SPH

Applications of SPH to compressible fluid mechanics problems comprise only a small sub-set of the application studies that have been conducted with SPH. SPH has quite a wide range of potential applications. The properties of different materials can be specified through the use of a suitable equation of state [157]. As a result of its origins, to date the majority of applications that SPH has been applied to have been in astrophysical domains [218, 187, 18, 24].

Another large sub-set of applications come from incompressible, free-surface fluid mechanics problems. SPH can be extended to the simulation of incompressible

fluids, through the use of a suitably stiff equation of state, and being Lagrangian in nature, SPH is suitable for free surface applications [155]. SPH has been applied to the injection and casting of metals by Cleary and Ha [47] and Ha, Ahuja and Cleary [88], to surface coating techniques by Reichl, Morris, Hourigan, Thompson and Stoneman [188]. Multi-phase applications have also been studied, including under-water explosions by Swegle and Attaway [231].

The application of SPH to compressible flows has been described by Monaghan [150]. Monaghan [152] describes the application of SPH to hypersonic flow; however, no reference is made to the simulation of the real gas effects associated with real hypersonic flows. Monaghan [158] discusses the similarities between SPH and solutions of compressible flows using Riemann solvers.

The application of SPH to shock tunnel flows has been investigated by Robinson [193]. This study focused on the simulation of the rupture mechanics of the primary diaphragm. The SPH technique was thought to provide the potential for a more complete description of diaphragm rupture, including the ability of modelling the fluid structure interaction with the fragments of the ruptured diaphragm. One and two dimensional models of the diaphragm rupture process were developed. The simulations demonstrated a qualitative view of the rupture process, although the use of boundary particles meant that the flow in the vicinity of walls was unphysical.

Morris [160] studied the interaction of planar shock waves with cylindrical bubbles. This is the same type of simulation that is described in Chapter 6. The simulations that were performed incorporated a new switch for the artificial viscosity term in the momentum and energy equations. The results were compared with simulations performed using a CFD code with a finite-volume formulation and no comparison with experimental results was provided.

Henneken and Icke [95] used SPH to simulating the forward facing step test case as descibed by Woodward and Colella [249]. These simulations are discussed later in this section.

Details of the SPH Technique

An SPH simulation is started by placing the particles in the domain such that the density of the fluid is represented by the relative density of the particle positions. To interpolate the information that we have at particle positions the information stored at a point in space is numerically distributed over the surrounding area by a kernel function, W(r). The degree of smoothing is defined by a parameter known as the smoothing length and can be either constant for all particles or vary depending on the local density.

The state vector describing particle properties is $\{\mathbf{r}, \mathbf{v}, e\}$, where $\mathbf{r}$ is the position vector, $\mathbf{v}$ is the velocity vector, and e is the specific internal energy. The derivatives of the state vector are given by $\{\mathbf{v}, \frac{d\mathbf{v}}{dt}, \frac{de}{dt}\}$.

Equation 3.25 shows this summation used for some fluid property, A, being interpolated at the sample point $\mathbf{r}$ using the kernel smoothing function $W(\mathbf{r}-\mathbf{r}_b, h)$. The contributions to the properties from all particles b in the domain are summed to produce the result.

$$A_s(\mathbf{r}) = \sum_b A_b \frac{m_b}{\rho_b} W(\mathbf{r} - \mathbf{r}_b, h) \tag{3.25}$$

Graphically this process is represented for a one dimensional case in Figure 3.3, where the masses of a series of particles in the r axis have been smoothed and the property ρ has been interpolated to a continuous function.

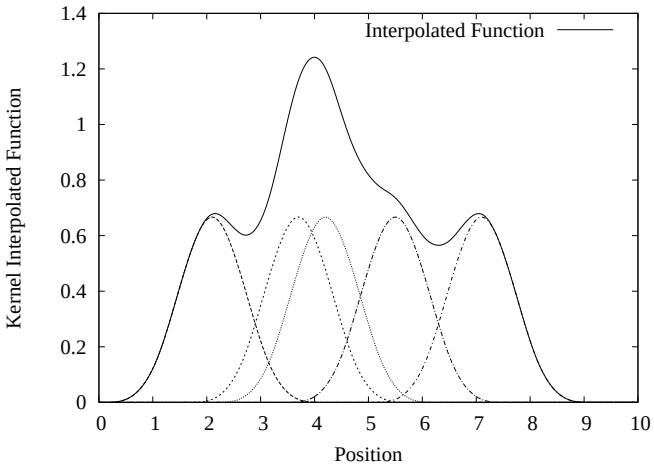

Figure 3.3: Gaussian kernel interpolation of density in one dimension from five particle positions.

where m_b is the mass of fluid assigned to the particle b and ρ_b is the density of the particle b. An important aspect of kernel interpolation is that, with the right choice of kernel, the gradient of fluid properties can be interpolated directly using the gradient of the kernel. With the Gaussian kernel this gradient is known analytically

and, therefore, can be used to interpolate these gradients in the same way that fluid properties are calculated. In the case of the Euler equations the gradient of pressure is required:

$$\frac{D}{Dt} = \frac{-1}{\rho}\frac{\partial p}{\partial \mathbf{x}} \tag{3.26}$$

where $\mathbf{v}$ is the velocity vector of the fluid and $\mathbf{x}$ is the coordinate vector, ρ is the density of the fluid and p is the pressure of the fluid. This equation is solved using the gradient of the kernel interpolant, resulting in Equation 3.27. This equation determines the acceleration experienced by the particle, a, using the pressure and density properties of the surrounding particles, denoted by b. The symmetric nature of this equation ensures that particle interactions are equal and opposite [153].

$$\frac{d\mathbf{v}_a}{dt} = -\sum_b m_b \left(\frac{p_a}{\rho_a^2} + \frac{p_b}{\rho_b^2}\right) \nabla_a W(\mathbf{r}_a - \mathbf{r}_b, h) \tag{3.27}$$

The rate of change of specific internal energy is calculated using:

$$\frac{de_a}{dt} = \frac{1}{2}\sum_b m_b \left(\frac{p_a}{\rho_a^2} + \frac{p_b}{\rho_b^2}\right) \mathbf{v}_a \cdot \nabla_a W(\mathbf{r}_a - \mathbf{r}_b, h) \tag{3.28}$$

Dissipation in the form of two types of artificial viscous pressures are usually added to Equations 3.27 and 3.28: a Von Neumann-Richtmyer artificial viscosity and a bulk viscosity. Pseudo code for the SPH technique is shown in Figure 3.4.

```
      1.  assign particles to flow domain according to initial density
      2.  assign initial properties to particles
   -> 3.  calculate densities at particle positions
   |  4.  calculate particle pressures and local sound speeds
   |  5.  calculate particle accelerations
   |  6.  calculate rate of change of internal energy
   |  7.  integrate particle properties forward through time step
   -- 8.  check if solution time has been reached
```

Figure 3.4: Pseudo code for the SPH method

Although the SPH tecnique can be computationally expensive, there are techniques for increasing its efficiency and, therefore, the size of simulations that can be run in practice. These include the use of efficiently structured code and compiler optimisations, particle sorting methods, such as cells and hierarchical trees [97], and overall parallelisation of the algorithm.

There are optimisations specific to the SPH method that significantly impact performance. Polynomials are often used for the interpolating kernels instead of Gaussian functions as they are less costly to evaluate. This has a significant effect on performance as the kernel function must be calculated for every particle's contribution to the summations used. Forces between particles are equal and opposite

and so the number of interactions actually calculated in the solution can be reduced by a factor of two. This optimisation is easily realised with a sequential solution, but care must be taken when implementing this in parallel.

Particles that are a long distance from the particle being updated will provide a negligible contribution to the summation. Special interpolating kernels recognise this and provide compact support [74]: outside of a certain radius the contribution is zero. This means that only nearby particles, usually within a radius of two times the smoothing length, need to be considered. Particles can then be sorted in order to quickly find their neighbours. For simulations in which the smoothing length does not vary, particles can be assigned to fixed-size cells. For a specified location, the nearest cells and, therefore, nearest particles are easily found. Figure 3.5 shows a particle, marked by a cross, and the circular region in which other particles will have a significant contribution. The cell (m, n) in which the particle lies is shaded and the surrounding cells that are included in the calculation by the sorting technique are lightly shaded. Hierarchical trees [97] can be used for simulations in which the smoothing length varies between particles.

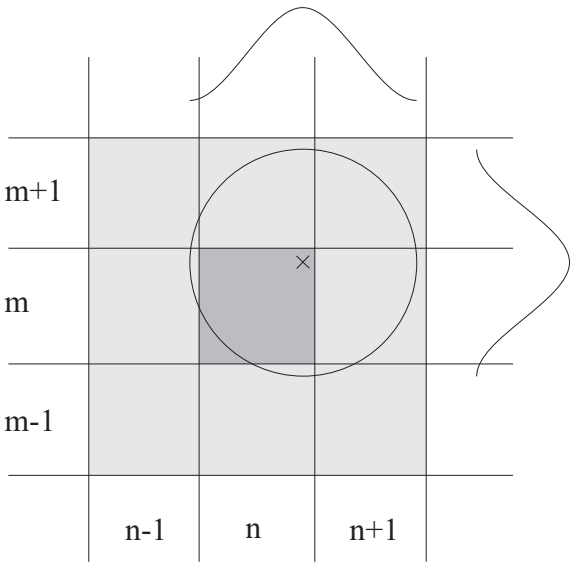

Figure 3.5: Cell sorting used for finding neighbouring particles to those in cell (m, n) with constant smoothing length.

The SPH Code

A CFD code based on the SPH technique was developed and applied to several test cases. Initial conditions can be specified in the code using either regular cartesian initial arrangments of particles or an implementation of a quasi-random sequence generator, which would assign particles to an arbitrary geometry in a quasi-random manner [23]. The code can interpolate the density of the particle field using either the summation density algorithm or the continuity equation. A constant smoothing length was used throughout a simulation. The cell based selection of neighbouring particles was included, which resulted in significant improvements in the speed at which a particular flow field could be solved. Generalised inflow and outflow boundary conditions were developed. These boundary conditions were applied successfully to the test case in the following section. Reflected boundary conditions, representing solid slip walls were also developed. These boundary conditions were extended to piecewise-linear boundary conditions and were applied to some cases. The code was implemented in parallel and its performance is described in Appendix A.

The Riemann problem has been studied using the SPH method by many authors [136, 193]. This test case was modelled with the SPH code developed in this thesis. Since this case has been described extensively, it will not be described in detail in this section apart from some general observations of the solutions obtained: The shock and contact discontinuities are smeared over approximately three smoothing lengths. Significant post shock oscillations, resulting from the use of artificial viscosity, are evident. A large oscillation in pressure is evident at the position of the contact surface and a small dip is evident near the end of the expansion.

Test Case: Forward Facing Step

The forward facing step is a test case used to investigate the ability of CFD codes to reproduce a standard flow field from another CFD code. The flow situation being modelled is two-dimensional, inviscid, supersonic flow (at a Mach number of 3) along a duct over a forward facing step. This test case is described by Woodward and Colella [249] and is used as a test case that includes supersonic and subsonic flow regions, strong shocks and expansions. The test case can be used for the comparison of CFD codes only and is not a physically realisable case, since the flow is impulsively started and is inviscid.

This case had been attempted perviously using SPH by Henneken and Icke [95]. Their simulations were limited by the inflow boundary conditions. The simulations used a constant number of particles, with particles being introduced to the upstream boundary as they passed through the downstream boundary. The result of this is

that the mass flux at the upstream boundary is dependent on the downstream mass flux, which is unphysical. Being a transient flow problem, the flow through the duct would not be expected to be constant. As the density is increased behind the shock and its reflection, the rate at which particles leave the flow field varies strongly over time. In addition, the particles are added to the upstream boundary at random heights, which introduced significant noise into the flow field.

In these simulations, the number of particles is allowed to vary through the simulation. This allows a steady inflow condition to be maintained. In addition, a significantly larger number of particles is used in the simulations in this section: greater than 24,000 rather than 600 particles. Being an inviscid test case, the solid walls of both the duct and the step are modelled using reflected boundaries.

Figures 3.6 and 3.7 show a sequence of the simulation of the forward facing step. Vectors at the position of each particle are shown in the frames. The initial conditions are specified as regular cartesian arrangements of particles. The upstream inflow boundary condition introduces particles into the flow field in regularly arranged rows. The first frame is shown soon after the reflection of the shock from the step. The regular arrangement of the particles is broken up as the particles pass through the shock. The particles near the top of the step have a definitely less ordered arrangement than that particles further out in the flow field. The increased numbers of particles in the high density regions behind the shock and its reflection can be seen throughout the sequence. Post shock oscillations are evident behind the shock in all of the frames.

Figure 3.8 shows a comparison of the results from Woodward and Colella [249] with the results from the SPH code simulation. Contours of the density field are shown in the frames. The SPH simulation achieves reasonable agreement with the results from Woodward and Colella; however, some significant differences are evident. The SPH results appear slightly behind in time from the results of Woodward and Colella and the effects of the post shock oscillations are apparent. In the last frame an additional bend in reflection of the shock from the top wall of the duct (in the segment between the Mach stem and the reflection from the top of the step) is evident. The origin of this bend is not known and the SPH simulations of Henneken and Icke [95], which did not resolve the shear layer behind the Mach stem, did not exhibit this problem. In addition, despite the relatively large number of particles used in the simulation the resolution of the SPH simulation is inferior to the simulation of Woodward and Colella, which was performed almost twenty years ago. An advantage of the SPH simulation is that it does not require the entropy fix that is applied to finite volume simulations of this test case [249], in order to remove the singularity at the top corner of the step.

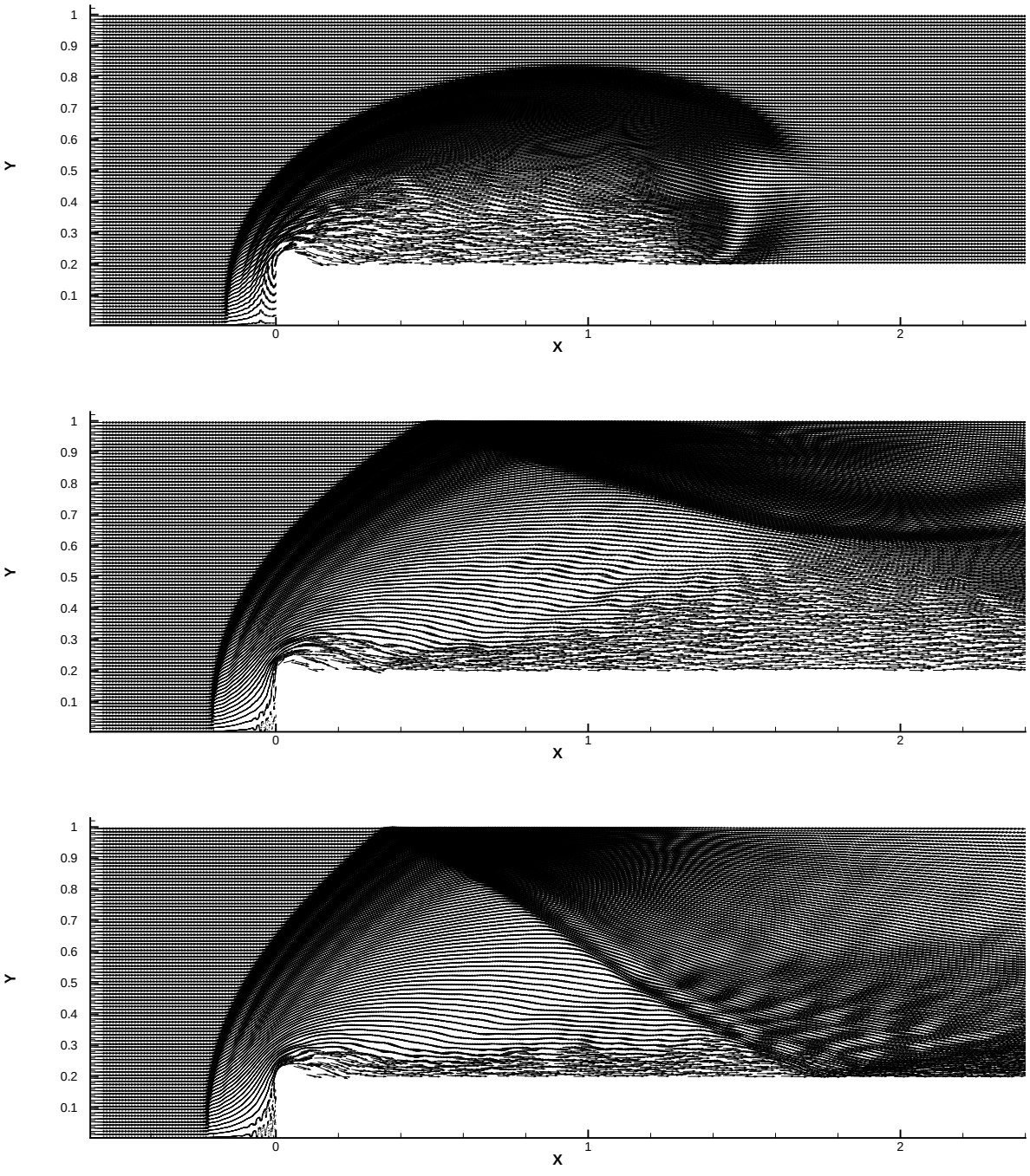

Figure 3.6: Part one of a sequence of the simulation of the forward facing step test case using SPH. Velocity vectors at the position of each particle are shown.

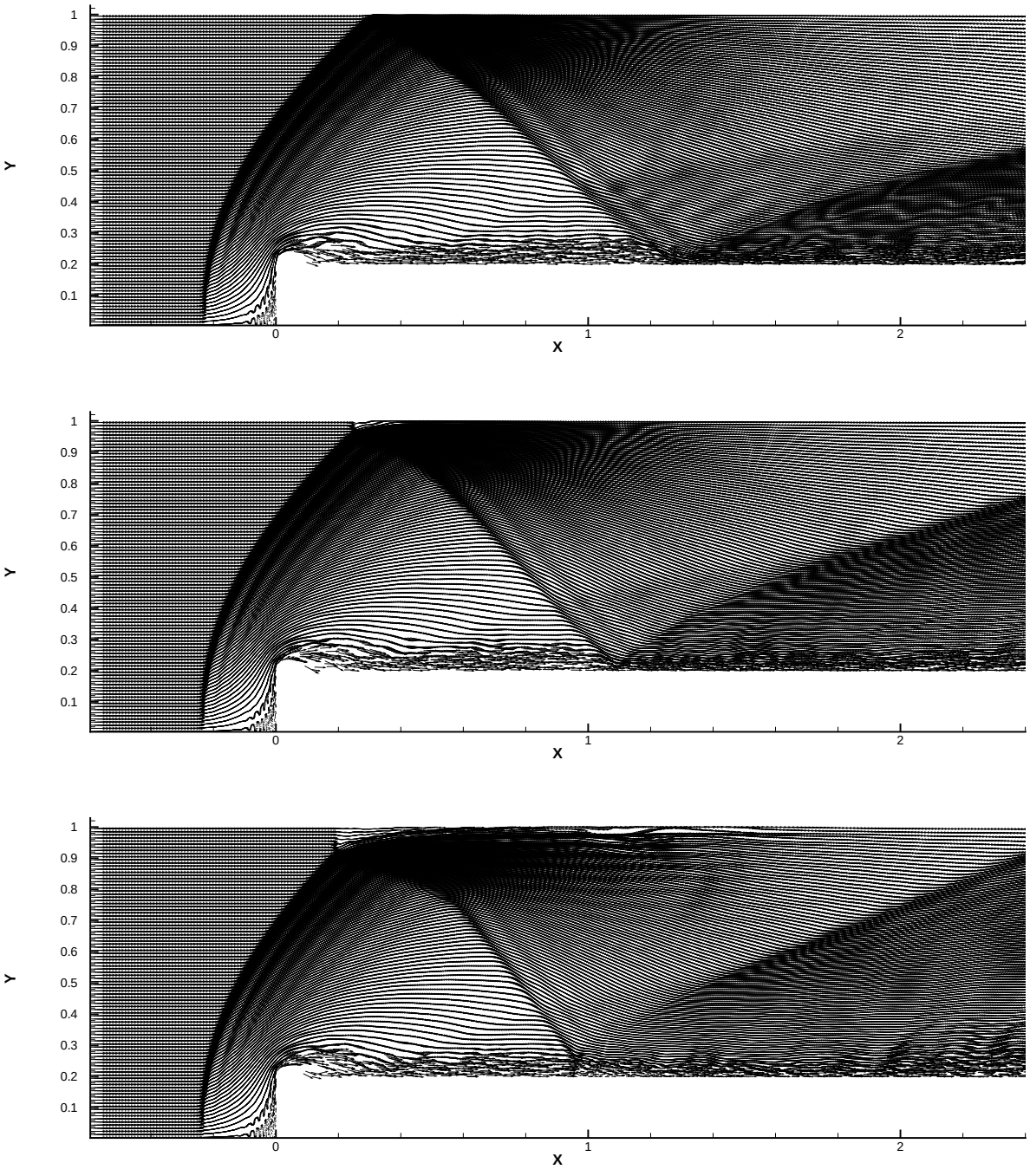

Figure 3.7: Part two of a sequence of the simulation of the forward facing step test case using SPH. Velocity vectors at the position of each particle are shown.

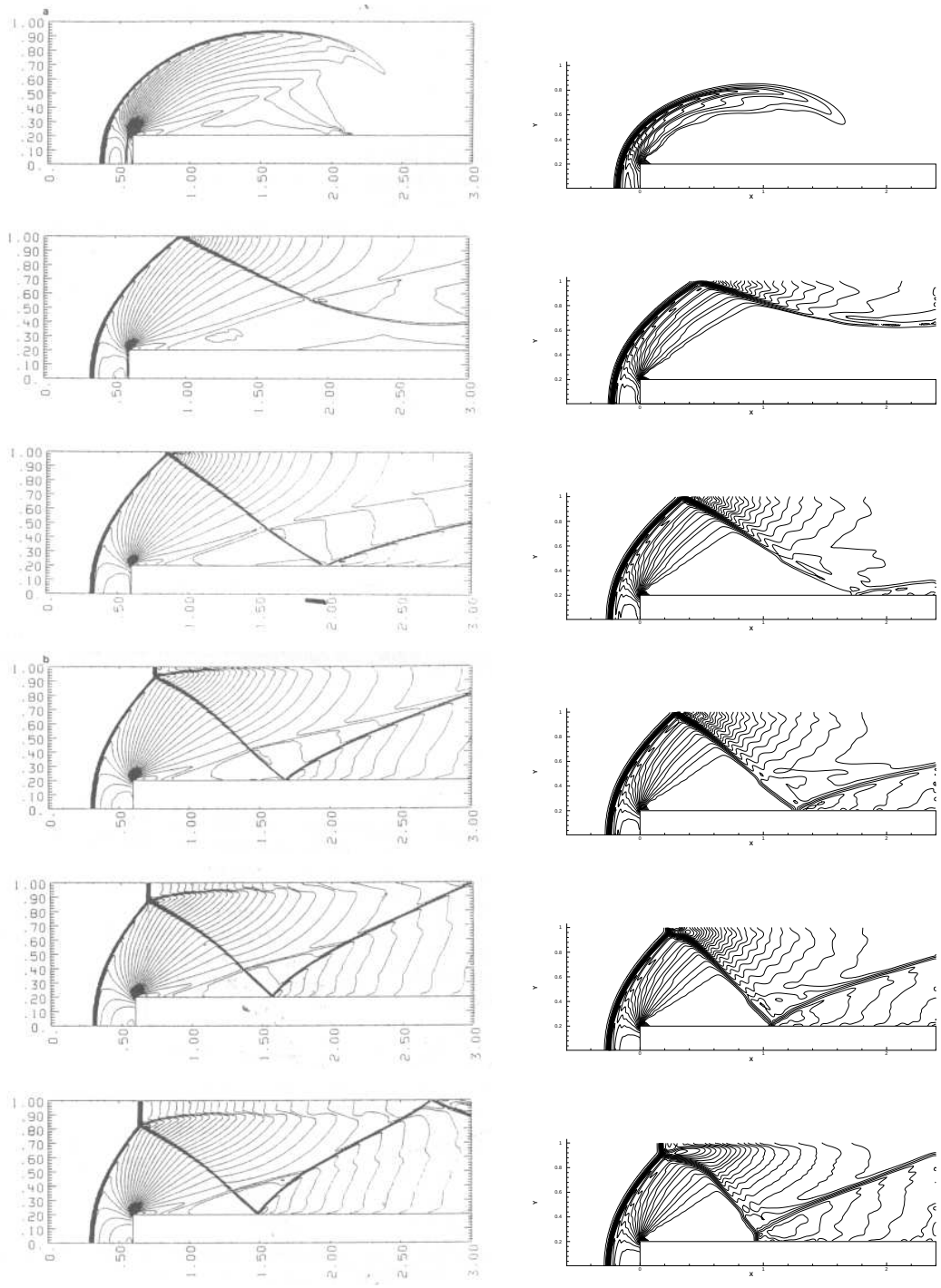

Figure 3.8: SPH Modelling of Forward Facing Step. Comparison of the results of Woodward and Colella (1984) with the results from the SPH code.

Limitations of SPH in Simulating Shock Tunnels

After the test cases were developed, a decision was made not to continue development of the SPH code, but to continue the shock tunnel simulation exclusively with MB_CNS. SPH is not a commonly used numerical technique and many of the problems associated with the use of SPH are a product of the relative lack of development of the technique and of the code. As an example of this, some of the problems which were observed were due to the implementation of a constant smoothing length. A method that incorporated a smoothing length that was varied as a function of the local density [205] would improve the applicability of the technique. This thesis is intended as an application study and a significant amount of time spent on the further development of numerical techniques would detract from the application side of the study. The code that was developed provided a useful test bed for parallel computing methods and so will be discussed further in this thesis in that context.

The development of test cases allow the code to be assessed in the context of what would be required in order to use the code in the simulation of complete shock tunnels. Some of the issues that were identified that would prevent the application of the SPH code were:

Use of Artificial Viscosity The requirement for an artificial viscosity means that SPH resolves shocks poorly in comparison to contemporary methods. Shocks are typically resolved over three smoothing lengths. In a constant smoothing length simulation, such as those in this thesis, the smoothing length, is based on the largest particle spacing in the simulation. This leads to an inefficient use of particles in most regions of the flow. This is in addition to the relatively low resolution of the SPH technique in general. The use of artificial viscosity also results in post shock oscillations.

Solid Boundaries The treatment of solid boundary conditions has always been a significant limitation of the SPH technique. Flat, slip boundaries can be implemented in a straight-forward manner with reflected boundary conditions. Curved, slip boundaries can be represented in a piece-wise linear manner with the same reflected boundary conditions. The complex geometry of the nozzle would not be represented satisfactorily with a piecewise linear profile. Reflected boundary conditions also do not provide for non-slip boundaries and the use of boundary particles to represent non-slip boundaries [48] is considered to be non-physical. Attempts to extend the simulations to the inclusion of curved boundaries with Lennard-Jones potentials [103] were unsuccessful. The potential at the wall would repulse any approaching particles with a constant exponential force. This repulsive force, although keeping the particles inside

the wall, was not physical. Discussion of boundary conditions are provided in Dilts and Haque [60]. Reflected boundary conditions were used for the forward facing step test case.

Boundary Layers Numerical techniques applied to the simulation of shock tunnels must be able to reproduce the shear stress and heat fluxes at the shock tube walls. This means that the techniques must be able to accurately predict the gradients of velocity and temperature, amongst other variables, at the wall. It is believed that many of the application studies, through their use of boundary particles, do not deal with solid wall boundaries in a way that would allow this.

Turbulence Models The Baldwin-Lomax eddy viscosity model [11] was used in the simulations using MB_CNS in this thesis. There are no equivalent implementations of this, or any other turbulence models that are usable in the SPH code. Sub-particle-scale scale turbulence models are feasible and have been proposed by Gotoh, Shibahara and Sakai [82]. This model has not been compared with any experimental results or tested in any extensive way. In addition, the model is not directly applicable to the SPH code that has been developed in this thesis. The simulation of turbulent boundary layers was also discussed in Cleary and Monaghan [48]; however, the treatment of turbulence in the boundary layer was not rigorous.

Interface Modelling There are significant complications with the way that interfaces are modelled in SPH. These problems include the non-physical penetraction of particles through the interface. Arranging particles in regular patterns, such as lattices, can result in lines of particles having artificial stiffness. This artificial stiffness can promote the unphysical penetration of particles through fluid interfaces. This penetration is also unpredictable. The issue of particle penetration has been discussed by Monaghan [154] and Latanzio, Monaghan, Pongracic and Schwarz [125]. These publications propose methods of preventing penetration; however, these methods are dissipative and this phenomenon in an undesirable feature of the SPH technique in general. This presents an important problem since the purpose of investigating SPH was for advantages that may have been provided in the low diffusion modelling of fluid interfaces. On the other hand, it is shown in Chapter 7 that the representation of the contact surface in MB_CNS is in fact sharper than the real contact surface. As a result of this, some additional, physical form of diffusion, would be required in the simulations, rather than a method of maintaining an even sharper interface.

Multiple Component Gases The simulation of mixtures of gases in SPH is still

relatively unstudied. The effect on kernel interpolation of simulations involving mixtures of particles representing different gases is not clearly understood. In addition, physical species diffusion models have been used extensively in Eulerian based methods. The particle based nature of SPH causes difficulty in formulating physical diffusion models at fluid interface.

Specification of Initial Conditions Specifying initial conditions on regular lattice like arrangements makes SPH susceptible to particle penetration. In addition, the assignment of particles to lattice arrangements near solid boundaries leaves voids where the boundary and the lattice do not align. The quasi-random sequence generator, described by Bratley and Fox [23], was implemented in the code, thus allowing a general method of filling the flow field with particles; however, the use of this type of initial condition led to extremely noisy flow fields.

Tension Instability Tension instability results from the use of peaked kernels. The gradient of the kernel reaches a maximum value at a finite distance from the particle position. This means that particles will experience an increasing pressure as they approach a particle to the location of this maximum kernel gradient; however, the pressure begins decrease as they continue to approach past this point. This can result in unphysical particle behaviour, such as particles stacking on one another. This problem can be solved through the use of conservative smoothing or moving least squares interpolants [186].

3.3 Previous Numerical Simulation of Shock Tunnels

Numerical simulations of shock tunnels are used to determine the operating conditions required to produce required test flow conditions and to investigate problems with the operation of the facilities. Previous numerical simulation studies can be generally divided into two categories depending on the way in which they approach the modelling:

1. quasi-one dimensional simulations of the entire facility
2. axisymmetric simulations of part of the facility (usually the end of the shock tube and the nozzle, with the conditions upstream of this section of tube assumed from theoretical profiles)

The quasi-one dimensional simulations are used to model the flow development through complete shock tunnels, assuming that the flow properties vary only along the length of the tube. Mass loss models must be used to account for the effect that the boundary layers have on the core flow [64]. Quasi-one dimensional simulations provide a computationally efficient method of studying the transient nature of shock tunnel flows. Examples of simulations of shock tunnels using quasi-one dimensional simulation techniques are provided in Zeitoun, Brun and Valetta [254], Jacobs [113] and Doolan and Jacobs [64].

Due to the computational requirements, few fully three dimensional modelling of shock tunnel flows have been conducted. Given the inherently axisymmetric nature of shock tunnels flows, studies have chosen to focus their computational effort on fine grid resolution in axisymmetric simulations. The literature was not found to contain any references to improvements obtained through the use of three dimensional simulations over axisymmetric simulations.

3.3.1 Axisymmetric Models

Burtschell, Brun and Zeitoun [29] performed numerical simulations of the, then proposed, TCM-2 free piston shock tunnel. The aim was the prediction of operating conditions for the proposed facility and the investigation of the unsteady phenomena, which were thought to restrict the performance of the proposed facility in high enthalpy operation. Quasi-one dimensional simulations of the piston compression process were performed. They attempted to simulate the whole shock tunnel with an axisymmetric simulation, but were restricted by the limited computing power

available at the time. Their mesh, although covering a representative geometry of the whole facility, only allowed for ten cells across the diameter of the tube; certainly not enough to resolve flow features such as boundary layers, or regions of shock interaction adequately. An explicit, unsteady method for solving the Euler equations with a Van-Leer decomposition was used in the solution. Conclusions were reached regarding the unsteady processes occurring the facility. This paper represents the first attempt at axisymmetric simulation of a shock tunnel facility.

Badcock [8] simulated the propagation of a shock along a shock tube to investigate the effects by which these flows deviate from the inviscid solution. The unsteady evolution of gas properties as the shock propagated along the tube were examined. A mixture of Roe's method and central difference schemes were used. The flow was modelled in two dimensions, with the scheme being implicit in the radial spatial dimension. The thermodynamics were assumed to be described by a perfect gas law. Comparisons were made with experimental data and with analytical solutions obtained via boundary layer equations. The effect of varying parameters, such as a uniform viscosity term, the heat conductivities and the boundary layer thickness were examined.

Lee and Lewis [130, 129] presented a numerical study of unsteady, viscous, hypersonic flows in two dimensional shock tunnel nozzles. The two dimensional Navier-Stokes equations were solved with an upwind TVD scheme. The simulations investigated the flow development time around aerodynamic models in the test section. The simulations modelled only the flow through the nozzle, and did not address the nature of the flow in the shock tube, assuming it to provide a steady reservoir for the nozzle. Lee and Nishida [131] also simulated the startup flows in hypersonic nozzles to investigate the flow establishment times.

Wilson, Sharma and Gillespie [247] performed time dependent, quasi-one dimensional and axisymmetric simulations the flow through the NASA Ames electric-arc driven shock tube facility. The simulations focused on the interaction of the reflected shock with the boundary layer, as evidence of this interaction was clearly evident in the experimental data. The operating conditions considered were over-tailored. This experimental data was used in comparison with the simulations. Some limitations were evident in the ability of the simulations to reproduce the experimental traces through the region of the interaction of the reflected shock with the boundary layer.

Figure 3.9 shows the interaction of the bifurcated reflected shock with the contact surface as simulated by Wilson, Sharma and Gillespie [247]. The top frame shows the flow before the interaction and the bottom frame shows the flow shortly after the interaction. Evident in the frame before the interaction is the jetting, which

results from the movement of gas through the oblique shocks at the foot of the reflected shock, and the approaching planar contact surface. In the frame after the interaction, the driver gas moving through the shock foot can be seen to be ahead of the driver gas moving along the centreline.

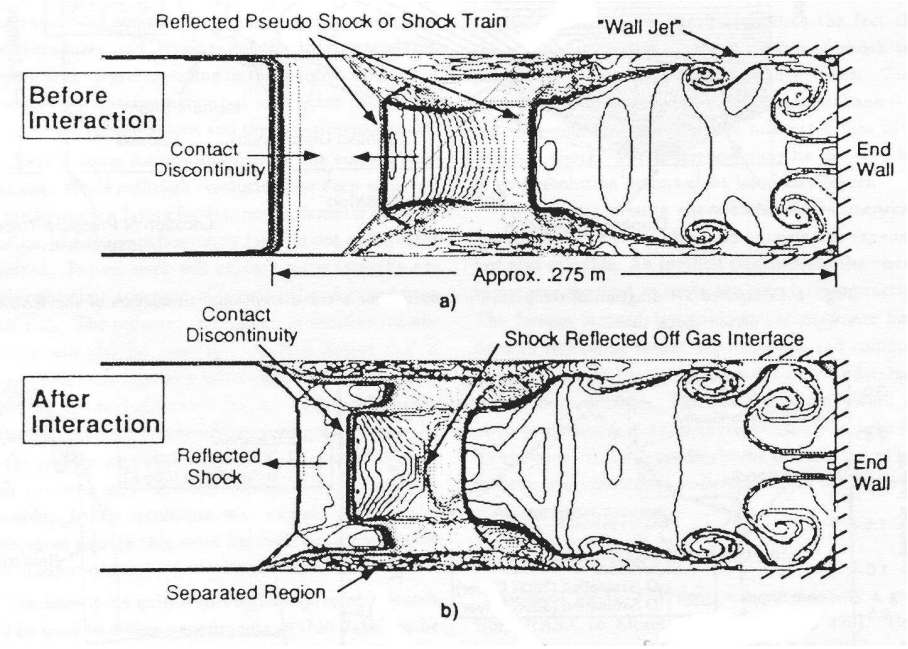

Figure 3.9: The interaction of the reflected shock with the boundary layer from the simulations performed by Wilson, Sharma and Gillespie [247]. The process before the interaction of the reflected shock structure with the contact surface is shown on the top and shortly following the interaction on the bottom.

Figure 3.10 shows the flow evolved later in time, including the late time evolution of the contact surface as it emerges from the shock structure. The driver gas can be seen to be moving along the wall of the shock tube, far ahead of the driver gas near the shock tube centreline. The pseudo shock train, following the reflected shock, is also evident. Large vortices are shown to form in the shear layer between the gas that has moved through the oblique shocks and the gas that has moved through the normal shock.

Tokarcik-Polsky and Cambier [234] investigated the process of shock reflection from the end wall of a shock tunnel and the resulting startup flow through the nozzle. The simulations were inviscid and focused on the formation of a vortex system which was believed to form along the centreline of the shock tube. The nozzle geometries used had a flat downstream face, and varying degrees of contour through the nozzle mouth. A non-physical jetting of gas along the nozzle centreline was observed for certain ratios of mesh resolution in the axial and radial directions. It was determined that including the effect of viscosity on the reflected shock tube flow would reduce

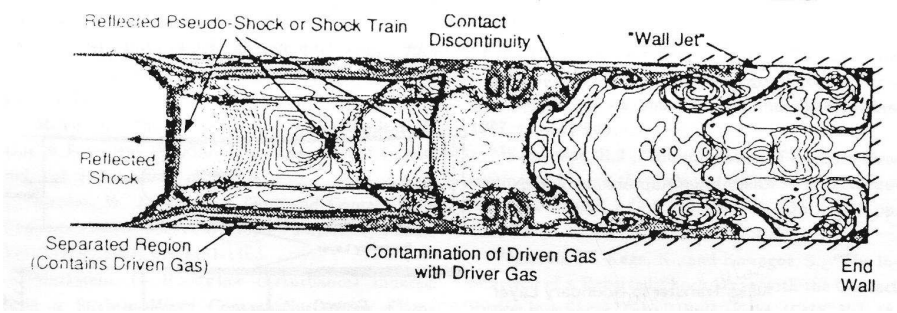

Figure 3.10: The flow field from the simulations performed by Wilson, Sharma and Gillespie [247] shown later in time from the frames in Figure 3.9.

this effect. The simulations showed that no vortex structure existed in the shock tube as was thought. The simulations were compared to experimental visualisation and showed a good correlation. The experimental results also failed to show a vortex structure in the shock tube.

Sharma and Wilson [206, 207] conducted axisymmetric simulations of the development of the flow along the length of a shock tube. Their study was aimed at extending the analytical studies of Mirels [149], Roshko [195] and Hooker [104] with simulations of a shock tube with viscous boundary layers. The axisymmetric flow through a shock tube was modelled using the thin layer Navier-Stokes equations. The gas model used accounted for finite-rate chemical processes and included a separate equation for vibrational energy. The flow conditions were chosen so that Mirels' correlations [149] were valid. Sections of the shock tube were modelled, with the contact surface assumed to be initially planar. The simulations were performed in the shock reference, with the section of tube between and around the shock and the contact surface modelled. The flow entered the domain from the upstream side of the shock. The boundary layers were assumed to remain laminar. The simulations modelled the temporal evolution of the boundary layer, mass leakage through the boundary layer, the acceleration of the contact surface and the deceleration of the shock. Some boundary layer parameters calculated in the simulations matched Mirels' correlations; however, some boundary layer parameters did not. The simulations did not address the reflection of the shock and the resulting interaction processes and, therefore, the test times from this study provide ideal limits for the test times of real shock tubes. The trends observed in the test times produced by the study were closely related to the trends based on Mirel's correlations, demonstrating the strong dependence of this study on this assumption.

Wilson [246] simulated the flow through the acceleration tube section of the HYPULSE expansion tube. These simulations extended the simulations of Sharma and Wilson [206, 207] by accounting for the effect of turbulence in the boundary layers

using the model of Cebeci and Smith [40]. Even though this study was of an expansion tube flow, it still has relevance to shock tunnel simulation. Two operating conditions were modelled, one producing flow with laminar boundary layers and the other with turbulent boundary layers. A grid refinement study confirmed that the solution converged to Mirels correlation. Experimental pressure traces were compared with traces from the simulations for both the laminar and turbulent operating conditions. In the traces, the overall characteristics of the pressure traces compared well. The simulations were limited to lower Reynolds numbers by the computational requirements of grid resolution in the boundary layer. The authors suggested the implementation of more advanced models, using larger grids and the modelling of more processes occurring in the operation of the facility, such as diaphragm rupture mechanics.

Weber et al. [240] studied the shock bifurcation resulting from the interaction of the reflected shock with the boundary layer and the effect of this interaction on the surrounding flow. Flow conditions resulting from shock Mach numbers of 2.6, 5.0 and 10.0 were simulated. Only the end region of the shock tube was included in the simulations in order to achieve sufficient resolution in this region. The shock tube was modelled as two dimensional. The boundary layer profile used in the reflected shock interaction simulations was obtained from a separate simulation of the propagation of the shock along a flat plate. These simulations provided a demonstration of the complex, transient flow field that results from the reflected shock interaction with the boundary layer. The effect of heat transfer, Reynolds number and incident shock strength were studied. Figure 3.11 shows contours of density and velocity vectors in the region of the interaction (on the left), and the profile of pressure through the interaction, for two different resolution grids (on the right). The stepped profile of pressure through the shock structure obtained in the simulations is evident in the image on the right of the figure.

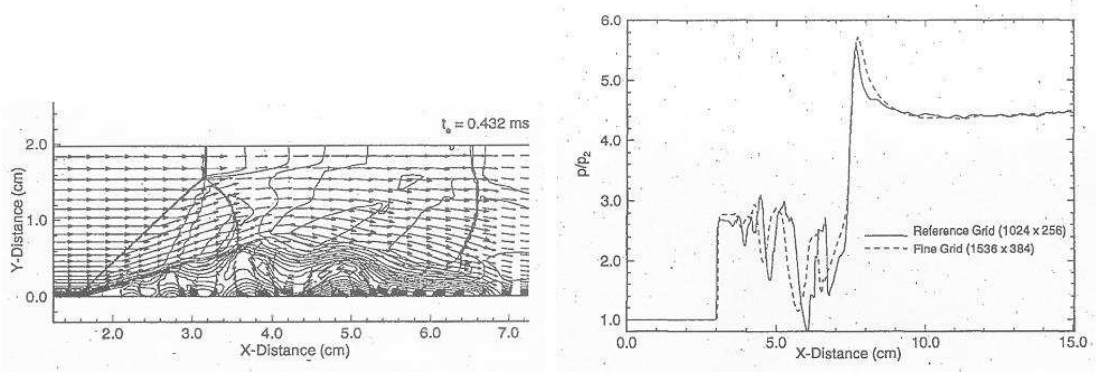

Figure 3.11: The interaction of the reflected shock with the boundary layer as simulated by Weber, Oran, Boris and Anderson, Jr. [240].

Chue [43] performed axisymmetric simulations of the end section of a shock tube in order to investigate the interaction of the reflected shock with a turbulent boundary layer. The Baldwin-Lomax eddy viscosity model [11] was used to account for the effect of turbulence in the boundary layers. This interaction was shown to promote the transport of boundary layer material towards the end of the shock tube. Tailored and off-tailored operating conditions were investigated. The computational domain covered only the region of the shock tube, with the turbulent boundary layer correlations of Mirels being used for the upstream boundary conditions. In addition to investigating the jetting of gas through the bifurcated shock foot, Chue also noted the generation of vorticity in the interaction of the reflected shock with the contact surface.

Hannemann et al. [90] conducted an earlier study of the shock tunnel operation in the same facility as this study. Simulations were compared to experimental results. This study modelled the transient flow through the whole facility using a quasi-one dimensional flow code, and the multi-dimensional flow through the nozzle.

The simulations of Petrie-Repar and Jacobs [176, 177] were discussed in Section 2.1.4 in the context of diaphragm rupture process.

Chue and Eitelberg [45] performed simulations of the HEG shock tunnel. These simulations were a progression from the simulations of Chue [43] and they also focused on the flow features occurring in the shock tube subsequent to the reflection of the shock. In the simulations of the shock tube flow, the computational domain covers the last 710 mm of the 17 m length of the shock tube. The initial conditions for the simulations were at the instant before the reflection of the shock. The initial flow conditions behind the incident shock were calculated from the boundary layer theory of Mirels [148]. All of the analyses conducted used a tailored operating condition.

The authors examined the interaction of the reflected shock with the boundary layer and contact surface in order to investigate their contribution to driver gas contamination and the generation of flow disturbances. The contact surface was assumed to initially be a planar, discontinuous interface between the driver and driven gases. The influence on the contact surface from the diaphragm rupture mechanics, mixing and instability was not modelled. These simulations assumed that the nozzle entrance acted as a mass sink, at which the flow was axial and choked.

Following the interaction with the contact surface, significant vorticity was observed in the contact surface, which resulted in the driver gas being broken up in a complex swirling motion and travelling downstream. This was identified as a form

of Richtmyer-Meshkov instability. The vortical structure generated at the contact surface was shown to have a counter-clockwise motion, which retarded further leakage of driver gas towards the end-plate in the short term. The bifurcated shock structure was shown to weaken as the reflected shock transmits through the contact surface and wall jetting was not apparent as the shock propagated into the driver gas region. The generation of vorticity, and the resulting reduction in the jetting through the shock foot indicated that the evolution of the flow following the contact surface interaction may be driven by the resulting mixing, rather than the shock foot jetting.

Simulated pressure traces were compared with experimental traces. The simulations were shown to achieve agreement with the experimental traces. The average pressure behind the reflected shock structure was lower than the ideally predicted pressure behind the reflected shock. Chue and Eitelberg proposed a theoretical model for the reservoir pressure behind the reflected shock. Pressures measured experimentally in HEG agreed with the predictions of this theory.

Figure 3.12 shows the flow field evolution resulting from the interaction between the reflected shock and the boundary layer and contact surface in the simulations performed by Chue and Eitelberg [45]. Temperature contours are shown on the left side of the figure and driver gas mass fraction contours are shown on the right side of the figure. The generation of significant amounts of vorticity can be seen in the frames at the later times. This includes, what appears to be, a vortex at the head of the driver gas. The time evolution does not run to times that would indicate any movement of this vortex.

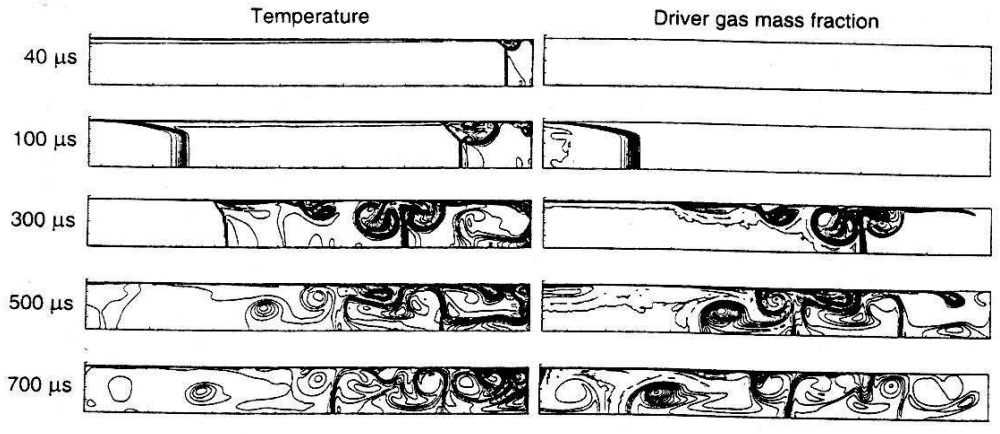

Figure 3.12: The flow field following the interaction of the reflected shock with the boundary layer and the contact surface, in the simulations performed by Chue and Eitelberg [45].

A particularly important geometric feature included in these simulations was

the 'particle stopper' that is used in HEG. The effect of this device on the flow field was examined. In separate simulations, the flow through the nozzle, including the start-up transients, was modelled assuming a steady inflow condition.

Kaneko, Men'shov and Nakamura [118] modelled the flow field of a nozzle starting process in two high enthalpy shock tunnels: the shock tunnel of Nagoya University and T5 at the California Institute of Technology. The computational domain covered the final 3 cm of a 9 m shock tunnel of radius 5 cm, with the full 198 cm length of the nozzle in the Nagoya case and the first 30 cm of the T5 nozzle from the secondary diaphragm. With such a small physical computational domain fine grid resolution was achieved. Their simulations use an axisymmetric, compressible Navier-Stoker solver which employs a hybrid scheme of implicit and explicit methods, along with AUSM to evaluate inviscid fluxes. The simulations focuses on the thermal and chemical non-equilibrium characteristics of the flow at the nozzle inlet in the initial stages of the nozzle starting process. The thermochemistry of the flow was modelled using a five species chemical reaction model for air and a two temperature model for thermal non-equilibrium.

CHAPTER 4

Review of Parallel Computing

The limit on resolution and complexity in Computational Fluid Dynamics (CFD) simulations of high speed compressible flows is determined by the available computing power. From the early 1950s to the present, the peak performance of the fastest computers available has increased approximately two orders of magnitude every decade [119]. This has meant that simulations of ever increasing detail have become possible as technology has improved. Despite the rapid and continuing improvements in performance of single processor computers, the high performance computing community has recognised that combining processors to work in parallel is a way of obtaining significantly more computing power.

The use of fast processors and efficient algorithms is not always sufficient to produce the simulations that are required in an reasonable time. In addition, as the development of computers continues, the performance of sequential computers may reach limits imposed by physical barriers such as the transmission speed of electrical signals and the physical spacing of the components on a computer or circuit, thermodynamic constraints and the high financial cost of production [63].

Combining computer processors to work together on a single simulation is way of obtaining more computing power. This is known as parallel computing. The increase in performance obtained is ideally proportional to the number of processors employed on the task. Parallel computing has developed to the stage where the efficient solution of compressible flow fields in parallel is now realistically possible; however, parallel computing remains a complex area and getting the theoretical performance out of an algorithm in practice is not always an easy task [101].

The concept of solving flow fields in parallel is nothing new. The first attempt at numerical simulation of fluid mechanics was described by Richardson [191]. In this book Richardson described his attempts at weather predictions using a mechanical calculator. Realising the inadequacies in these calculations, Richardson envisaged a room with 64,000 people seated in galleries in a large spherical room; each person would be responsible for a grid zone on a map of the world. The group would be

conducted from a person in the centre of the room, as the group marched through time and updated weather predictions were sent to cities around the world [119].

As the word *supercomputer* is used to refer to the most powerful computers of the time, this rapid rate of development has meant that the list of computers referred to as supercomputers is constantly changing over time, and, at present, the term refers exclusively to parallel computers. A Cray X-MP, released in 1982, and capable of just under 1 Gflops (peak), is now far beyond obsolete when compared to modern supercomputers, such as the APAC National Facility (a Compaq Alphaserver SC), with a performance of over 1 Tflops (peak).

Simulations performed in this thesis aim to simulate a complete shock tunnel, from the driver section to the dump-tank. Previous simulations of shock tunnels have, in the past, been limited by assumptions associated with only modelling part of a facility; this has been necessary due to the limitations of the computing power available. In order to model a complete shock tunnel facility with sufficient detail, large computational meshes are required. The fine resolution mesh simulations performed in Chapter 7, require one month of CPU time, even on the fast processors of the APAC National Facility. If this simulation was solved using four processors then this solution time would be reduced to one week. Running the solution on thirty processors would reduce the waiting time to around one day. These are much more practical time frames in which to conduct engineering research.

This chapter provides a survey of aspects of parallel computing hardware and software. The focus will be on general purpose parallel systems and the application of simulating of compressible fluid dynamics. Section 4.1 will cover the history of computing, concentrating on the high end of performance. Special importance will be given to the introduction and development of the parallel computer. Section 4.2 will outline the most commonly used classifications for computing hardware. Sections 4.3 and 4.4 will deal with software and programming models of parallel computers. These models identify particular features of computers of relevance to software developers. The aim of this chapter is to select a standard approach that can be used in compressible CFD code development. Chapter 5 will investigate the implementation of parallelism in the CFD codes MB_CNS and the SPH code.

4.1 Implementations of Computer Architectures

The digital computer had it's conceptual foundations with mechanical devices such as Charles Babbage's Difference Engine which was first produced in 1822 [119]. The use of mechanical devices for computational work had severe limitations in that the device could perform only addition and multiplication, and the numbers had to

be entered by hand - something greatly limiting large calculations. It was not until 1937 that George Stibitz realised that electronic logic gates could be used to perform arithmetic operations using binary numbers much more effectively than mechanical devices. World War II brought about the need for ballistics tables for artillery shells, and the large amounts of calculations required to write them. In 1943, six years after the first electrical relay was conceived, the technology had developed to a stage where the first electronic computer, the Mark I, went into operation to compute ballistic tables [119].

The World War II provided much impetus to the development of the digital computer and by the end of the war, probably the most famous of the early computers, ENIAC, was being built. This computer was followed by many other prototype computers throughout the late 1940's and 1950's, including EDSAC, BINAC and SEAC [250]. The designs of these early computers was along four lines of development, depending on which type of memory they used, either Acoustical Delay Line (ADL), Cathode Ray Tube (CRT), Core or Drum. Other notable computers of this period were UNIVAC I, the IBM 701 and the UNIVAC 1103. Core memory technology, which replaced CRT memories with ferrite cores, was the only type of memory to persist beyond 1957. Software was relatively undeveloped and was limited to assembly languages [119].

The history of digital electronic computing technology has been grouped into five generations of development [107]. Software technology has developed alongside hardware technology and so both are considered together. The early computers represented the first generation of computers (1946-1956) in this classification.

The second generation (1956-1967) is characterised by the introduction of discrete transistors. Memory technology was improved through the improved use of core memory and the high level languages Algol and Fortran were introduced. Representative computer systems of this period include the IBM 7030 and the Univac LARC.

The third generation (1967-1978) saw the introduction of (Small Scale) integrated circuit technology. The C language was introduced [120]. Raw computational performance was the focus for the development and the reason for the success of vector computers during the 1970s. Computers of this period included the PDP-11 and the IBM 360/370. [224].

The forth generation (1978-1989) saw a paradigm shift in the use of computers: the introduction of the IBM PC, and its compatibles, to the home user market. Very Large Scale integrated (VLSI) Circuits were developed as well as solid state memory. The first parallel computers appeared around 1978, and included the CDC6600 and

the IBM360/91. Also, during this period the first concerted effort to implement multiple processor and vector technology in supercomputing was attempted. Symmetric multiprocessing computers were first developed, an example being the Cray Y-MP. On the software side, message passing libraries were introduced as well as parallelising compilers [210]. Supercomputers of this period include the Cray X/MP and Vax 9000, but by far the greatest impact on computing during this period was the advent of the PC. Many people now had access to significant computing resources. During the 1980s, the availability of standard development environments and application software packages became more important. Figure 4.1 shows the relative speeds and the time frame of the development of computing technologies from the 1920s to the 1990s. The figure's vertical axis is the computing power (in operations per second), divided by the cost of the computer and is plotted with a logarithmic scale against the horizontal axis of years.

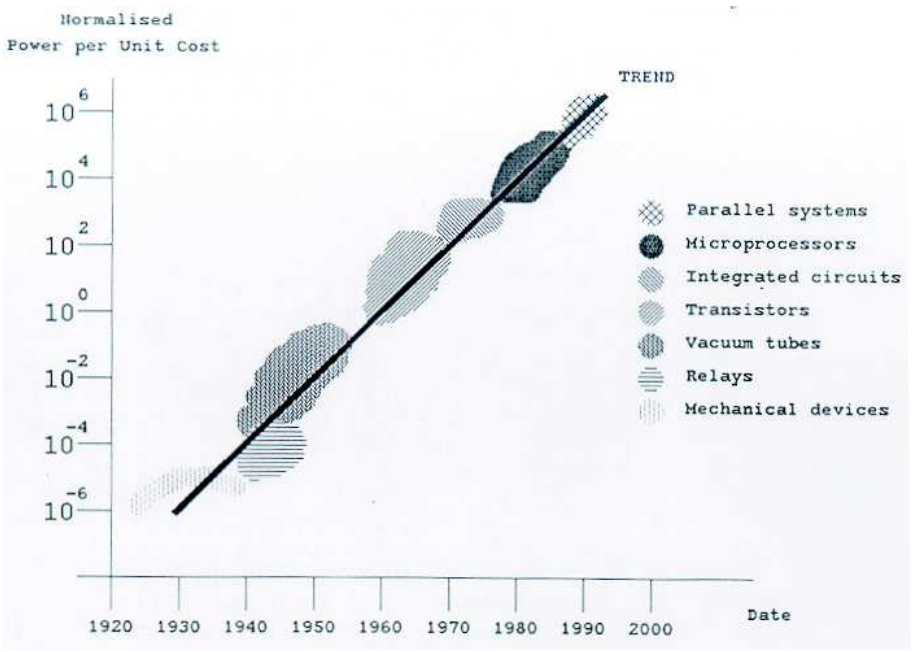

Figure 4.1: The increasing speed of early computing technology. Reproduced from Kaufmann III and Smarr (1993) [119].

The fifth generation (from 1990 to the present) featured the impact on general computing brought about by the introduction of the Internet. The importance of scalability [13] and portability [56] in parallel computers was realised. Object-oriented languages such as C++ were introduced, as well as the platform independent language Java [14]. The potential of combining workstations into clusters for cheap and efficient parallel computations was made a reality as mass market processor, I/O and networking technology improved at an unprecedented rate [33]. Processor technology extended to the use of Ultra Large Scale Integrated (ULSI)

circuits. Characteristic supercomputers of this period include the IBM SP-2 and the Cray/SGI Origin 2000. Massively Parallel Processor (MPP) systems became more popular during the 1990s due to their better price to performance ratios and seemingly limitless scalability. In the medium performance markets; however, MPPs were replaced by the easier to program Symmetric Multi-Processor (SMP) systems as the decade progressed [224].

Despite the rapid increase in computational power of computers even in the mid-range, the CFD practitioner's insatiable appetite for computing power means that they are primarily interested in the supercomputers of the day.

4.1.1 A Short History of Supercomputing

It is generally accepted that modern supercomputing began with the delivery of the first vector computer, the Cray 1, to the Los Alamos Scientific Laboratory in 1972 [119]. Vector computers process arrays (vectors) rather than single data items (scalars), using arrays of fast registers and pipelining of data and instructions. Before the Cray 1, supercomputers were still scalar systems and did not differ significantly in their architecture from main stream computers. The Cray 1 architecture gave it a performance advantage of over an order of magnitude above the fast scalar systems of the time [224]. Three Japanese computer manufacturers (Fujitsu, Hitachi and NEC) entered the high performance vector computer arena at the end of the 1970s. These systems were, at first, sold only in Japan, but were later exported to the USA and Europe [224].

The Cray X-MP was released in 1982 and contained two processors, making it the first Parallel Vector Processor (PVP). The system was enlarged to four processors in 1984. At first the use of multiple processors was aimed more at increasing the throughput of computing centres with independent jobs running on each processor, but the development of parallelisation within a single program soon followed. The Cray 2, first delivered soon after, in 1985, offered a peak performance more than twice that of a four processor Cray X-MP. The Japanese manufacturers preferred to concentrate on chip technology and multiple pipelining, but soon followed the trend and introduced multiple processor versions of their computers [224]. These PVP computers enjoyed great commercial success during the 1980s.

The 1980s were also the period in which parallel computing became wide spread, with many different companies developing their own architectures, models and languages. Vector computers were being produced by Cray, Control Data Corporation (CDC) (later ETA), Fujitsu, Hitachi, NEC and Convex. Other Single Instruction, Multiple Data (SIMD) systems were being produced by the Thinking Machines Cor-

poration and MasPar [208]. Parallel languages included CSP [216], Occam [216], NIL, Ada, Concurrent C, Distributed Processes, SR, Emerald, Argus, Aeolus, Par-Alfl, Concurrent PROLOG, Linda and Orca, all of which are described in Bal et al. (1989) [10]. Many of these languages were specific to a particular architecture and many featured implicit forms of parallelisation in which the programmer was less involved in how the job was distributed amongst the processors. Vectors processors had been successfully programmed via vectorising compilers for some time, but parallelising compilers were not nearly as successful [123].

Implicit parallelisation was a very attractive option from the programmers point of view; however, the success of these languages was limited by a few very important factors [210]. The promise of reliable and automatic parallelisation of sequential algorithms remained far off and programmers were reluctant to take the gamble of moving away from mainstream languages, such as C and Fortran. This feeling was compounded by the apparent comings and goings of computer companies experimenting by releasing new and untested technologies [123]. For users, investing large amounts of time in a language that was developed for parallel execution on a specific architecture, which may not prove useful in the long run, seemed too much of a risk. This meant that many of these languages failed to attract the large user base that they needed for survival. Many of these specialised languages seemed promising, but soon faded away, only adding to the uncertainty facing programmers [10].

In 1984, a standard UNIX operating system, UNICOS, was introduced to all Cray systems. This innovation, coupled with the availability of vectorising compilers, meant that more software vendors started porting their applications to Cray systems. The obvious commercial benefits soon followed with Cray selling a large number of systems in the automotive and oil industries. This made an important contribution to Cray dominating supercomputing for more than a decade. UNICOS is still available today in Cray/SGI supercomputers [204].

The success of Cray showed that the importance of standard development environments, across and between ranges of computers, cannot be understated [250]. Raw computational performance drove the development and resulting success of vector computers during the 1970s. Portability and scalability, today seen as two essential qualities for computing systems, were not nearly as important at that time [224]. At the start of the 1980s, as users gained more experience with parallel systems and found that they were wasting time moving even simple programs between systems, people's attitudes began to change. When an obsolete system was being replaced with a new system, the availability of standard development environments and application software packages became very influential in the decision of which computer to purchase. The importance of portability in computing systems began

to make its mark; systems that did not provide compatibility with other systems did not survive. Of the 14 major manufacturers in the early 1990s, only four survived, that is the three Japanese manufacturers and IBM (which at this stage was still only just entering the high performance market). Around this time, SGI, Hewlett-Packard (HP), Sun and Compaq also entered the supercomputing market, mainly by buying smaller companies [224]. In addition, HP and Compaq have recently merged.

In the second half of the 1980s, parallel computers began to appear with distributed memory, these were known as Massively Parallel Processors (MPPs) [107]. This design removed the limitations on the scalability of the shared memory designs of parallel vector processors. The first MPPs were developed by the US Defense Advanced Research Agency (DARPA), wanting to build computers as large as possible as part of the Strategic Computing Initiative (SCI). Soon there were many companies developing and producing MPPs, including Intel, nCube, Floating Point Systems (FPS), Kendall Square Research (KSR), Meiko, Parsytec, Telmat, Suprenum and BBN [224]. The list of the fastest 500 supercomputers in the world is available at online [235] and is updated every six months. In June 1993, the first MPP systems began to appear on the top 500 list.

MPPs made their debut at a time when the importance of portability and scalability were becoming more obvious. In terms of portability, using "explicit-everything" parallelisation, in which the programmer specifies all aspects of communication between processes, started to stand out as the only practical solution available. Algorithms based on the explicit-everything approach can be executed on far more systems as there is little ambiguity in the code and therefore reliance on details of the platform being used. The down side is that this approach requires significantly more work to develop a high performance code. The scalability of MPPs was, and still is, unmatched; by physically distributing the memory, and by using message passing libraries, the systems could be scaled to thousands of independent processors [210, 38].

These MPP systems provided another important advantage in that, unlike the PVPs available at the time, they were produced using largely commodity processors and networking hardware. This meant that, by taking full advantage of mass production of its components, the cost of the systems could be greatly reduced. Intel produced the Paragon/XP series based on its i860 chips and technologies which were being developed for desktop PCs. IBM produced the SP1 and SP2, based on their popular RS6000 workstations [224].

This trend of using commodity components, driven by the potential decrease in cost, has led to a subclass of MPPs, known as workstation clusters. Clusters

use actual workstations, connected together by standard networking components. Parallelism is most commonly achieved through message passing. Clusters, by definition, use completely Commercial Off The Shelf (COTS) components, meaning that a cluster can be built entirely from hardware purchased at a local computer store - a very different approach to the proprietary PVP systems popular in the 1980s [223].

As architectures moved away from PVPs and toward MPPs, software became the limiting factor in scalability. An answer to the problem of finding a suitable parallel language is to use parallel languages actually built around the widely used sequential languages, C and Fortran. These new languages include Fortran 90, High Performance Fortran (HPF) [100], Parallel C++ and Parallel Computing Forum (PCF) Fortran. These languages attempt to open up the field of parallel computing to many more users, but are still limited in portability by architecture related issues. To achieve acceptable performance, the programmer must still address the data layout problem and data motion problems, as well as other issues of portability [123].

Parallel languages still did not have the attraction to draw people away from the main stream sequential languages. An extension to the idea of parallel languages built around the mainstream sequential languages is to provide libraries that can be called from what would normally be sequential C or Fortran code which specify the parallel functionality. The structure of explicit message passing is well suited to the use of these libraries. The first parallel library which ran with C and Fortran was the Parallel Virtual Machine (PVM) [75]. Message Passing Interface (MPI) libraries [161] and variations, such as the super-step communication of Bulk Synchronous Parallel (BSP) libraries [171], followed. MPI libraries have been by far the most widely used form of parallelisation for high performance applications since the early 1990s.

A third approach, based on specialised hardware, is to physically distribute the memory, but enable programs to appear as if they are using a single memory space. An example of this type of architecture is the SGI/Cray Origin 2000, which implements a Cache Coherent, Non Uniform Memory Architecture (CC-NUMA). This approach still requires the use of parallel libraries, such as OpenMP [168]. The programmer tells the compiler which sections of the algorithm should run in parallel and provides prompts as to how this should be done - it is not necessary to provide all details of the parallelism as with the explicit-everything approach. Currently the performance of distributed shared memory libraries, such as Treadmarks [5], for other than specially designed hardware is somewhat limited.

It was thought by many people that parallel computing would take over from

sequential computing, but, despite its promise, this has clearly not happened. For parallel computing to be a practical solution for the needs of high performance computing in the future, much work still remains to be done in the formalisation of models, the standardisation of programming languages and the development of hardware. Improvements in processor technology are starting to show signs of reaching fundamental physical limits and this will only raise the importance of parallelism in future computer architectures.

It appears that evolutionary rather than revolutionary development of technologies have been the basis of development in the supercomputing industry; revolutionary changes have rarely been able to make an impact on the market. In recent years only companies that have participated in the markets for massive database management and financial applications have been able to maintain enough business to be able to develop specialised hardware for the numerical high performance market as well [224].

At present, the upper end of the top 500 list [235] is dominated computers built for the Accelerated Strategic Computing Initiative (ASCI), which aims at simulating nuclear weapon tests. ASCI Red was installed at the Sandia National Laboratory in 1997, and with its 9472 Intel Pentium Xeon processors was the first to exceed the 1 Tflops mark on the LINPACK benchmark. ASCI Blue Mountain, a cluster of SGI Origin 2000 systems containing 6144 processors, achieved 1.6 Tflops. The next computer in the series, ASCI Blue Pacific, built by IBM, ran at 3.87 Tflops. The recently installed ASCI White, also built by IBM, incorporating 8,192 processors, runs at 12.3 Tflops. That makes White three times faster than Blue Pacific, installed one year before. Projections for the ASCI program call for a 100 Tflops system by April 2003 [224].

The vast scale of numerical simulations capable of simulating weather patterns is realised to a greater extent today than in the time that Richardson proposed his weather predicting room [191]. The first machine thought to be capable of such simulations has recently been constructed. This machine, known as the Earth Simulator, consists of 640 NEC SX-7 VPP nodes, each with eight vector processors and runs with a maximum performance of 35.86 Tflops (peak performance is 40.96 Tflops). The computer was funded by the Japanese Government and it is aimed at simulating the global weather environment. To put this computer in the perspective of Richardson's weather prediction room, the average (numerically skilled) person requires around 100 seconds to perform one floating point calculation with two thirteen digit numbers. This corresponds to a maximum performance of 10^{-2} flops [119]. This means that it would require the whole room of 64,000 people working together for over 1800 years to reproduce the number of operations performed by the Earth

Simulator each second [69].

Figure 4.2 shows a graph of the peak computing speed achieved by supercomputers over past thirty years. The same general trend of computer speed through time is evident throughout the graph. The speeds of these computers are still consistent with the two orders of magnitude per decade trend.

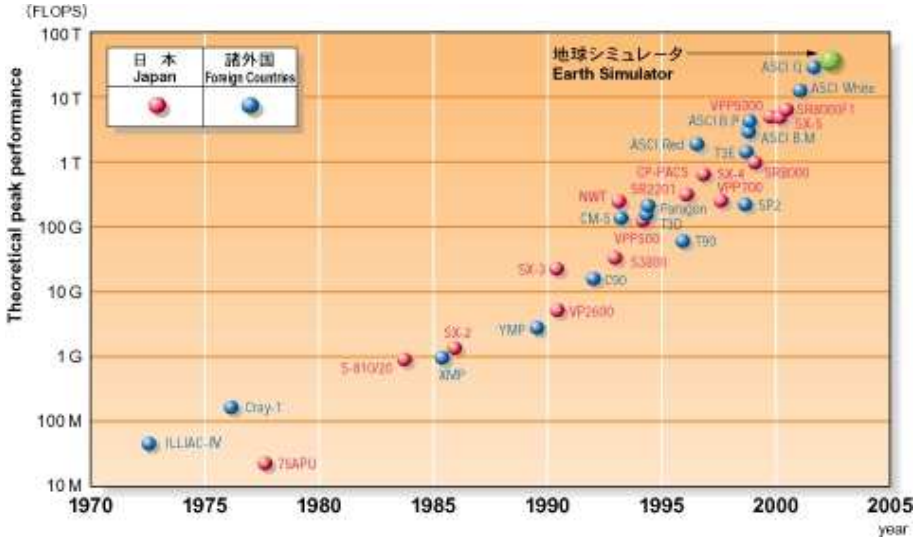

Figure 4.2: The peak performance of computers over the past 30 years. Reproduced from the Earth Simulator computer internet page [69].

4.2 Architectures for Parallel Hardware

4.2.1 Flynn Taxonomy

In 1966, Flynn [70] divided computers into four basic classes based on parallelism in both the flow of data and instructions to the processors. These classifications (known as Flynn Taxonomy) are still useful today; however, there has been significant diversification in the areas described and many systems do not fit neatly into any one category.

SISD (Single Instruction stream, Single Data Stream) describes any sequential processing single processor computer. This class includes any computer with a single scalar processor, such as standard desktop PC's. This style of computer is also known as a von Neumann computer which is characterised by the division of a task into separate sequential elements such as fetching a piece of data followed by performing some operation on that data and then storing the result.

SIMD (Single Instruction stream, Multiple Data stream) describes a model of parallel execution in which all processors must execute the same operation in the same clock cycle, but acting on different data. Vector processors are a form of SIMD computer in which vectors of data are operated on by processors specially designed for processing whole vectors simultaneously. SIMD computers use data parallelism meaning that the parallelism is achieved through acting on difference segments of the data set simultaneously. The processors are usually mesh connected and communication tends to be efficient because the operations on each processor are inherently synchronised. SIMD computers include the MASPAR, the Connection Machine and Vector processing computers.

MIMD (Multiple Instruction stream, Multiple Data Stream) describes an assembly of processing elements which are more free in their operation than SIMD processors. The degree to which the processors are independent in their operation is not specified by the MIMD model, nor are details of the interprocess connectivity. The processors are free to perform their operations, which are followed by communication and a global synchronisation. The granularity of parallelism describes the relative amount of computation that is done between synchronisation. The granularity can range from the extreme fine grained parallelism of SIMD computers to coarse grained parallelism, in which large amounts of data is processed before the results are shared between the processors.

MISD (Multiple Instruction stream, Single Data Stream) There are no known implementations of this type of computer [28], although, it can be argued that pipelining data is an example of MISD parallelism. Pipelining increases computational efficiency by directly connecting the output of a specialised computational unit to the input of another in a line. As a result of its basic structure, a single piece of data is not being acted on simultaneously by the elements and therefore it is not generally accepted as SIMD parallelism [166].

SPMD (Single Program, Multiple Data) describes an extension of the MIMD class and is an addition to the basic four areas that Flynn described. SPMD programs consist of a series of homogeneous processes, that is, it is the same piece of code being executed on different segments of a data set; however, there is not the level of synronisation seen in SIMD computers. This class is the one of particular interest to CFD as it allows parallelisation through domain decomposition; several processes using the same algorithm can work on the data describing different sections of the flow field.

4.2.2 Physical Machine Models

Currently used large scale computer systems are grouped into six physical machine models, SIMD machines (which were described earlier) and five types of MIMD machines [107]:

PVP (Parallel Vector Processor) systems differ from other types of systems in that they comprise a number of powerful vector processors, that simultaneously apply the same arithmetic operations to different data. Many of the building blocks of the machine are custom made, particularly the specialised vector processors themselves. A high-bandwidth crossbar switch is used to connect these vector processors to a number of shared memory modules. PVP machines generally do not need to use memory caches as their whole memory is fast. Data and instructions are usually buffered before reaching the vector registers. PVP systems include the Cray C-90 and T-90 and the NEC SX-5.

SMP (Symmetric Multi-Processor) systems utilise commodity processors with both on-chip and off-chip caches. The processors are connected to a shared memory through a high speed bus and a crossbar switch. The basis of the SMP architecture is that all processors must be symmetric, that is, all processors must have equal access to the shared memory, known as Uniform Memory Access (UMA), the I/O devices and the operating system services. SMP systems include the Sun E10000 and the Intel SP2.

MPP (Massively Parallel Processor) systems utilise commodity processors as computational nodes and memory physically distributed over these nodes in private address spaces. The MIMD processors execute asynchronously, and interconnection between the nodes is achieved through high bandwidth, low latency networks. One of the main reasons for using MPP systems is that they can be scaled up to an arbitrarily large number of nodes, with processors being synchronised, and communication being achieved through message passing operations. MPP systems include the Intel Paragon and the IBM SP-2. The scalability of MPP systems has led to their extensive use in the ASCI project; ASCI White is an MPP system made up of 8,192 processors.

DSM (Distributed Shared Memory) systems differ from SMP systems in that the memory is physically distributed and that, in general, individual processors do not have equal access to all sections of the memory, known as Non-Uniform Memory Access (NUMA). Although the memory is physically distributed, specialised hardware and software create what is, to the programmer, a shared memory system with a single address space. These systems have become very popular over recent years as they provide an extremely usable platform for parallel code development whilst maintaining good performance and scalability. DSM systems include the SGI Origin 2000 and the Convex Exemplar.

Workstation Clusters are similar to MPP systems, except that the nodes are actually workstations with any unnecessary peripherals, such as keyboards and monitors, removed. The modular nature of the PC has exemplified the potential of scalable computers because their individual components can be upgraded whilst maintaining the rest of the system [33, 2]. Examples of this are that memory can be added to sockets on the motherboard and network interface circuitry can be upgraded simply by replacing a card together with some packaged software. One of the first examples of a workstation cluster was the Beowulf Cluster developed in 1994 by Sterling et al. [223].

During the 1980s, when PVP systems were most popular, there were great differences in price, performance and technology between so called supercomputing hardware and the hardware that was released on workstations. These differences have decreased dramatically over the last decade and, at present, most large supercomputers are built from components that are comparable to, or even interchangable with, commercial off the shelf (COTS) components available for workstation computers. This mass development of high performance computing hardware has benefits at both ends of the computing spectrum, increasing low-end performance, and reducing high-end costs. The performance of COTS networking and I/O components

is also increasing rapidly, offering the prospect of an even less well defined distinction between traditional supercomputing and what can be built with commodity hardware.

Figure 4.3 shows the numbers of different architectures in the top 500 list over the 1990s. This was the period during which MPPs began to dominate high performance computing. An increase in the number of MPPs in the list is evident as well as the absence of any single processor or SIMD architectures after 1997. Workstation clusters entered the list in 1998 and have their number has increased steadily since then. These computers are evident in the top right corner of the plot. The ordering of the architectures does not imply a ranking in speed for the machines; however, SMPs and SMP Clusters are placed side by side to show how SMP clusters are replacing the singular machines in the list.

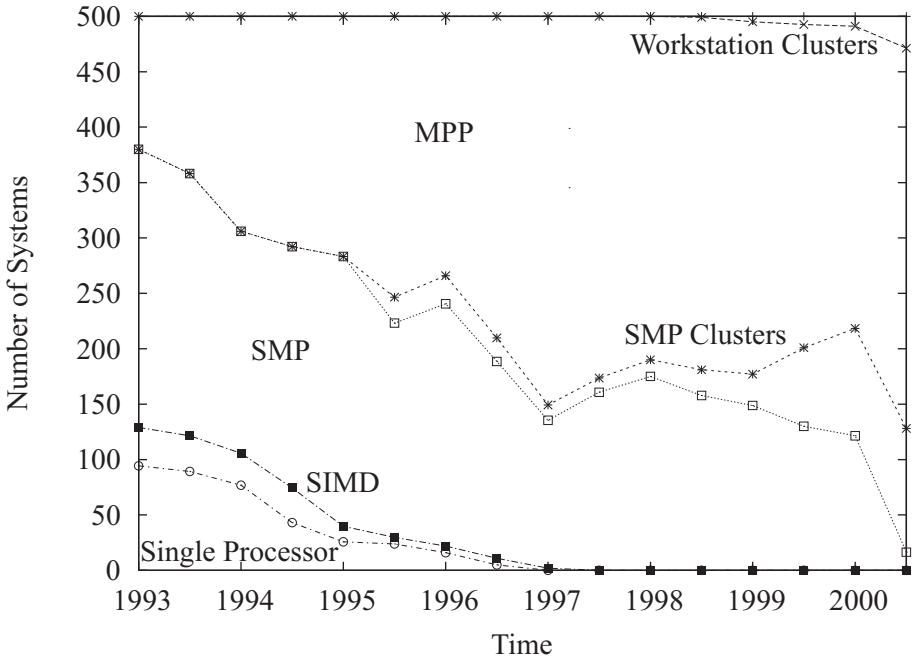

Figure 4.3: Cumulative plot of the number of each type of architecture in the Top 500 list over the period from 1993 to 2000. The type of architecture is labelled in, or near, its area.

4.3 Models for Parallel Computers

Abstract machine models are used in the design and analysis of algorithms. They are conceptual models that allow details about the physical design of the machine to be ignored. Abstract parallel models characterise those capabilities of a parallel computer that are fundamental to parallel computation. The abstraction does not actually provide any structural information or explicit implementation details, but should allow a relatively precise representation of the performance of the system and the relative time costs of parallel computation. An effective parallel model must provide three fundamental properties: scalability, portability and predictability [107]. Actual computer architectures are based on these abstract models for all but the most specialised computer systems.

The aim of abstract parallel models is to provide programmers, software developers and computer designers a model that represents the fundamental issues associated with high level parallel implementation. An abstract parallel model can be characterised by several semantic attributes and performance attributes. The semantic attributes recognised are [107]:

Homogeneity indicating how alike the processors of a parallel computer behave when executing a program in parallel. Using Flynn Taxonomy, most models utilise a Multiple Instruction stream, Multiple Data stream (MIMD) setup. If at each cycle all processors must execute the same instruction (that is there is only one instruction stream) the system will be a Single Instruction stream, Multiple Data stream (SIMD) machine.

Synchrony indicating how tightly synchronised the processors are. The tightest level of synchrony is at instruction level, meaning that at each cycle, all memory read operations from all processors must be performed before any processor can perform a memory write or a branch. Real MIMD machines are asynchronous, that is, each processor executes at its own pace, independent of the speed of the other processes and if a process has to wait for other processes, additional synchronisation operations must be executed.

Interaction mechanism indicating how parallel processes may interact to affect the behaviour of one another. Commonly, this is either through shared variables (or shared memory), where all processors can access the same shared variables or though message passing in which the variables of all processes are invisible to other processes and, when required, must be explicitly sent and received between them.

Address space is the set of memory locations accessible by the processes. In some systems, all memory locations reside in a single address space (from the programmers point of view), while in others each process has its own memory address space. A relationship between interaction mechanism and address space exists, in that, for a system with a single address space, shared memory is much more conveniently implemented and for a system with multiple address spaces, message passing is more convenient.

Memory model specifies how the system deals with shared memory address access conflicts. Consistency rules are used to resolve these conflicts. The most strict of these is the Exclusive Read, Exclusive Write (EREW rule) by which a memory location can be read or written by at most one processor per cycle. The Concurrent Read, Exclusive Write (CREW) and Concurrent Read, Concurrent Write (CRCW) rules allow more flexible management in allowing memory locations to be accessed by multiple processors, but introduce the need for conflict resolving policies.

4.3.1 Parallel Performance and Overheads

Performance attributes are of direct interest to the CFD programmer, but are highly platform dependent. Commonly used performance attributes include [107]: the number of processors, the clock rate (in MHz), the sequential and parallel execution time and various ratios including the speedup and efficiency due to parallelisation. These parameters are tightly coupled to another attribute, the scalability of a parallel system.

Although many people regard clock speed as the defining factor in processor speed, especially in the home computer market, there are many aspects of processor design that define the actual rate at which useful calculations can be done. The use of memory hierarchy is a very important aspect of efficient processor design. The peak speed at which modern processors can process information is much faster than the currently available general memory. To cope with this speed difference data is retrieved from the slow main memory in chunks and is buffered in very fast caches. This allows the processor to have a much more consistent flow of data supplied to it.

The aim of executing an algorithm in parallel is to divide the work up between the processors. Thus, for parallelism with an efficiency of one, two processors would run an algorithm twice as fast. This is never the case as there are always "overheads" associated with running the algorithm over multiple processors. Overheads

are defined as any extra work that must be done, in addition to the useful work, purely to achieve parallelism. Overheads include:

Communication overhead the work required to share updated variables between processors

Synchronization overhead the work required to align the processors at a particular point in the algorithm which depends on all variables being up to date

Parallelism overhead which accounts for miscellaneous overheads including assigning new processors to do work, dealing with the operating system and closing connections to processors after being used

Another overhead which may be a factor in reducing parallel efficiency is the load imbalance overhead. It is due to processors having an unequal amount of work to do. Commonly this may also be due to the use of processors of differing speed, but given the same amount of work to do. Load balancing techniques can be used to ensure that the work is shared appropriately and that at all times processors have useful work to do. Static load balancing can be used when the relative amounts of work that should be assigned to each processor is known before hand.

4.3.2 Scalability

A computer system is said to be scalable if processors, memory and I/O components can be added to facilitate an increase in performance and functionality. Ideally this increase should be proportional to the increase in system resources [107]. The most obvious way of scaling a parallel system to achieve greater performance is by adding more processors; however, there is a trade-off in that increasing the number of processors also increases the overheads associated with communication and synchronisation. This means that the increased amount of work that can be done may not all be useful work. Parallel efficiency is the actual speedup divided by the theoretical speedup from running on multiple processors. Efficiencies of around ninety percent can be obtained with efficient codes, running on MPP systems. An inefficient use of parallelism is a waste of computing power and in many cases, an excessive number of processors will often severely impede performance.

Scaling a computer by increasing the number of processors alone is insufficient to achieve an increase in performance. A more accurate way of thinking of this type of scaling is in an increase in actual machine size as the resources used by the processors must also be scaled appropriately. The communication subsystem (including the interconnection, the interface and the associated software) may often

have to be improved to handle the increased load. Another key resource that must be scaled is memory, including cache memory. Some systems scale better than others due to their architectures; the SGI Origin 2000, being a Distributed Shared Memory (DSM) system is limited to 128 processors [204], whereas Massively Parallel Processor (MPP) systems are can be scaled to an unlimited number of processors due to their more straight forward interconnection network. ASCI White which consists of 8,192 processors is an example of these systems.

A second form of scalability, technology scalability, refers to the ability to replace components of a computer with newer, more advanced, versions. This scaling is not limited to parallel computers, but is common to any component based system. An example of this are the processors on the Origin 2000; an SGI Origin 2000 computer purchased in 1997 would have featured the R10000 processor, which in 2001 was upgraded to the R12000 processor, offering improved performance. Another example, relevant to a workstation cluster is the upgrading of 10 Mbit/s Ethernet networks to 100 Mbit/s Fast Ethernet networks by upgrading network cards, switches and cables, whilst being able to retain all hardware not directly involved in the networking.

Software scalability is, in many ways, just as important as hardware scalability. Software scalability includes adding new versions of operating systems, drivers and applications, especially better compilers and libraries. Software scalability not only allows a system to keep abreast of the current level of software development, but also to take advantage of system improvements resulting from hardware scaling, although, software should not become obsolete with the introduction of newer generation hardware. In contrast to the fast pace of hardware technology, application software often moves much more slowly. For example, Fortran 77 is still widely used, although new compilers are released every few years.

The aim of a scalable system is to provide a flexible, cost effective computational tool that will be able to handle ever increasing problems and applications over time [107].

4.3.3 Parallel Random Access Machine (PRAM)

The most general abstract parallel computer model is the Parallel Random Access Machine (PRAM) model, proposed by Fortune and Wyllie [71]. The PRAM model is the parallel computer analogy to the von Neumann model for sequential computing. A system based on the PRAM model will access multiple instruction streams and multiple data streams. It consists of an arbitrarily large number of processors in parallel, all accessing a shared memory space. The PRAM model is synchronous at instruction level, that is, tightly synchronous. At each cycle, all memory read

operations from all instructions must be performed before any processor can perform a memory write. At each cycle each processor executes exactly one instruction, which may be a null instruction in the case where the processor is idle in that cycle. A single instruction can be either one of three operations: fetch one or two words from the memory as operands, perform an arithmetic logic operation on loaded data or store the result back into a memory address [119, 77].

The interaction mechanism of the PRAM model is through shared memory, with all memory locations residing in a single address space. Additional to this all processors must be able to access all memory locations in the same amount of time, known as Uniform Memory Access (UMA). Machines that do not satisfy this condition are known as Non Uniform Memory Access (NUMA) machines. Memory address conflicts in PRAM machines are resolved by the Exclusive Read, Exclusive Write (EREW) rule. The PRAM model does not take into account overheads: due to the cycle level synchrony the synchronisation overhead is assumed to be zero and the communication and parallelism overheads are ignored. This means that the only overhead that is included in the PRAM model is the load imbalance overhead, accounted for in the fact that at a cycle, a processor may execute a null instruction.

The PRAM model is idealised and, although computers are based on the model, only some of its features are ever used in any one system [126]. The Origin 2000 is an example of a computer based on the PRAM model that does not meet the specification fully. It consists of a number of symmetric processors accessing a common memory space, but the system does not provide UMA. This is because the shared memory space is physically distributed amongst the processors, known as Distributed Shared Memory (DSM), and memory on the local processors is faster to access than memory stored on other processors. With this approach, processors can write to different copies of variables to one another, so the validity of the data in each processor's cache must be ensured. This is done through a Cache Coherency (CC) protocol, thus the Origin 2000 a called a CC-NUMA system.

4.3.4 Bulk Synchronous Parallel (BSP)

The Bulk Synchronous Parallel (BSP) model was introduced to overcome the shortcomings of the PRAM model, while attempting to maintain its simplicity [237]. The aim of the BSP model is to build on the PRAM model by introducing additional parameters to capture the overheads associated with the execution of a program in parallel. Like the PRAM model, the BSP model is based on a MIMD system as all processes can execute different instructions on different data within the same time step. A BSP system consists of a set of processor and memory pairs that are

connected by an arbitrary interconnection network.

Although the basic unit of time is still the cycle, the program executes as a strict sequence of "super steps". During each super step, computation occurs and at its completion there is a communication operation between the processors followed by a barrier synchronisation. Both of these operations introduce overheads which can be easily quantified. This barrier causes all of the processors to complete the current super step before any can proceed to the next superstep. This means that the BSP model is loosely synchronous (at the super step level), compared to the instruction level, tight synchrony of the PRAM model [194]. Within each of the super steps, different processes execute asynchronously. The implication of this is that the BSP model is closely related to the SPMD class of computers and more closely approximates the behaviour of some real parallel machines.

The BSP model uses a single address space in which a processor can access not only its own local memory, but also any remote memory local to another processor. The model does not explicitly require any specific form of interaction mechanism, but usually implements a shared memory and message passing. Within a super step a processor can access only data in its own local memory except for at the barrier synchronisation. This means that on each processes computation occurs independently of the other processes and that the writing of all data must be completed before any data is read (during the computation of the next super step).

The BSP model presents a more realistic representation of a real computer system, because it accounts for all overheads except that for process management [126]. The execution time of a super step is the sum of three components: the computation time, the synchronisation overhead and the communication overhead. The computation time is the maximum number of cycles spent on computation operations by any processor, also accounting for load imbalance (this was the only overhead accounted for in the PRAM model). The synchronisation overhead is then the time taken to align the processes, forming a barrier up to where all work which must be completed before any process can continue. The synchronisation overhead has as its lower bound the network latency (that is the time for a word to propagate through the physical network). The communication overhead is related to the time taken for each node to send a series of words to various nodes and for these nodes to receive the words. The BSP model does not allow for the overlapping of these overheads. A further refinement to the PRAM and BSP abstract parallel models is the Phase Parallel model, but BSP model is sufficient for analysing the parallel performance of SPMD applications such as most CFD applications [107].

The concepts described by the BSP model form the basis for its own special library, BSPlib [171], developed at Oxford University. As well as this, BSP as

a parallel computing model can also be used as a basis for the structure and the analysis of parallelisation with other libraries, including MPI. These are the software models, along with OpenMP, that will be used to build the parallel CFD codes described in Chapter 5.

4.4 Parallel Programming Software Models

4.4.1 Explicit parallelisation

Explicit parallelisation means that it is the programmers responsibility to use language constructs, compiler directives and library functions to specify all aspects of the parallel execution of the program; the parallelising procedures must be explicitly specified in the source code. The three most dominant explicit models are:

Data parallel involves executing the same instruction on different data streams in the same clock cycle on multiple processors; this is the SIMD model. Parallelism is exploited at the data set level with a single instruction stream and loosely synchronous operations. Although specifying the data layout, the parallelism is often implicit in the applications through loop parallelisation. Languages using this approach include Fortran 90 and High Performance Fortran (HPF) [190, 133].

Message passing involves a program executing with multiple processes in which the programmer is responsible for almost all aspects of the parallelisation such as data handling, communication and synchronisation operations. The individual processes may execute asynchronously, with barrier and blocking communication processes to ensure correct execution order when necessary. The processes all use different address spaces so that data variables local to one process are not directly available to other processes. The processes cannot read from, or write directly to, each other's address spaces and all communication is by message passing operations. As well as this, both work load and data must be explicitly allocated by the programmer. Applications intended for message passing systems usually exploit coarse grained parallelism to minimise the effect of latency. The two standard libraries implementing this model are the Parallel Virtual Machine (PVM) [229] and the Message Passing Interface (MPI) [87, 215], which can be implemented on nearly all types of parallel computers.

Shared memory programs feature multiple threads (the shared memory analogy of a process), all accessing a single shared address space. Threads operate asynchronously, and the workload can allocated either explicitly or implicitly. Communication is done implicitly through shared reads and writes of variables. Explicit synchronisations can be performed to maintain the correct execution order between the threads. For a long time the shared memory model lacked a widely accepted standard (comparable to MPI). OpenMP has recently emerged as a firm standard [168]. The OpenMP standard is backed by

the major manufacturers of shared memory machines including SGI, IBM, Intel and Compaq. The shared memory model is a programming model and can be implemented on any MIMD system. Some recent attempts at implementing the model on MPP systems, where software transparently performs the communication of variables between threads, have met with inefficient results [51]. In contrast, DSM systems running OpenMP, in which the communication is performed transparently by a sophisticated hardware and software combination, have been very successful.

4.4.2 Implicit Parallelisation

Implicit parallelisation is when the programmer does not specify to the compiler the particular aspects of parallelism is achieved, but rather, at most prompts the compiler as to where to exploit it. The most popular approach to achieving implicit parallelisation is by the use of automatic parallelising compilers on sequential programs that have already been written. The compiler first performs a dependency analysis on the sequential program, this involves both data dependence and control dependence. If an operation in one process requires data from another process then the latter process must be completed before the former can continue; however, if two processes are determined to be independent, then they can be scheduled to be executed in parallel. Most techniques for automatic parallelisation focus on restructuring techniques for exploiting parallelism in Fortran "do" loops and C "for" loops. One major advantage of implicit parallelisation is that it allows previously written sequential programs (which may have been developed and verified at large cost) to be used on parallel systems. This also means that the programs are theoretically portable between different systems [210]. An examples of a commercially available implicit paralleliser is the KAP automatic parallelising compiler [124].

The efficiency of parallel code generated from sequential code by automatic parallelisers has often been questioned. Manually transformed codes can generally show significant improvement over those generated automatically, even by current state-of-the-art parallelisers [21]. Automatic parallelisers have significant potential for improvement as the majority of transformations that are normally applied manually should be able to be automated. Ideally transformations should be completely derivable from the source code, without requiring any information about the application. In 1992, Blume and Eigenmann [21] conducted a study of the effectiveness of the technology implemented by automatic parallelisers and showed that particular aspects had not reached a level of maturity such that effective parallel efficiency could be achieved reliably. There has been limited success since that time in finding answers to these inadequacies and the focus of development has shifted to explicit

methods of parallelisation [87].

With some automatic parallelisers, the programmer can give "directives" to the compiler by providing it with more information. For example, if all iterations of a complex loop are known to be independent, the programmer can provide this information to the compiler. These directives can allow the compiler to do a much better job of parallelising the sequential code, and so in this way, the distinction between explicit and implicit parallelisation has become blurred. OpenMP is considered explicit parallelisation even though all aspects of the parallelisation need not be specified to the compiler.

4.5 The Choice of Parallel Methods.

Using processors in parallel gives the user access to computing power that would not otherwise be currently available. Parallel computing is an essential aspect of modern CFD development, requiring much effort to successfully utilise in a code, but giving performance benefits as a result.

The rapidly changing parallel computing marketplace has contributed to the fact that, until recently, no particular method of parallel computing has managed to gain mainstream acceptance. The vast number of choices available to the consumer in the past has now settled into what is basically a choice between shared memory and distributed memory computers.

The review in this section discussed the benefits of explicit parallelisation. In Chapter 5 the implementation of the *explicit everything* approach to parallelisation will be explored. This approach is thought to be most suitable to compressible CFD as it allows greater control and portability than implicit methods.

Implicit parallelisation has benefits in ease of programming; however, compiler technology is not yet at the stage where portable, efficient code is reliably generated. Explicit parallelisation, although requiring significantly more work in development, can provide efficient code that can be run on different types of platforms.

MPI is the standard for message passing libraries on distributed memory computers and OpenMP is the standard for parallel libraries on shared memory computers. OpenMP relies on having specialised shared memory hardware to run. Message passing libraries can be used on almost all types of computers, including the emerging class of cheap and powerful parallel computers based on workstation clusters. These methods work with standard programming languages, such as C or Fortran. This makes them suitable to developing parallel versions of CFD codes that have already been developed.

CHAPTER 5

Implementation of Parallel Computer Codes

Chapter 4 provided an overview of parallel computing, an historical perspective and an outline of the classifications applied to parallel computing hardware and software. The benefits of explicit parallelism, using parallel libraries that are built on top of traditional languages, such as C and Fortran, were discussed. These librares define, or prompt the compiler as to, how the parallelism is achieved. Benefits in using these libraries are based on improvements in the portability and predictability of the resulting code, and the convenience in continuing to use C and Fortran in parallel code development. It was decided through this review, and given the computers that are available, to use both OpenMP and MPI in the development of the CFD codes discussed in this thesis.

This chapter will investigate the implementation of parallelism in actual Computational Fluid Dynamics (CFD) codes. Before the discussion on the CFD codes, a simple application will be described to illustrate the overall structure of a parallel program in each of the parallel methods. The parallel codes were implemented using three methods: OpenMP is a compiler directive based approach using the shared memory model, MPI is based on the distributed memory model with message passing constructs and BSP is based on message passing constructs, like MPI, but imposes a more restrictive structure to the parallel code. These three methods make up the vast majority of the code use for parallel programming on distributed and shared memory parallel computers. Vector computers generally use platform-specific automatic vectorising languages and will not be discussed in this chapter.

OpenMP [168] is the newly established programming standard for shared memory computers and the Message Passing Interface (MPI) [161] is the well-established standard for message passing on distributed memory computers. The success of MPI for distributed memory computers is largely seen to be due to the fact that for the first time a great priority was given to establishing it as a standard right from the outset [87]. This success prompted the developers of shared memory computers

to establish their own standard in OpenMP. The push for firm standards in parallel computing is important because it promotes portability. In contrast to these two programming models we will also investigate the Bulk Synchronous Parallel (BSP) [171] method, which is seeking to establish itself as a standard by providing advantages over the other two methods.

The fine resolution mesh simulations of the Drummond Tunnel, performed in Chapter 7, require one month of CPU time on the APAC National Facility. If OpenMP directives were used to divide this solution amongst four processors, then this solution time could be reduced to one week. The architecture of the APAC National Facility, being made up of four processor SMP boxes, means that the shared memory approach of OpenMP cannot be used between these boxes. The message passing approach of MPI can be used between these boxes and a solution on 30 processors, using MPI, would take around one day. This demonstrates the importance in using parallel computing in obtaining the large simulations in this thesis.

The implementation of a simple code, used for calculating π based on a random sampling of points around a circle, is used to provide an example of how parallelism is achieved with the three methods. The implementation of parallelism of two CFD codes, based on the numerical techniques that were described Chapter 3, are then described. The implementation of the SPH code is described in detail using OpenMP, MPI and BSP. Then the implementation of MB_CNS, using OpenMP and MPI, is described. Following the discussion of the implementation, the performance and efficiency of parallel versions of MB_CNS will be discussed. The performance and efficiency of the parallel versions of the SPH code are discussed in Appendix A.

5.1 Parallelisation

Whilst parallel speedup is achieved by dividing up the computational work, this sharing of work has its penalties or overheads: information must be communicated between the processes and, as well as this, the processes must be regularly synchronised and managed. These overheads represent work that would not otherwise have to be done in a sequential code. The goal of parallel computing is the division of work amongst multiple processors without incurring significant overheads brought about by that division.

The process of dividing the work for parallel execution in CFD applications is known as *domain decomposition* as it is the fluid domain itself that is distributed. There are different methods, of varying complexity and resulting efficiency, used to achieve this division. Domain decomposition in MPI and BSP requires that the developer specify particles to be assigned to processes from the array as a function of the process number in the rank. This process is automatic in OpenMP and allows dynamic allocation of the work to the threads.

Data dependency analysis must be performed before parallelisation of an algorithm is attempted. By this, it must be ensured that processes, or threads, are not reading data that is being written to memory by another process, or thread, in the same block of computation. This would result in a race between the two as to whether the new data is written first or the old data is read first. *Race condition* is a term used to describe this situation. OpenMP performs a data dependency analysis before allowing a segment of code to be executed in parallel; however, in MPI and BSP this responsibility is with the developer.

As a parallel program proceeds, the point will be reached at which processes will require information from other processes. This requires the results to be communicated between the two. This can be achieved in different ways depending on the parallel hardware and software being used. Communication may be performed concurrently with computation through the use of communication buffers; however, for the most part they are separate and communication forms a distinct break in computation. This is particularly the case with explicit CFD codes which work through a sequence of time steps.

Communication usually makes up a significant proportion of the extra work incurred through parallelisation. Communication time is made up of two parts: latency and the actual data transfer time. Latency is the delay in which the link between the two sides of the data transfer is established. The smaller the amount of data being transferred, the more significant the effect of latency; this makes it usually more efficient to transfer larger blocks of data when communication is performed.

The interprocess communication latency on the SGI Origin 2000 is 15 μs, and the communication bandwidth is 80 MB/s. Interconnection networks used on Beowulf clusters vary; however, latencies are usually higher, being from 19 μs and 30 μs, for a switched fast ethernet based interconnection [199], and bandwidths are usually smaller.

Synchronisation, usually tightly coupled with communication, involves ensuring that the units are all up to the same stage in the algorithm before allowing any of them to continue, at any point where this is critical. Synchronisation can be an expensive operation when many processors are used.

Distributed memory inter-process communication may be either blocking or non-blocking. Blocking communication means that processors that complete the communication routines first are stopped from proceeding with new computation until all processors have reached the same stage. MPI collective communications are blocking, forcing all processes to suspend until all communication is complete. This means that after a collective communication call, an `MPI_Barrier` call is not necessary and would only result in a second synchronization, thus doubling the overhead. BSP collective communications are non-blocking and explicit synchronisation is required. Both procedures achieve the same result.

The granularity of parallelism refers to the relative size of segments that the code can be broken into to be run in parallel. The finer grained the parallelism is, the more regularly communication and synchronisation, bounding regions of computation, is required. CFD applications are generally formulated to allow coarse grained parallelism. This is often more efficient because it reduces the significance of system latencies.

Load balancing ensures that each process spends the same amount of time doing the work assigned to it. The issue of load balancing may be more complicated than simply sharing equal numbers of computational cells or points between processes. Where processors of differing speed are used, load balancing may mean giving more work to faster processors accordingly. Dynamic allocation of work in OpenMP programs can be conveniently used for load balancing.

When running an algorithm in parallel, sections of the algorithm may be fundamentally not parallelisable. This means that while most of the algorithm is processed in parallel, some parts of the code can only be solved sequentially. These regions have a serious impact on parallel efficiency. For example, in the SPH code, the sequential work that cannot be parallelised includes stacking any new particles in the array as they are created; this action cannot be performed independently with different particles on the same array. All processes, or threads, perform the work

in sequential regions redundantly, which may seem wasteful, but in fact increases efficiency. When a parallel program is running, any processors that are not working on the problem, for instance during a sequential region, are not released to the system and so are still charged to the program (if CPU usage time is recorded for users on the system). If only one process worked on the sequential region its results would need to be communicated to the other processes introducing additional communication overhead without saving any processor time.

The same basic principle of parallelism is common to OpenMP, MPI and BSP; however, the three methods are quite different in respect of their details. A simple example of a naturally parallel program will be used in the following sections to illustrate how parallelism is achieved with each method.

MPI and BSP use the distributed memory model and OpenMP uses the shared memory model. The implementation using both approaches will be described later, using the simple parallel model as a basis. With the shared memory model, all processors access the same global memory space, and so, in the case of a CFD code, there is a single copy of the whole flow field stored in this memory. In the distributed memory model each process stores data in its own memory space making it more like the simple parallel model, which is described later, than the shared memory model. If processors require updated data that is being stored by another process then this information must be transferred to it by message passing. With message passing, discrete blocks of information are requested and then sent and received explicitly by the two participating processors.

5.1.1 A Simple Application

A simple Monte Carlo simulation to estimate the value of π will be used to demonstrate the parallelisation of an algorithm using the approaches of OpenMP, MPI and BSP. The algorithm randomly generates points within a unit square where the fraction of points that lie within a quarter circle of unit radius can be used to estimate π. The area of the quarter circle is $\pi/4$ and the area of the square is 1. Thus, with a random distribution of points, the fraction of points that lie within the quarter circle would be approximately $\frac{\pi}{4}$. The sequential code, written in C, for this algorithm is shown below and features a single main loop in which points are generated and checked to see if they lie within the circle. This section of the program is naturally parallel as different processes, or threads, can sample points independently and the results can be combined at the end. The sequential implementation of the π calculating program is shown in Figure 5.1.

```
/* file:picalc.c */
#include <stdio.h>
#include <stdlib.h>

int main() {
        float pi_estimate;
        float x, y, r_squared;
        long i, total_points=10000, inside_count = 0;

  for (i=1; i<=total_points; i++) {
    x = rand()/RAND_MAX;
    y = rand()/RAND_MAX;
    r_squared = x*x+y*y;
    if (r_squared < 1.0) inside_count++;
  }

  pi_estimate = 4.0*(double)inside_count/(double)total_points;
  printf("Estimate of PI = %g for total points = %ld\n", pi_estimate, total_points);

  return 0;
}
```

Figure 5.1: Sequential implementation of the π calculating program.

Shared Memory (OpenMP)

OpenMP is a specification for a set of compiler directives, library routines and environment variables that can be used to specify shared memory parallelism in Fortran and C programs. Standardisation efforts in shared memory parallel programming are focusing on the development of OpenMP. Before OpenMP, each vendor produced its own set of constructs for shared memory parallel computing. For example, on SGI computers such as the Origin series of computers, OpenMP has replaced SGI-PowerC. A previous attempt at a standard specification for shared memory, ANSI X3H5 was never formally adopted. OpenMP is far more advanced than its predecessor on SGI computers, PowerC, being much more than a loop paralleliser. OpenMP allows the parallelisation of general regions of code, the nesting of parallel regions inside one another and more control over the details of the parallel execution.

As the SGI-OpenMP library used on the SGI Origin 2000 and 3400 is tuned to these systems, good performance would be expected. On these computers, underlying hardware and software takes care of any data transfer between threads transparently giving the impression, to the programmer, that they are using a shared memory computer. The level of usage of OpenMP is likely to increase with the use of CC-NUMA type DSM computers as it is very effective on this type of computer.

At present OpenMP does not run on distributed memory computers because it is based on using a shared memory space. Such an implementation would probably have to be built on top of MPI and would therefore incur significant performance penalties.

OpenMP directives are prompts given to the compiler; they do not explicitly

specify parallelisation. Compiler directives intended for OpenMP will not be understood by other interpreters and, as pragmas in the C programming language, will be ignored when not in use. With an OpenMP capable compiler a tag, such as `-omp` on the SGI Origin 2000, will cause these pragmas to be read and acted upon. Regions of the code that the programmer wants to run in parallel are bounded by the directive:

```
#pragma omp parallel
{
...
}
```

Within this parallel region, the compiler can be prompted that the main loop of the π calculating program can be run in parallel. This is demonstrated in the OpenMP parallel implementation of the π calculating program, shown in Figure 5.2.

```
/* file:picalc_omp.c */
#include <stdio.h>
#include <stdlib.h>

int main() {
        double pi_estimate, x, y, r_squared;
        int i, total_points=10000, inside_count = 0;

#pragma omp parallel
{
  srand(omp_get_thread_num());
  #pragma omp for private(x, y, r_squared) reduction(+:inside_count)
  for (i=1; i<=total_points; i++) {
    x = (double)rand()/RAND_MAX;
    y = (double)rand()/RAND_MAX;
    r_squared = x*x+y*y;
    if (r_squared < 1.0) inside_count++;
  }
}
  pi_estimate = 4.0*(double)inside_count/(double)total_points;
  printf("Estimate of PI = %g for total points = %d\n",
          pi_estimate, total_points);

  return 0;
}
```

Figure 5.2: OpenMP implementation of the π calculating program.

This example shows that the structure of the code is not changed by the addition of OpenMP parallelism for this simple, naturally parallel program. The two pragmas have prompted the compiler that the calculations in the following loop can be run in parallel. The number of threads used in an OpenMP is specified as the environment variable `OMP_NUM_THREADS` and changing this number does not require the code to be recompiled. OpenMP parallel code for the pi example is shown. The OpenMP pragma statement prompts the compiler as to the locations of potentially parallel regions. In this case the loop is divided amongst multiple threads and the `reduction` clause is used to sum each of the threads contribution to the total `inside_count`.

The call to `srand()` seeds the random number generator, using the thread number, so that each thread is working with a different set of random numbers; the threads using different sets of random numbers is essential for the parallel computation.

This code would be compiled and run, using four processors, on the SGI Origin series computers with the following commands:

```
cc -O3 -mp picalc_omp.c -o picalc_omp.exe -lm
setenv OMP_NUM_THREADS 4
./picalc_omp.exe
```

Message Passing Interface (MPI)

The Message Passing Interface (MPI) was proposed in 1993 by a committee, formed at the Supercomputing '92 conference, known as the MPI forum. MPI was intended to become the standard specification for communication protocols in MIMD distributed memory parallel computers [3]. This has largely come to be a reality with almost all modern supercomputers having MPI libraries available.

The basis of message passing is that all processes store only a local memory, but are able to communicate with the other processes by sending and receiving messages. One of the defining features of message passing is that the sending and receiving of messages are separate operations [87].

One of the primary reasons for the success of MPI is that, from the very beginning, computer manufacturers, Universities, government laboratories and industry have all contributed to the standard. MPI is largely seen to have taken over from PVM and has used features and notation from that standard to build upon. A lot of the development of MPI has been with workstation clusters in mind, as discussed in Chapter 4, and with future trends in supercomputing almost certainly being focused towards these types of architectures, the use of MPI seems set to increase.

The MPI libraries used in this paper complied with version 1.2 of the standard. Three of the most widely used libraries, LAM-MPI [236], MPICH [86] and MPI/Pro [162] were used in the tests on the beowulf cluster, and the SGI Message Passing Toolkit (MPT) was used on the Origin 2000. The MPT is a library based on the standard that is specially tuned for the Origin 2000 architecture. The MPICH and MPI/Pro libraries originated from Argonne National Laboratory and the University of Chicago, with MPI/Pro only differing in that it is marketed on a commercial basis and comes with technical support. The two libraries are specially targeted for workstation clusters, and there was no discernable difference in the performance of the two libraries in the tests, as would be expected.

The complexity of MPI can range from simple send and receive commands to collective communication procedures involving structures of data. MPI can be described as being as large as the user decides, where an entire algorithm can be written using the basic six commands, out of an available 125, incorporating all of the functionality available in the standard.

MPI continues to be developed with the MPI-2 standard, whose definition was completed in 1997. The new standard includes significant extensions to the MPI-1 programming model and the introduction of new functions, such as one sided communication, and the replacement of some functions with more efficient implementations [215].

A program using MPI must include the MPI header file `mpi.h`. In MPI the number of processes to be used is specified from the command line at execution time. The two arguments must be added to the `main()` function to account for the two command line arguments that specify the number of processors to `mpirun`:

```
main(int argc, char **argv) {
```

Processes cannot be created and destroyed during execution and will be used throughout, regardless of whether they are doing useful work; however, these processes cannot be given any MPI specific commands until MPI is initialized, with:

```
MPI_Init(&argc, &argv);
```

MPI communications are conducted within groups. The most broad group for MPI communication is `MPI_COMM_WORLD`, which will be used for all communications due to the simple nature of our communication needs. The number of processes, P_{total}, specified to mpirun is constant throughout the solution procedure.

The total number of processes used is stored in the local variable `num_proc` and each processes is given a identification number (from 0 to P_{total}`-1`), stored locally as `mpi_pid`.

```
MPI_Comm_size(MPI_COMM_WORLD, &num_proc);
MPI_Comm_rank(MPI_COMM_WORLD, &mpi_pid);
```

The MPI parallel implementation of the π calculating program is shown in Figure 5.3. This implementation is more complicated than would normally be needed in that each process simply could work through indices 1 to `total_points/p`; however, this implementation does assign points to be calculated in the same way as the SPH code described in Section 5.2.1, and so is a useful example. Specifically, this means that points are treated as if they come from the same list of indices numbered from 1 to `total_points`.

```
/* file:picalc_mpi.c */
#include <stdio.h>
#include <stdlib.h>
#include "mpi.h"

int main(int argc, char **argv) {
        double pi_estimate, x, y, r_squared;
        int total_points = 10000, inside_count = 0;
        int i, num_proc, mpi_pid, start, end;

  setbuf(stdout, (char *)0);

  MPI_Init(&argc, &argv);
  MPI_Comm_size(MPI_COMM_WORLD, &num_proc);
  MPI_Comm_rank(MPI_COMM_WORLD, &mpi_pid);

  srand(mpi_pid);
  start = mpi_pid*(int)((double)total_points/(double)num_proc);
  if (mpi_pid == num_proc-1) end = total_points;
  else end = (mpi_pid+1)*(int)((double)total_points/(double)num_proc);
  printf("... process %d has %d to %d\n", mpi_pid, start, end);

  for (i=start; i<=end; i++) {
    x = rand()/RAND_MAX;
    y = rand()/RAND_MAX;
    r_squared = x*x+y*y;
    if (r_squared < 1.0) inside_count++;
  }

  MPI_Reduce(&inside_count, &inside_count, 1, MPI_INT, MPI_SUM, 0, MPI_COMM_WORLD);
  MPI_Finalize();

  pi_estimate = 4.0*(double)inside_count/(double)total_points;
  printf("estimate of PI = %g for total points = %d\n",
         pi_estimate, total_points);

  return 0;
}
```

Figure 5.3: MPI implementation of the π calculating program.

The call to `MPI_Init` initialises the MPI parallelism taking the number of processors to use as input from the command line. `MPI_Comm_size` and `MPI_Comm_rank` stores how many processes are active and which number they have been individually assigned. For simplicity the number of points to be tested is truncated, making it divisible by the number of processes with no residual. In MPI, division of the problem is done manually in MPI and is achieved by specifying the indices that will be used in the main `for` loop as a function of `mpi_pid` and `num_proc`. The variables `start` and `end` are stored separately by each process, and then each process independently runs on its fraction of the points and stores the number of points that lie within the quarter circle.

The reduction operation `MPI_Reduce` takes the values stored at the address `&inside_count` from each process and performs the MPI operation `MPI_SUM`, which sums all of the values stored by the threads. The result is stored at the address of `inside_count` on the root process, process zero. The one refers to the number of

data values on each thread that are being summed. `MPI_INT` is the data type that is being summed, which is equivalent to type `INT`. `MPI_COMM_WORLD` is the communication group being used. The call to `srand()` seeds the random number generator for each process differently, using the process number.

This code would be compiled and run, using four processors, on the SGI Origin series computers with the following commands:

```
cc -O3 picalc_mpi.c -o picalc_mpi.exe -lm -lmpi
mpirun -np 4 ./picalc_mpi.exe
```

Bulk Synchronous Parallel (BSP)

The Bulk Synchronous Parallel (BSP) model of parallel computing was first proposed in 1990 by Valiant [237]. Since then the model has been implemented by a research group at Oxford University [171] producing BSPlib which is a library of BSP communication and synchronisation primitives. These primitives are callable from both Fortran and C and the library includes performance analysis, benchmarking and debugging tools. The aim of BSP is to produce a parallel library that is architecture independent, provides scalable parallel performance and yet is conceptually simple. These objectives have largely been achieved in BSPlib; however, results were only obtained for the Origin 2000 since the installation of the library on the Beowulf cluster failed.

Within a superstep each process performs its independent computations, followed by a global computation phase and then a barrier synchronisation. In this way BSP imposes a more strict, but simple underlying structure on the parallel code, allowing parallelisation to be achieved with less complication and effort, and greatly aiding in the analysis of the performance of the code. The superstep-based structure of BSP corresponds well with the time step structure of explicit CFD codes and so provides a useful model.

Unlike MPI, BSP threads are created at the time that BSP is initialised, using `bsp_begin()`, and therefore any commands before then are only executed by the first thread, which is effectively sequential code. BSP code must include the BSP header file `bsp.h`. The BSP parallel implementation of the π calculating program is shown in Figure 5.4.

A program using BSP can accept the number of processors to be used as input from the command line, as with MPI. BSP does not feature a dedicated `reduce` function, as was used in the MPI version of the code. For this reason `malloc` is used to create an array with an element for each process running; when processes are finished doing their part of the calculation, their local copy of `inside_count` is sent to the master process and put in an element of this array. The result is then summed by the master process to obtain the answer.

The memory space used by the `receive_buffer` array is written to by the other processes using Direct Remote Memory Access (DRMA) and must be made available to the other processes by using `bsp_push_reg`; it is released after the processes have written to it using `bsp_pop_reg`. Again, the call to `srand()` seeds the random number generator for each process differently, using the process number.

```
/* file:picalc_bsp.c */
#include <stdio.h>
#include <stdlib.h>
#include <bsp.h>

int main(int argc, char **argv) {
double pi_estimate, x, y, r_squared;
int i, num_proc, start, end;
int total_points = 10000, inside_count = 0, ;
        int *receive_buffer;

  num_proc = atoi(argv[2]);
  receive_buffer = malloc(num_proc*sizeof(int));
  bsp_begin(num_proc);
  bsp_push_reg(receive_buffer, num_proc*sizeof(int));

  srand(bsp_pid());
  start = bsp_pid()*(int)((double)total_points/(double)num_proc);
  if (bsp_pid() == num_proc-1)
     end = total_points;
  else
     end = (bsp_pid()+1)*(int)((double)total_points/(double)num_proc);
  printf("... process %d has %d to %d\n", mpi_pid, start, end);

  for (i=start; i<=end; i++) {
    x = rand()/RAND_MAX;
    y = rand()/RAND_MAX;
    r_squared = x*x+y*y;
    if (r_squared < 1.0) inside_count++;
  }

  bsp_put(0, &inside_count, receive_buffer, bsp_pid()*sizeof(int), sizeof(int));
  bsp_sync();

  bsp_pop_reg(receive_buffer);
  bsp_end();

  for (i=1; i<num_proc; i++) inside_count += receive_buffer[i];
  pi_estimate = 4.0*(double)inside_count/(double)total_points;
  printf("estimate of PI = %g for total points = %d\n",
          pi_estimate, total_points);

  return 0;
}
```

Figure 5.4: BSP implementation of the π calculating program.

This code would be compiled and run, using four processors, on the SGI Origin series computers by the following commands:

```
bspcc -flibrary-level 2 -O3 picalc_bsp.c -o picalc_bsp.exe -lm
./picalc_bsp.exe -np 4
```

5.2 Parallel Implementation

This section will describe the implementation of parallelism in the SPH code and in MB_CNS. The SPH technique exhibits a natural parallelism and MB_CNS utilises a multi-block parallelism. The parallelism in the SPH code will be described in more detail.

The accepted notation for referring to each of the separate instances of an SPMD program operating in parallel is a *thread* when using a shared memory space, and a *process* when using a distributed memory space. For this reason, OpenMP threads and MPI (or BSP) processes will be referred to. These are distinct from the term *processors* which refers to the hardware that threads and processes run on. In this chapter, it can be assumed that each thread or process is run on its own processor unless explicitly stated.

5.2.1 Parallel SPH

The Smoothed Particle Hydrodynamics (SPH) technique was described in Section 3.2.1. The SPH technique is naturally parallel as the solution procedure can be divided up in a straight forward manner. The majority of the work in the SPH method is in calculating particle interactions and these interactions are independent of one another. This natural parallelism is demonstrated in Figure 5.5 where the effect of particle A on B can be calculated independently of the interaction of C and D.

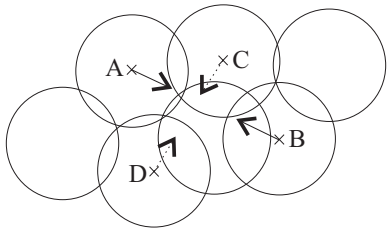

Figure 5.5: Using kernel interpolation, the calculations for any particles A with B are independent of C with D.

The class of parallelism implemented in most CFD codes is described using Flynn taxonomy [70] as Single-Program Multiple-Data (SPMD) parallelism. With the SPMD model of parallel computation each process works on a different segment of the same data set; in the case of CFD the same flow domain. As seen in the simple π calculation program in Section 5.1.1, the distribution of work is done automatically in OpenMP, but must be specified explicitly with MPI and BSP. OpenMP gives a number of options in how the division is done to allow for load balancing.

The most simple method of dividing up the work in an SPH code is to arbitrarily assign $1/p$ of the particles, where p is the number of processes being used, to each process. This will be referred to as the *simple* parallel model. This model is the basis for the shared memory model and is one of the two main choices for distributed memory programs.

Figure 5.6 shows the division of work in the simple parallel model for the SPH method. A simulation using four processes is shown. Each process stores its own copy of the array of particles which may, or may not, be accessible by the other processes. During a time step each process updates the properties of the particles assigned to it, with each process being assigned a different segment of the data. The segment assigned to each process is shown as shaded in the Figure. This is a *logical* partitioning of the data set according to how the array is stored by the program. At the end of a time step, each process has updated information for its section of the

array (at time t) and out of date information for the other sections (at time $t - \delta t$).

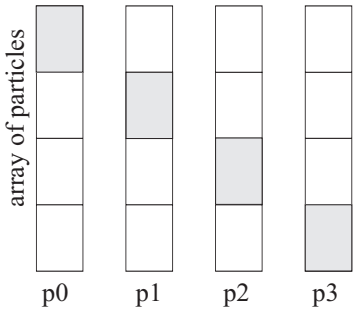

Figure 5.6: *Logical* decomposition for the *simple* parallel model for SPH programs.

In order to carry out new calculations using this simple parallel model, each process must have a complete copy of the particle array. The calculation of properties for any particle may involve particles from anywhere in the array depending on their physical locations in the simulation; this is not related to their logical position in the particle array. This means that, once each process is finished updating its segment of the data, the new data must be shared amongst the other processes. This data transfer is shown in Figure 5.7 for the particles associated with process zero.

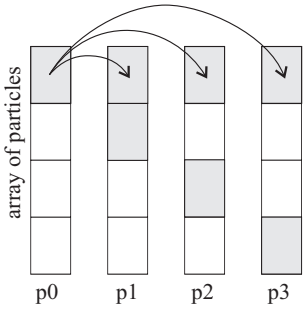

Figure 5.7: Communication for the *simple* parallel model for SPH programs.

It is an important feature of the SPH method, given its explicit nature, that the calculation of any property can not result in race condition. This is because for any calculation being performed the same variable is never being read as is being calculated; for example, even the calculation of density, using kernel interpolation, only requires particle positions to proceed.

The OpenMP version of the SPH code achieves parallelism through all threads using a shared memory space. Shared memory parallelism with the SPH method only differs from the simple domain parallel method described in Figure 5.6 in the way that data is stored and shared. Since all processors access the same global

memory space there is only the need for one copy of the particle array; the simple model had a copy of the array for each processor.

The basic *shared memory* model is shown in Figure 5.8, where process 0 is reading the properties of particles from anywhere in the data structure and writing updated results to the section of the global array assigned to it. The rest of the array is updated by the other processes concurrently.

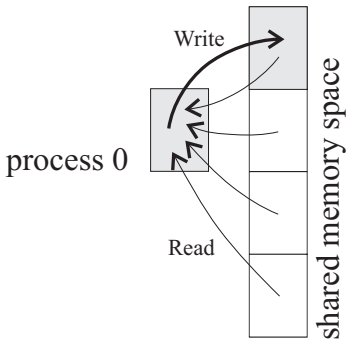

Figure 5.8: Shared memory parallelism for process 0.

The whole particle array must be up to date before any threads can start the new time step since, as with the simple model, updating a particle may require information from anywhere in the array. With the shared memory model there is still only copy of the data; this means that, unlike the simple parallel model, it does not suffer from the storage problems with multiple copies of the array. With the shared memory space, communication occurs concurrently with computation since updated results are written to the globally accessible array; however, with SPH, race conditon will not be incurred because no variable is ever read from and written to the particle array in the same region of computation. The shared memory model, although being very effective for limited numbers of processors, is limited in scalability by the need for all processors to practically be able to access the same memory space.

The MPI and BSP versions of the parallel SPH code use the *distributed memory* model. In this model, each process stores data in its own memory space which is not directly accessible to other processes. This makes it more like the simple parallel model described in Figure 5.6 than the shared memory model. All processors must receive all updated particle information before the start of a new time step since, as with the simple model, updating a particle may require information from any other particle. This results in a large amount of data storage and a large amount of communication that must be performed to keep this data up to date. This method is, however, relatively simple to implement and analyse. If processors require

updated data that is being stored by another process then this information must be transferred to it by message passing. This is based on the communication style shown for the simple parallel model in Figure 5.7. With message passing, discrete blocks of information are requested and then sent and received explicitly by the two participating processors. Group communication routines can be used to perform specific types of communications between assigned groups of processes.

With a code based on the Lagrangian description of fluid motion, such as SPH, the computational elements enter and leave the flow domain during each time step. This means that particle array must be re-partitioned at the start of every time step regardless of the model of parallelism used.

Load balancing is particularly an issue in SPH simulations. Particles in regions of higher density have more neighbours and, therefore, each particle require more computational work. If equal numbers of particles are given to each process then those working in regions of higher density will have more work to do and, therefore, will take longer to do it. This is a problem in simulations involving discontinuities in density, as with shock tube simulations. OpenMP allows dynamic allocation of work to threads to account for this type of effect. An answer to load balancing, also applicable to MPI and BSP, is to randomly assign particles to processes so that each would have a roughly the same number of particles from any high density regions. This would lead to problems with memory access and cache utilisation, potentially degrading overall performance.

The simple parallel model, although providing a convenient structure, is inefficient for the purpose of large simulations. In terms of storage, the entire array of particles must be stored on each process; this is 972Mb of memory that must be regularly accessed for a one million particle simulation. One important aspect of parallelisation of a CFD code is the reduction in memory space that each process must work within. In terms of communication, transferring the entire array of particles to every process is inefficient and, as well as this, the amount of data that must be transferred increases proportionally with the number of processes.

Multi-block parallelism, which is described with MB_CNS in Section 5.2.2, is made possible in an SPH code through the use of compact support in the interpolating kernel. The blocks that divide the domain could be made up of the sorting cells used to find neighbouring particles, as was described in Section 3.2.1. An implementation based on multi-block parallelism would result in significant savings in memory usage (by not having to store the whole particle array in each process). Any savings in communication time would likely be offset by a significant amount of extra work in accounting for the continuous movement of particles between the blocks.

Shared Memory (OpenMP)

The structure of the OpenMP parallel version of the SPH code is, for the most part, the same as the sequential version. The difference lies in that when a particular task is performed, such as the calculation of density using kernel interpolation, multiple threads are spawned which update information for a section of particles assigned to them in the shared memory space. For these sections of the code, the compiler arranges multiple threads to be used in the calculations. There are sections of the code that are not executed in parallel, the most significant of which is assigning particles to the boxes used for sorting the particles to find neighbours.

Particle properties are updated by `for` loops, executed in parallel, which cycle through the list of particles. OpenMP spawns the new threads and automatically distributes the particles amongst them. In the following example the compiler is prompted by the `omp for` pragma that the following `for` loop should be run in parallel:

```
#pragma omp parallel{
#pragma omp for private(i, x_ij, y_ij) shared(ptc, total_ptc, h)
  for (i=0; i<total_ptc; i++) {
    ...
  }
}
```

This loop would be used in the code to update the properties; in the case of step 4 in the pseudo code, density is updated by taking as input each particles position and the smoothing length, h. These properties can be evaluated independently, and, therefore, in parallel. The loop will be run in parallel with each thread using its own, `private`, copy of the loop index, `i`, and the relative positions of the two particles, `x_ij` and `y_ij`. The same copy of the particle array, `ptc`, the total number of particles, `total_ptc`, and the smoothing length, `h`, are used by all threads; these are specified as `shared` variables in the pragma. OpenMP relies on shared memory to achieve communication since the particle array is `shared`, the threads are all reading from and, later, writing to the same array.

At the end of a loop, the parallel region ends and there is an implicit synchronisation of the threads. This is important as no threads can move on to calculations that require data that has not been updated by another thread. This process is repeated for each new property calculation and are repeated for each time step in the simulation. The `for` loop, if it were being run sequentially, would run from 0 to `total_ptc`. This structure is important as, if the pragmas were ignored, the sequential loop would still work correctly.

The way that elements from the loop, particles in our case, are assigned to the threads can be specified by the developer. These options include dynamic scheduling in which more work is assigned to threads as they complete work.

The performance of OpenMP will be used as the benchmark for parallel performance of the parallel SPH code as it is aimed at being easy to implement and still perform well on the Origin 2000.

Message Passing Interface (MPI)

The MPI parallel version of the SPH code, although aiming for the same result as the OpenMP version, must include more of the detail of how the parallelism is achieved. The pseudo code for the MPI version is shown in Figure 5.9. The structure of the pseudo code is not changed from the sequential version, however, the blocking group communications at steps $5a$ and $8a$ are added. The densities, pressures and sound speeds calculated at 4 and 5 are required by the calculations at 6 and 7 and so communication is required at $5a$. When the particle's properties are integrated in time, at 8, this information must be communicated to the other processes, at $8a$, before the next time step can commence. With the simple parallel model used, particle information may be required from any position in the particle array and so each process must update the information stored in the arrays on other processors. The broadcast performed by each process is a blocking communication routine. This eliminates the need for an explicit barrier synchronisation at these points.

```
      1.  assign particles to flow domain according to initial density
      2.  assign initial properties to particles
   -> 3.  assign particles to cells
  |   4.  calculate densities at particle positions
  |   5.  calculate particle pressures and local sound speeds
  |   5a.  blocking group communication to update particle array
  |   6.  calculate particle accelerations
  |   7.  calculate rate of change of internal energy
  |   8.  integrate particle properties forward through time step
  |   8a.  blocking group communication to update particle array
   -- 9.  check if solution time has been reached
```

Figure 5.9: Pseudo code for the MPI version of the parallel SPH method.

On each time step, domain decomposition is performed by distributing the particles amongst the processes. In keeping with the simple parallel model, each process must have access to the whole particle array. The particles are distributed for computation by specifying the indices of the particles that bound the region to be calculated: `start` and `end`. This is done for each process as a function of `mpi_pid`. The number of remainder particles is also stored as `extra_particles`; these particles are handled by the last process. These extra particles being updated by the last

process will lead to a load imbalance; however, this is insignificant since the number of extra particles is at most `num_proc-1`.

```
start = mpi_pid * (int)(total_ptc/num_proc);
if (mpi_pid == num_proc-1) end = total_ptc;
else end = (mpi_pid+1) * (int)(total_ptc/num_proc);
extra_particles = total_ptc - num_proc * (int)(total_ptc/num_proc);
```

Once the computations have been performed and the data has been updated, the data must be shared; this occurs at at the end of both steps 5 and 8 in the pseudo code in Figure 5.9, at steps 5a and 8a. Each particle in the array stores 17 properties, each stored as double-precision floating point variables. Only some of these variables store the thermodynamic properties of the particle, the variables are used by the program, such as in cell sorting. For the communication at step 5a, only the density, pressure and sound speed have been updated and, for efficiency, only these variables need to be sent to the other processors.

Figure 5.10 shows the layout of the variables in the particle structure array. Each particle is a structure, consisting of the 17 variables, in an array. Precise access to the array is made possible by the use of pointers: the pointer `ptc` refers to the address of the entire particle array, `ptc[i]` refers to the address of particle `i` in the array and `&ptc[i].p` refers to the address of the property `p` of particle `i` in the array. MPI allows the definition of different types of variables to be declared, including

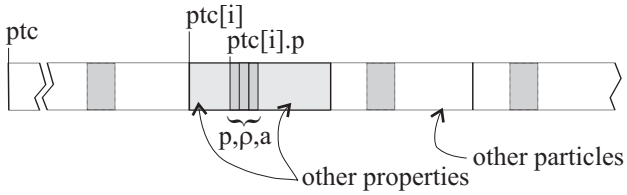

Figure 5.10: Layout of variables to be tranferred from the particle array

vectors, to be used in communication routines. The command `MPI_Vector` can be used to create a vector type that will be used to send only the necessary variables. In the case of step 5a, these are pressure, density and sound speed. As was shown in Figure 5.10, the density, pressure and sound speed variables are stored as three consecutive *doubles* within the particle structure. The vector type for `particles` particles that have been updated by the process, sending `send_variables` variables out of the `total_variables` variables that are stored for each particle (in this case the vector spans the 17 for each particle storing the 3 updated properties), all of type `MPI_DOUBLE` and assigning this vector to `MPI_Vector_1` is assigned with:

```
MPI_Type_vector(particles, send_variables, total_variables, MPI_DOUBLE, &MPI_Vector_1);
MPI_Type_commit(&MPI_Vector_1);
```

Each process can now broadcast this vector to the other processors, `process_recv`, using `MPI_Bcast`. For the correct variables to be sent in the vector type, the starting address used for the transfer should be the address of the first property to be sent, in the first particle to be sent, from the array; this is the pressure in the emphi'th particle: `&ptc[i*sendcount].p`. Since we are updating similar arrays on the processes, ihis memory address is the same on the sending and receiving processes. The vector type `MPI_Vector_1` encompases all of the particles to be sent and, therefore, only 1 copy of the vector is sent.

```
for (i=0; i<num_proc; i++)
  MPI_Bcast(&ptc[i*sendcount].p, 1, MPI_Vector_1, process_recv, MPI_COMM_WORLD);
```

Each processor performs this operation, and so we have an *all to all* group communication. The vector type can be freed using:

```
MPI_Type_free(&MPI_Vector_1);
```

The remainder particles, that did not fit into `total_ptc/num_proc`, have been updated by the last process. The updated properties are transferred to the other processes using a broadcast. Rather than define another type of vector to transfer these results, all of the properties are sent since it is a small amount of data.

```
MPI_Bcast(&ptc[num_proc*(int)(total_ptc/num_proc)], extra_particles*variables,
          MPI_DOUBLE, num_proc-1, MPI_COMM_WORLD);
```

Blocking communications are used, therefore barrier synchronisations are not required to follow the communications. This means that the program does not continue executing lines beyond the collective communication until it is guaranteed to be complete. The system resources being used by MPI are released and the MPI procedures are finalised using the command:

```
MPI_Finalize();
```

Bulk Synchronous Parallel (BSP)

The pseudo code for the BSP parallel version of SPH code is shown in Figure 5.11. It has the same structure as the MPI code, however, non-blocking group communications are used at steps $5a$ and $8a$, so explicit barrier synchronisations are now necessary; these are added at $5b$ and $8b$.

When the processes are initially spawned, a copy of the entire particle structure array, `ptc`, as well as the cell field structure and the geometry structure are given to each process. The region of the SPH code involving parallelism is initiated with the command:

```
  bsp_begin(num_proc);
```

```
        1.  assign particles to flow domain according to initial density
        2.  assign initial properties to particles
     -> 3.  assign particles to cells
     |  4.  calculate densities at particle positions
     |  5.  calculate particle pressures and local sound speeds
     |  5a.  non-blocking group communication to update particle array
     |  5b.  explicit synchronisation of all processes using bsp_sync()
     |       ------------ end of superstep ------------
     |  6.  calculate particle accelerations
     |  7.  calculate rate of change of internal energy
     |  8.  integrate particle properties forward through time step
     |  8a.  non-blocking group communication to update particle array
     |  8b.  explicit synchronisation of all processes using bsp_sync()
     |       ------------ end of superstep ------------
     -- 9.  check if solution time has been reached
```

Figure 5.11: Pseudo code for the BSP version of the parallel SPH method.

The memory area storing the particle structure array on each process is made globally visible for Direct Remote Memory Access (DRMA) transfers between threads. The memory occupied by the particle array, starting at the pointer to the array, `ptc`, and covering `total_ptc*17*sizeof(double)` bits, is registered as available to DRMA operations, using:

```
bsp_push_reg(ptc, total_ptc*17*sizeof(double));
```

As with MPI, domain decomposition occurs on each time step. This is necessary as the total number of particles in the simulation changes from each time step due to boundary conditions. Each process has access to the data from the entire set of particles, as with the simple parallel model. The particles are distributed for computation by specifying the indices of the bounding particles of the region to be calculated as a function of the process identification number. The communication routines in BSP are slightly more flexible in some regards and so storing the number of remainder particles is not necessary, as it is with MPI.

```
start = bsp_pid()*(int)(total_ptc/num_proc);
if (bsp_pid() == num_proc-1) end = total_ptc;
else end = (bsp_pid()+1)*(int)(total_ptc/num_proc);
```

The current BSP code does not use collective communication routines, rather performing send operations for each process to process communication. BSP using DRMA, as with MPI-2, allows for one sided communications and these are used in the BSP code, with the command, `bsp_put()`. One sided communication is where the send command can be issued to the sending process and the receiving process will automatically receive the data. The communication routine steps through the processes as sender and, for each sender, steps through the other processes as receivers. This is an *all to all* communication, but allows communication from a

process to itself to be excluded. The variable `ptc_mem_size` stores the number of bits that each process takes up in memory.

```
for (receiver=0; receiver<num_proc; receiver++) {
  if (bsp_pid() != receiver) {
    if (bsp_pid() == num_proc-1) {

      bsp_put(receiver, ptc[bsp_pid()*(int)(total_ptc/num_proc)],
              ptc, bsp_pid()*(int)(total_ptc/num_proc)*ptc_mem_size,
              (total_ptc+1-(bsp_pid()*(int)(total_ptc/num_proc)))*ptc_mem_size);
    } else {

        bsp_put(receiver, ptc[bsp_pid()*(int)(total_ptc/num_proc)],
                ptc, bsp_pid()*(int)(total_ptc/num_proc)*ptc_mem_size,
                (int)(total_ptc/num_proc)*ptc_mem_size);
    }
  }
}
```

This communication is shown in Figure 5.12, for process 1, having updated the light shaded segment of the data itself, and receiving DRMA transfers of data from the other processes to update the rest of its copy of the array, shown as dark shaded. Following this collective communication, it is necessary to perform a barrier syn-

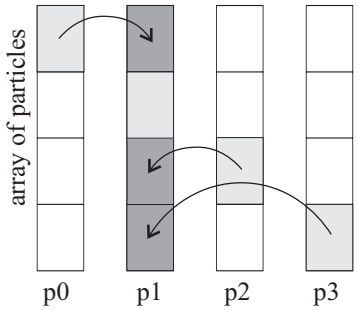

Figure 5.12: BSP DRMA data transfer for process 1 receiving.

chronisation to end the superstep. Communications are guaranteed to be completed by the end of the superstep, because of the barrier synchronisation. The next step may then proceed with a complete set of updated data. The synchronisation time is counted in the communication time for BSP communication regions as the actual process of communication continues after function call returns. This is known as non-blocking communication. The communication is only guaranteed to be complete once the call to the synchronisation routine, `bsp_sync()`,returns.

At the end of a simulation, the system resources being used by BSP are released and the procedures are finalised, as well as deregistering the particle structure array for BSP communications, with the commands:

```
bsp_pop_reg(ptc);
bsp_end();
```

5.2.2 Parallel MB_CNS

MB_CNS, which was described in Section 3.1.1, was originally developed as CNS4U, a single block Navier-Stokes solver [110]. This code was developed with the intention of extending it to a multi-block solver in order to enable the solution of flow fields in parallel. The equations being solved by MB_CNS are hyperbolic, and the time steps are chosen such that information can move only part way through a cell in any time step. This means that updated flow information only needs to be communicated directly at the block boundaries. The result, MB_CNS, is a multi-block solver in which parallel execution can be achieved by solving the flow in each block using a different processor [114], with the updated results at block boundaries transferred between the processors during every time step.

The OpenMP parallel version of MB_CNS was used predominantly for the simulation in this thesis. This restricted the number of processors used to four when running on the APAC National Facility. The MPI version of MB_CNS has been developed and its performance analysed as a part of this thesis. The resulting code, which allows the use of a larger number of processors on the APAC National Facility, will be needed by future studies especially those involving finite rate chemistry and radiation modelling, which are significantly more expensive than the simulations described in this thesis.

Shared Memory (OpenMP)

Shared memory parallelism was originally implemented in the code using SGI PowerC, which was the predecessor of OpenMP on the SGI Power Challenge computer. The shared memory implementation of MB_CNS is structured using the multi-block arrangement, with the flow in each block being solved separately. This makes the same structure compatible with the distributed memory implementation. Each block may be solved on a single thread, or the work done solving a block may be shared between threads.

By using the multi-block arrangement, each block uses the ghost cell boundaries. Since all threads are reading from the same memory space each block can simply read the boundary data from the bordering blocks. In the MB_CNS implementation, explicit sends and receives of data were required to perform this communication between the blocks.

Once the ghost cells have been updated. Large loops are specified to run in parallel using:

```
#pragma omp parallel for shared(G,bd) private(jb) schedule(runtime)
  for (jb = 0; jb < G.nblock; ++jb) {
```

```
    ...
}
```

Where the `for` loop steps through each of the blocks, `jb`, updating the flow properties from this block in the global flow data structure `G`. This process is repeated for the predictor and corrector steps of the time integrator, updating the ghost cells with the intermediate values between the steps. The self-contained packaging of the data for each block ensures that there are no memory conflicts.

Like in the OpenMP version of the SPH code, the only difference between the sequential version of MB_CNS and the OpenMP version are the few compiler directives that are added to the `main()` function to indicate which loops can be solved in parallel.

Message Passing Interface (MPI)

Due to the complexity of the communication routines used this version of MB_CNS, the actual coding will not be discussed in detail. Actions such as the initialisation of processes are similar to that in the discussion of the parallel SPH implementation.

The MPI version of MB_CNS solves each block in a separate memory space. The code is implemented in a one block per process arrangement. This was done to maximise parallel efficiency and to prevent the code from becoming too complex. As a result the code is quite restrictive in load balancing, since all blocks should be of about the same size. Parallel efficiency would suffer significantly if the size of blocks differs by a significant amount; however, static load balancing can be achieved by arranging several MPI processes to particular computational nodes.

The data arrays for each block are dimensioned such that there is a buffer region, two cells deep, around the active cells. The buffer region is required to be two cells deep by the reconstruction scheme used in the code. The active cells completely define the flow domain covered by the block and the buffer region contains ghost cells which are used to hold a copy of the flow information from adjacent blocks or to implement the boundary conditions.

For a boundary common to two blocks, the ghost cells in the buffer region of each block overlap the active cells of the adjacent block. The only interaction that occurs between blocks is the exchange of boundary data, prior to the reconstruction phase of each time step. The exchange of cell-average data along the block boundaries takes place as a direct copy from the active-cell of one block to the ghost-cell of the other block. Thus, the cells along the common boundary of each block must match in both number and position.

The code allows for any arrangement of connections to be made between blocks. The information on the connections between block boundaries is stored in a (global) connectivity array. For each boundary on each block, this array stores the block index of the adjacent block on each boundary, as well as the name of the connecting boundary on the adjacent block. To keep the code simple, the two-way nature of the exchange is explicitly stored in the connectivity array. Thus, if the East boundary of block 0 is connected to the West boundary of block 1, the array stores the information for that relation as part of the data for block 0 *and* it also stores the corresponding (inverted) information as part of the data for block 1.

Figure 5.13 shows the arrangement of data for the solution of block one. The rows of ghost cells used in the solution of block one have been copied from blocks two and three.

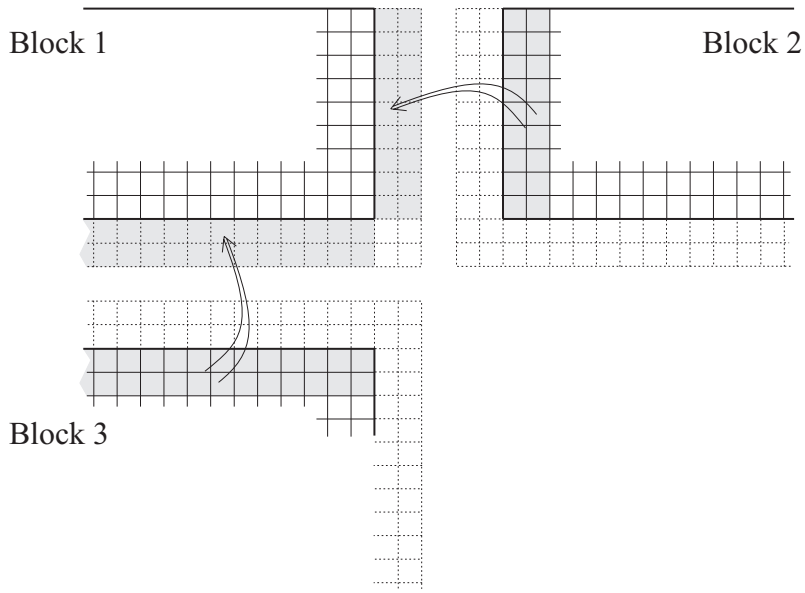

Figure 5.13: Block one with the rows of ghost cells on its east and south faces that are copied from the west and north boundaries of blocks two and three respectively.

The program is written as an (outer) time-stepping loop which does the calculation of each time step in a number of phases. At various times during a time step, for example at both the predictor and corrector steps of the time integrator, the boundary data is updated. Fluid properties required by neighbouring blocks must be sent and received explicitly between the processes. When an inter-block communication is required by the code, each block steps through its boundaries with bordering blocks. At each of these boundaries, the process posts non-blocking receives, using `MPI_Irecv()`, and blocking sends, using `MPI_Send()`. `MPI_Barrier()` is used at the end of each communication, thus providing the effect of a blocking communication over the whole routine. At present the number of calls to `MPI_Barrier()` are quite

conservative; further data dependency analysis will show whether all of the calls are necessary. Except for this block to block communication and the occasional checking of time step magnitudes, the rest of the calculation can be done independently for all blocks.

This model also has the potential to perform computational and communication concurrently [145]. As the properties of the cells away from block boundaries are being updated, the communication of data from the previous time step can be taking place at the boundaries. The communication of boundary conditions should must be completed in the time before they are required.

5.3 Parallel Performance

In this section, the performance of the OpenMP and MPI parallel implementations of MB_CNS will be analysed. The performance of the parallel implementations of the SPH code are included in Appendix A. The performance of the SPH code is described in more detail than MB_CNS, including discussion of the effect of parallelisation on memory access from each of the processors and the effect of data conflicts on the performance of the individual processor caches.

In this section the parallel performance of MB_CNS is examined on the Australian Partnership for Advanced Computing (APAC) National Facility. This computer is a HP (originally built by Compaq) Alphaserver CS, consisting of 127 nodes, each containing a four processor EV68 Symmetric Multi-Processor (SMP) box. The whole machine has a peak theoretical performance of over one teraflop (achieving 825 Gflops on Linpack). This architecture, with memory only being shared within the four processor SMP nodes, OpenMP cannot be scaled above four processors.

Many of the simulations used in this thesis were solved on the Queensland Parallel Supercomputing Foundation (QPSF) operated SGI Origin 3400. The performance of MB_CNS on this computer is not analysed as the heavy usage of the computer means that the performance is significantly influenced by other users and the performance results are not reliable for this reason.

5.3.1 Performance of Parallel MB_CNS

The solution procedure used in MB_CNS works on each cell in the flow field. This means that as the mesh is uniformly refined, the amount of work per step increases by N_x^2, where N_x is one of the dimensions of the mesh. The allowable time-step decreases in proportion to the cell size and, as a result, the total amount of computational work scales by an additional factor of N_x. The resulting solution time scales with N_x^3 for uniform refinement. The use of multiple processors in parallel ideally reduces the elapsed time by a factor of P, where P is the number of processors used. The logarithm of the elapsed time should scale as: logT = 3/2logN_t - log P, where N_t is the total number of cells in the simulation. The graphs shown in this section use logarithmic axes for both the solution size and the times so that the variation of solution and elapsed time plots as a straight line with a gradient of 3/2. The lines for elapsed times on different numbers of processors would be offset according to the number of processors used.

The OpenMP parallel version of MB_CNS, running on four processors is used for the majority of the large simulations in this thesis. Using four processors maintained an acceptable level of efficiency across the range of mesh sizes used in these simulations. Four processors was also the upper limit for OpenMP on the APAC National Facility.

The performance of the parallel versions of MB_CNS will be investigated using small diagnostic simulations, specifically aimed at analysing the parallel performance of the code. The parallel efficiencies are calculated by comparing the elapsed solution time in parallel with the elapsed sequential time divided by the number of processors.

Shared Memory (OpenMP)

Since OpenMP, running on four processors, is used for the large simulations in Chapter 7, the actual performance in these simulations is used in this analysis in conjunction with the performance in the small diagnostic solutions.

The solution times (in CPU hours) and the elapsed times required for the simulations of the complete Drummond Tunnel (to be discusses in Chapter 7) are shown in Table 5.1. The calculations were run on the APAC National Facility using OpenMP on four processors. The simulations were run to 10 ms of simulation time, which allowed for the simulation to run to beyond the end of the test flow duration. The solution times for the coarse, medium and fine meshes are shown, along with the number of cells in each mesh.

During the Drummond Tunnel simulations, MB_CNS achieved an average of 225,000 cell updates per second on the APAC National Facility and 115,000 cell

Table 5.1: Solution times for the Helium driving Nitrogen case simulations of the complete Drummond Tunnel on the APAC National Facility.

	Number of Cells in the simulation	Solution Time (CPU hours)	elapsed Time (hours)
Coarse Mesh	80,850	60.7	17.5
Medium mesh	181,890	237.9	64.2
Fine mesh	257,600	699.9	183.4

updates per second on the QPSF SGI Origin 3400. The speed on the Origin 3400, when affected the high usage of the machine, was reduced to as low as 65,000 cell updates per second for extended periods.

The solution time (in CPU hours) required for the simulations of the shock induced deformation of the Helium bubble in Chapter 6 was 137.5 hours. The mesh used in this simulation had 334,080 cells. The solution took 36.1 hours of elapsed time to run using OpenMP on the QPSF SGI Origin 3400. The simulations were run to 1 ms of simulation time.

The CPU times required for the range of simulations, including the small diagnostic solutions and the large simulations, are shown in Figure 5.14. The CPU times can be seen to scale with simulation size with a linear relationship. The predictable scale of the solution time, between the small diagnostic solutions and the large Drummond Tunnel solutions, is evident. The gradient of the line is 3/2 as was predicted. There is a slight increase in CPU time as the number of processors is increased, due to the effect of parallel overheads.

The elapsed times for these simulations are shown in Figure 5.15. The decrease in the elapsed time with an increased number of processors is evident. There is little, or no, decrease in the elapsed time for the smallest simulation size on four processors. This demonstrates that the parallel overheads dominate the elapsed solution time for this simulation.

The resulting parallel efficiencies for the OpenMP simulations are shown in Figure 5.16. The efficiency of the single processor simulation can be seen to vary in a range between 0.90 and 1.0. The efficiency being below 1.0 is caused by parallel overheads that are still present on one processor. For a given simulation size, the efficiency of the simulations decreases with an increased number of processors. The efficiency increases with an increase in simulation size. Some of the parallel overheads experienced in these simulations are independent of the simulation size. As a

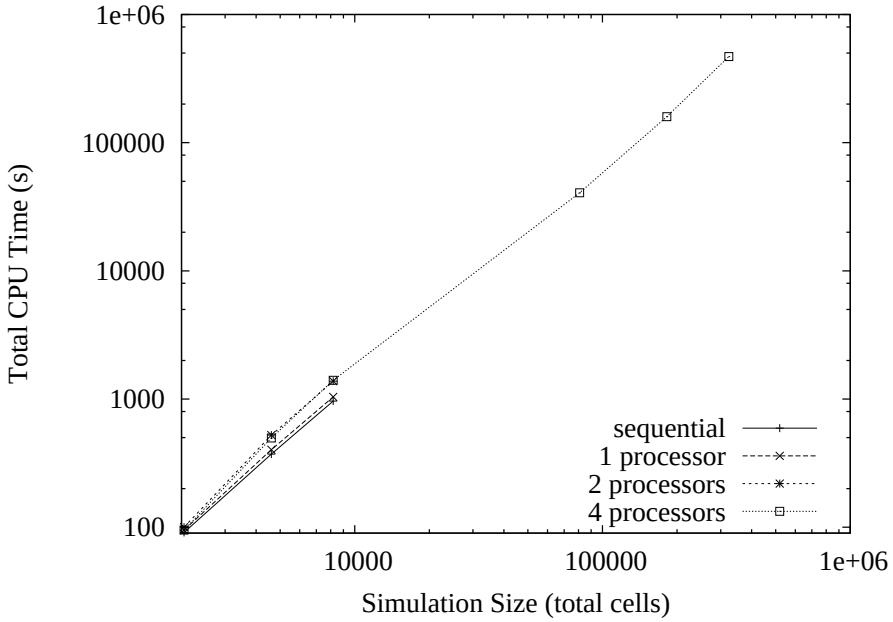

Figure 5.14: CPU times required for simulations using the OpenMP parallel version of MB_CNS on the APAC National Facility.

result, some of the overheads, such as the communication between threads, become less of a proportion of the total solution time as the simulation size is increased. A sharp drop in efficiency is evident for the smallest simulation size on two and four processors. This is caused by the parallel overheads dominating the elapsed solution time.

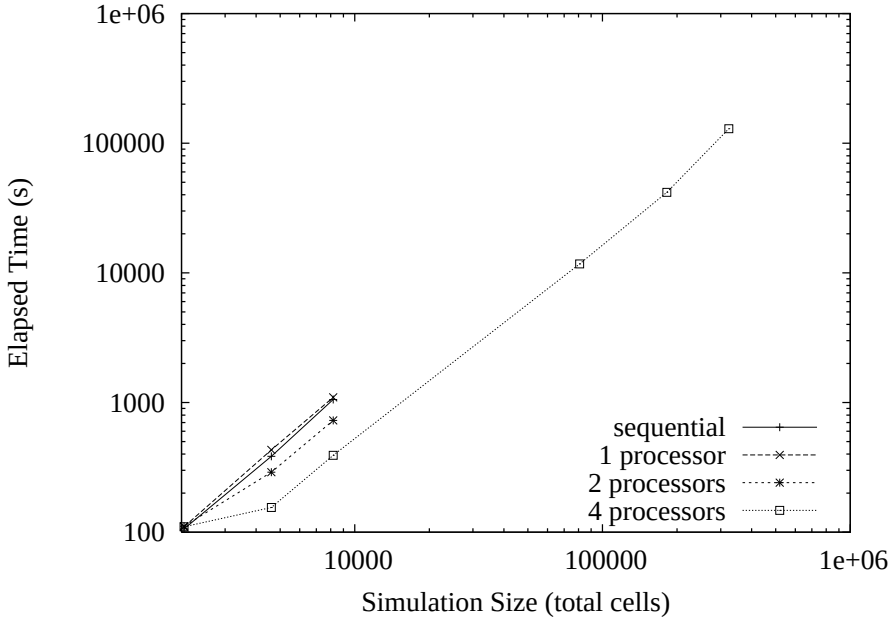

Figure 5.15: elapsed solution times for simulations using the OpenMP parallel version of MB_CNS on the APAC National Facility.

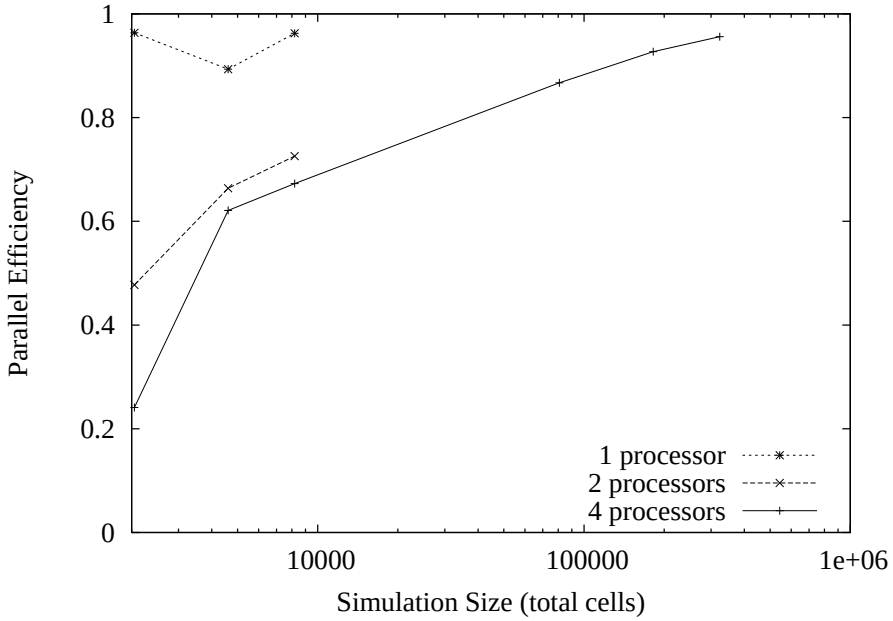

Figure 5.16: Parallel efficiencies achieved in simulations using the OpenMP parallel version of MB_CNS on the APAC National Facility.

Message Passing Interface (MPI)

Since the MPI version of MB_CNS is aimed at the larger simulations, the performance of the MPI version has been analysed for solutions on 4, 8 and 16 processors. Because the MPI version was not applied to any of the large simulations in this thesis, the analysis of the MPI version is limited to the small diagnostic simulations. With the MPI parallel version of MB_CNS, the number of processors must be equal to the number of blocks used. Scaling the number of processors past 16 was not thought to be useful for the relatively small simulations that were run.

Figure 5.17 shows the CPU times required for the small diagnostic solutions with the MPI version of MB_CNS. There is a significant amount of increased work evident with the increased number of processors. This results from the small meshes used, and the large number of blocks required. As the number of blocks are increased, the number of boundaries at which flow data must be transferred increases. This adds work for the simulations on larger number of processors. The sequential simulation was run with four blocks. The CPU times can be seen to scale with simulation size with a linear relationship.

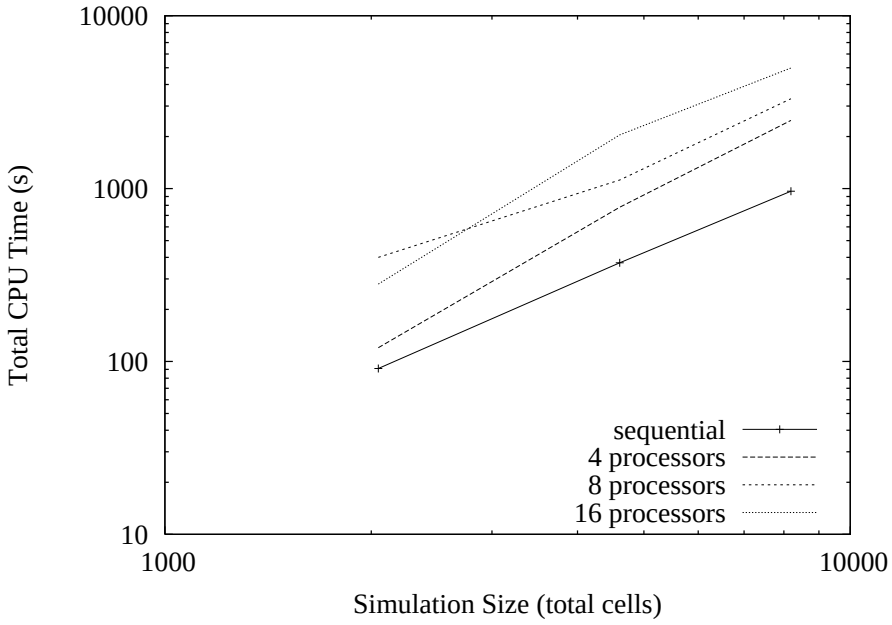

Figure 5.17: CPU times required for simulations using the MPI parallel version of MB_CNS on the APAC National Facility.

Figure 5.18 shows the elapsed times for these simulations. The decrease in the elapsed time with an increased number of processors, up to eight processors, is generally evident; however, the elapsed time for the smallest sixteen processor solution is longer that for the four and eight processor solution. For the smallest mesh size,

there is little, or no, decrease in elapsed solution time for the parallel simulations. This demonstrates that the parallel overheads, especially the communication overhead, dominate the elapsed solution time for the smallest mesh size and for the solutions on sixteen processors.

Figure 5.19 shows the resulting parallel efficiencies for the MPI solutions. The efficiencies across all of the simulation sizes, for the four, eight and sixteen processor simulations are low. The results from the large numbers of processors used for the small solutions, and the resulting proportion of the work that is in the communication overhead. The efficiency for the sixteen processor simulation is below 0.1, showing that the performance is poor. For the four and eight processor simulations, the efficiency increased to the middle mesh size solution and decreases again. The reason for this decrease is not known, but there is a consistent trend between the two processor numbers.

Despite the relatively poor performance demonstrated for the MPI version of the code, we expect that significant gains in efficiency will be obtained as the size of the simulations is scaled upward. The availability of the OpenMP capable supercomputers during this thesis biased the amount of computational work done towards that implementation of the code. With the low prices, and high performance, of commodity PC workstations, we expect that most of the large calculations performed in the future will be on clusters of linux workstations. As a result of this, the MPI version of MB_CNS will become the predominantly used implementation of the code.

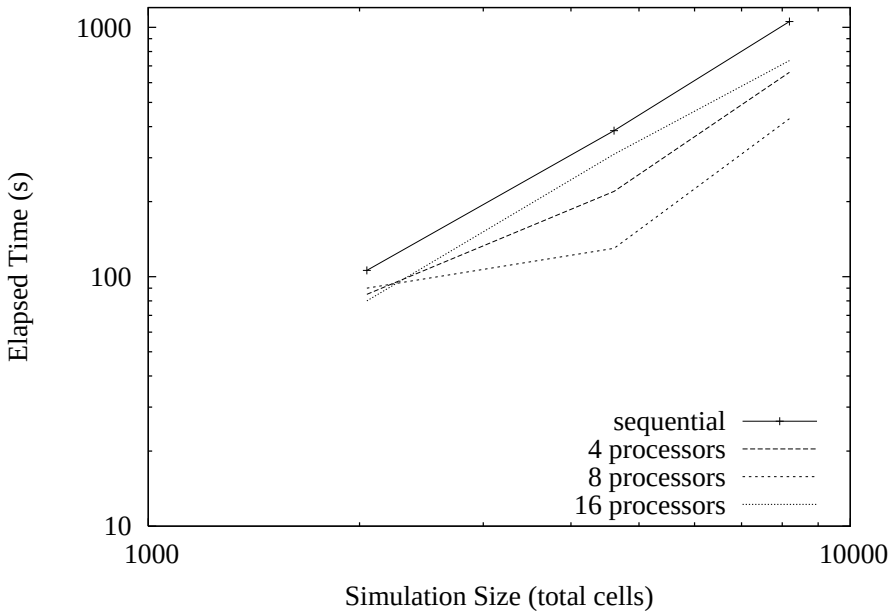

Figure 5.18: elapsed solution times for simulations using the MPI parallel version of MB_CNS on the APAC National Facility.

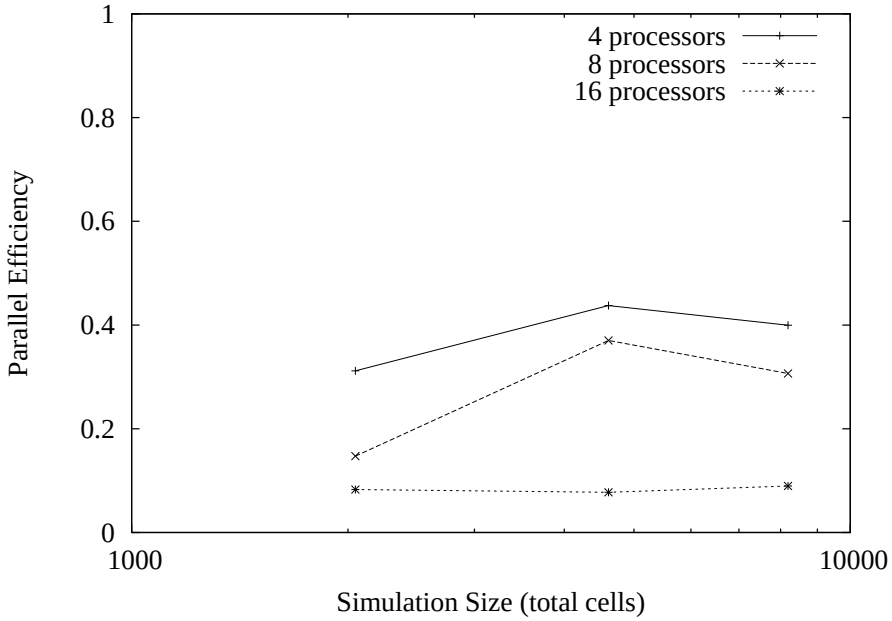

Figure 5.19: Parallel efficiencies achieved in simulations using the MPI parallel version of MB_CNS on the APAC National Facility.

CHAPTER 6

Code Validation: Shock Induced Deformation of Bubbles

The test flow durations achieved in reflected shock tunnels are often small fractions of what is predicted by idealised descriptions of these facilities. Idealised descriptions fail to take into account the real gas phenomena in these facilities, such as the complex interactions that occur as the reflected shock travels back upstream through the oncoming driven gas.

Contamination of the test flow with driver gas was identified by Stalker and Crane [221] as the principle cause of the end of the test time in reflected shock tunnels in high enthalpy operation. The test gas moves along the shock tube ahead of the driver gas so,v for contamination of the test flow with driver gas to occur, some mechanism must be responsible for projecting the driver gas through the test gas.

It is believed that instability in the contact surface between the driver gas and the driven gas is a major contributing factor to the projection of driver gas into the driven gas. As the contact surface moves along the shock tube it may be susceptible to a compressible variant of the Rayleigh-Taylor instability [225] and as the reflected shock reaches the interface it is subject to the Richtmyer-Meshkov instability [25]. The combination of these two instabilities can have a significant effect on the interface, introducing large amounts of vorticity and mixing at the contact surface.

The description of the interaction of shock waves with interfaces between different gases is complicated and simple analytical models are difficult to formulate [89]. The flows involve multiple shock reflections, refractions and the transmission of the shock waves through media of differing sound speed, as well as the instabilities in the gas interfaces. The complex, transient nature of the resulting flow fields means that high resolution computer simulation using, Computational Fluid Dynamics (CFD), is required in order to accurately model the evolution of these instabilities over time.

The importance of these interactions to the operation of a shock tunnel means

that a CFD code used to model shock tunnels must be able to model the mechanisms leading to the instabilities in the contact surface. Idealised experiments are required, both for improving our understanding of the complex phenomenological behaviour of the systems, and for the validation of the numerical techniques and assumptions used in the simulation.

The interaction of shock waves with bubbles of light and heavy gas is studied as a test case for the ability of the code to model the shock induced deformation and instability of these interfaces. The interaction of shock waves with cylindrical bubbles is a test case studied as a model of how shock waves induce instabilities at density stratified gas interfaces, and of the mechanisms by which these instabilities deform the bubble; this deformation results in the formation of large vortices in both cases considered. These instabilities also generate fine-scale turbulence and intensify mixing at the interfaces [89]. the ability of the code to model mixtures of different perfect gases in the same simulation.

This case has well defined initial conditions and experimental photographs, which can be compared directly with the simulations. The particular experimental results that will be used for comparison were preformed by Haas and Sturtevant [89]. These experiments consider isolated, cylindrical gas inhomogeneities, which are impacted by a weak, planar shock. A schematic of the flow field studied by Haas and Sturtevant is shown in Figure 6.1. Two different test gases are considered, both leading to different shock interaction and deformation phenomena.

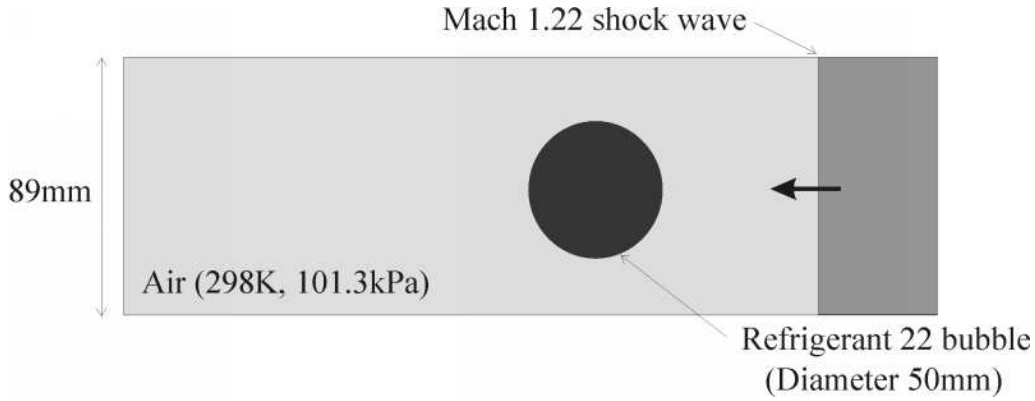

Figure 6.1: Schematic diagram of the experiments performed by Haas and Sturtevant [89] and used in the comparison with the simulations in this chapter.

In addition to code validation and for its analogy to shock tunnel flows, this is an interesting case in its own right. The study of flows involving contact surface instabilities is also of interest in the mixing of light gaseous fuels being injected in Scramjet engines. Given the short amount of time available for the mixing of the fuel

and air available in order for efficient combustion to take place in the combustion chamber, a method of enhancing the mixing are important. The passage of the fuel, with a different density to the air, through shocks in the combustion chamber may result in Richtmyer-Meshkov instabilities. The resulting deformation would enhance mixing and increase the area of contact between the two gases. This mixing mechanism has been studied by Lee [132] using three dimensional simulations of a Scramjet combustor.

As the shock passes over the bubble, both the test gas in the cylinder and the surrounding air are accelerated impulsively. The bubble is not significantly disturbed during the time that the shock passes through the bubble; however, it is left in an unstable configuration by the shock and it continues to deform over time.

A contact surface between gases of differing density is subject to Rayleigh-Taylor instabilities if it is experiencing an acceleration perpendicular to the surface in which the heavier fluid is pushing on the lighter fluid. The instability causes the heavier fluid to enter the lighter fluid, resulting in mixing and turbulence at the interface. The study of the stability of interfaces between fluids of differing density began with Taylor [232], who performed a linear stability analysis of a harmonically perturbed interface under gravitational acceleration. Corresponding experiments were performed which agreed with the theory of Taylor for initial sinusoidal amplitudes below 0.4 times the initial wavelength [253].

The Richtmyer-Meshkov instability can occur following the interaction of a shock wave with an interface separating two fluids of different properties. Any perturbations in the interface will be amplified following the passage of the shock. The driving mechanism for the Richtmyer-Meshkov instability is the baroclinic generation of vorticity, which results from the misalignment of the pressure gradient across the shock and the local density gradient across the fluid interface [25]. The Richtmyer-Meshkov instability is also referred to as the shock-induced Rayleigh-Taylor instability [89].

The shock-induced impulsive acceleration of an interface with sinusoidal perturbations was studied analytically by Richtmyer [192]. A linear representation of the flow following the passage of the shock was used and a formula that captures the early time motion of the interface. This formula was applicable to a restricted range of Mach numbers and density ratios. The first experimental modelling to study the interaction of shock waves with density inhomogeneities were performed by Meshkov [146], who studied the passage of a shock wave through a perturbed planar surface. The experimental results of Meshkov agreed qualitatively for early times with the analytical model of Richtmyer.

The shock bubble interaction flow field represents a fundamental departure from

the classical Richtmyer-Meshkov instability, in which a planar shock passes through a perturbed surface, or a perturbed shock passes through a planar surface [179]. Such a system initially exhibits a linear growth but, in contrast, the shock bubble interaction exhibits a non-linear growth even from early times. Attempts have been made to apply the linear theory to the shock bubble flow field [142].

The passage of the shock induces vorticity at the surface of the bubble and, therefore, shear between the two gases. Shear in the interface between the gases of differing density means that the interface is susceptible to Kelvin-Helmholtz instabilities. The numerical modelling of these instabilities is heavily dependent on the grid resolution used and the resulting numerical viscosity.

6.1 Experimental Modelling

The experimental study of Haas and Sturtevant [89] was conducted in the GALCIT 15 cm diameter shock tube at the California Institute of Technology. Due the requirement for uniform inflow conditions, a 'cookie-cutter' test section is used in the shock tube, further reducing the test section to 8.9 cm in square section and 120 cm in length. They compare their results to acoustic theory for wave transmission through media of differing density. Other experimental work in this area was conducted by Jacobs [109].

The experiments provide a recorded physical flow pattern, using shadowgraphs, with which numerical simulation can be validated. The experimental study considered both cylindrical and spherical bubbles; however, the simulation in this chapter will consider only the cylindrical bubble cases.

The experimental setup aims to demonstrate weak shock waves interacting with cylinders of two different gases in air; one is a lighter gas, Helium, and the other is a heavier gas, Refrigerant 22. These gases have sound speeds significantly different from air: at room temperature, the speed of sound in Helium is 2.9 times faster than in air and, in Refrigerant 22, it is 1.9 times slower than in air. The properties of the gases used in the simulations are shown in Table 6.1. The gases inside the cylinder were enclosed in a film and were at the same temperature and pressure as the surrounding air. When the incident shock wave contacts the Helium the shock speed will accelerate, meaning that the Helium cylinder will act as a divergent lens; with the Refrigerant 22, the shock will decelerate, the cylinder acting as a convergent lens.

Before the passage of the shock, the bubbles are stationary and in thermal and mechanical equilibrium with the surrounding air. A small amount of leakage of the

Table 6.1: Properties of the gases used in the experiments

Gas	Ratio of Specific Heats (γ)	Gas Constant (R)	Speed of Sound at 296 K
Air	1.4	287 J/(mol.K)	344.9 ms^{-1}
Helium	1.667	2077 J/(mol.K)	1012.4 ms^{-1}
Refrigerant 22	1.249	91 J/mol.K)	180.3 ms^{-1}

test gas was observed in the experiments and was accounted for in the simulations. The experimental runs consisted of a cylinder of 5 cm diameter in an experimental section 8.9 cm; the gases were at atmospheric pressure, assumed to be 101.3 kPa, and at ambient temperature which was assumed to be 298 K.

The cylindrical volume, containing the test gas was enclosed in a 0.5 μm thick nitrocellulose membrane wrapped on 5 cm diameter, 3 mm thick pyrex windows, which served as the transparent ends of the cylinder.

The experimentation of Haas and Sturtevant was carefully controlled; however, an ideally isolated experimental arrangement could not be achieved. The factors that must be accounted for when setting up the numerical model of the system, and in comparison of the numerical and experimental results, include:

1. A spark shadowgraph optical system was used to capture the waves and interfaces present in the flow. Only one photograph was recorded per run of the shock tube. The sequence of photographs throughout the course of the deformation of the bubble was recorded by delaying the time that the photograph was taken successively with each run. This relies on the repeatability of conditions in the shock tube between runs. With this method, clear photographs with good spatial resolution are obtained; however, this method results in an uncertainty in shock speed estimation, which was thought to be of the order of 10% or less under most circumstances.

2. The inertia of the nitrocellulose film that separates the two gases will have an effect on the flow, although this is difficult to quantify. This will generally show up as a delay in the interface accelerating early in the experiment. This influence may have varied between experiments, due to differences in the effectiveness of the rupture. The film may continue to influence the flow throughout the experiment as it is carried downstream.

3. It was noted that there was a certain amount of diffusion of gas across the film

separating the test gas from the surrounding air. The experimentally measured shock speeds inside the cylinder ($943\,ms^{-1}$) were quite different than from pure Helium gas ($1073\,ms^{-1}$); the shock speed was calculated to correspond to a 28%, by mass, contamination of the Helium test gas with air. To compensate for this, the initial condition was used in the simulations assumed homogeneously mixed mass fractions of 72% of Helium and 28% for air; this provides a good model of the contaminated Helium test gas. In the case of the refrigerant 22 cylinder, the experimental shock speeds showed that the contamination of the test gas with air was negligible; it was estimated as 3.4% by mass, resulting in a variation in shock speed below the error in shock speed measurement.

4. The cylindrical sections were kept at a slight over-pressure to stretch the cylinder to its intended shape. This would cause the test gas to leak into the surrounding air. This gas would diffuse out and cause sound speed gradients in the region surrounding the cylinder. To counter this, the air surrounding the cylinder was continually refreshed and so these gradients were small. They were thought to be insufficient to affect the comparison with computational results which assume a perfect separation of the gases.

5. The experimental section, being only 8.9 cm in diameter, was not muich larger than the size of the bubble. This means that shock waves reflected, and refracted, from the bubble would quickly reflect back into the flow from the side walls. This increases the complexity of the flow, but can be modelled by setting the computational domain to this same width.

6. The various support parts used in the generation of the cylindrical section of gas, such as the end windows, connecting beam and the membrane itself, were thought to cause some spurious gas effects at the windows of the shock tube and so were captured in the shadowgraphs. These effects did not affect the majority of the flow and, for the most part, are limited to the lower half of the photographs.

Where possible, these factors should be accounted for in the simulations or noted in the comparison with the simulation results.

Haas and Sturtevant [89] recorded static pressure traces at several points along the plane of symmetry as a part of their experimental study. These pressure traces would provide a quantitative measure of a flow property that could be used for comparison with the simulations; however, Quirk and Karni [184] noted that the traces cannot be relied on as an accurate benchmark since the measuring process

was invasive. The measurements were made by using a movable wall placed in the shock tube with a pressure transducer mounted in it. This means that the transducer was recording was the pressure behind the waves reflected off of this surface and not the local pressure, had the wall not been there.

6.2 Previous Computational Modelling

The experimental work of Haas and Sturtevant has been the subject of a significant number of numerical studies. The first attempt at numerical simulation of this case was by Picone and Boris [179]. However, in these earlier studies the flow was modelled using a single gas rather than the exact binary system used by the experiment.

Quirk and Karni [184] extended the early work of Picone and Boris [179] by carrying out high resolution numerical simulation of the experimental work of Haas and Sturtevant [89]. Quirk and Karni used a CFD code that utilised adaptive mesh refinement. The algorithm led to between a forty and fifty-fold decrease in computer time required to solve the flow field, for a given resolution achieved. The elapsed solution time was further reduced by running the simulations in parallel on a cluster of workstations. Figure 6.2 shows an images from the simulations of Quirk and Karni [184], demonstrating the fine resolution of the flow field that was achieved.

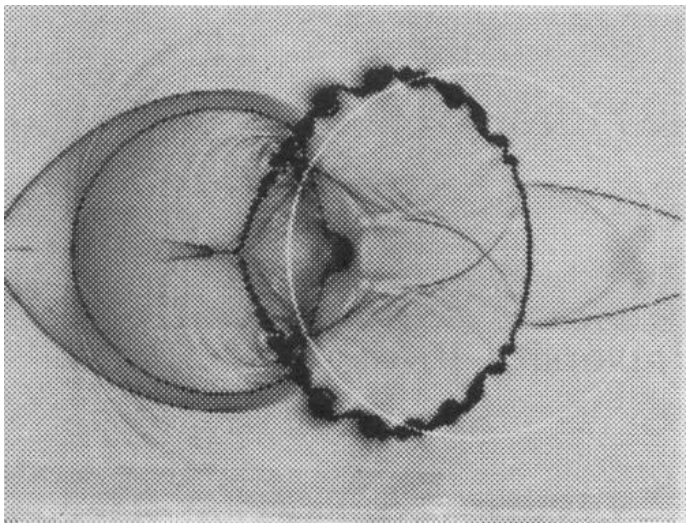

Figure 6.2: An image from the simulation of Quirk and Karni [184], demonstrating the high resolution that was achieved.

Yang, Kubota and Zukoski [251] applied the flux corrected transport (FCT) computational methodology to the conservation equations in order to analyse the

shock bubble interaction. They derived scaling laws for the amount of circulation produced in the interaction, the time scale of the interaction and the vortex spacing.

Don and Quillen [62] applied two high order shock capturing schemes to the simulations of shock bubble interactions. The simulations were evolved to late times, examining the effect of the instability in promoting the mixing of the two gases. The focus of the study was the investigation of the effectiveness of the numerical techniques used in simulating the flow.

Bagabir and Drikakis [9] investigated the Mach number effects on the interaction of shock waves with cylindrical bubbles. The variation in the evolution of the flow fields, resulting from the interaction of the bubbles with shocks of different Mach number, was examined. At higher Mach numbers, larger distortions of the bubble were shown to occur and the resultant rolled-up structure was shown to form at earlier times.

Morris and Monaghan [160] simulated the interaction using the Smoothed Particle Hydrodynamics technique that was described in Section 3.2.1. The case was modelled as a demonstration of an artificial viscosity switch that had been developed. The simulations were compared with other simulations, obtained using a more conventional finite-volume based technique, and no comparison with experiments was provided.

6.3 Simulation Using MB_CNS

Numerical simulation of the experiments of Haas and Sturtevant [89] was performed with the compressible CFD code MB_CNS [114]. Simulations were performed using the adaptive flux calculator described in Section 3.1.1. The cylindrical initial conditions and relatively wide test section meant that the simulation could be performed as a two dimensional simulation.

The computational mesh used in the simulations is shown in Figure 6.3. MB_CNS is multi-block code so the initial arrangement of gases in the simulation was achieved by forming the cylindrical gas layout out of the blocks. The blocks inside the cylinder were initially assigned as containing Helium, or Refrigerant 22, and the blocks outside of the cylinder were assigned as containing air.

In hind-sight the design of the mesh would have benefited from the use of a curved upstream boundary. This would have not only saved computational effort in solving the flow upstream of the bubble, but would have helped to prevent waves from being reflected back into the flow field. The mesh was refined towards the original position of the bubble. For a transient flow field such as this, much of

the computational effort was wasted in the fine mesh in the blocks in the original position of the bubble.

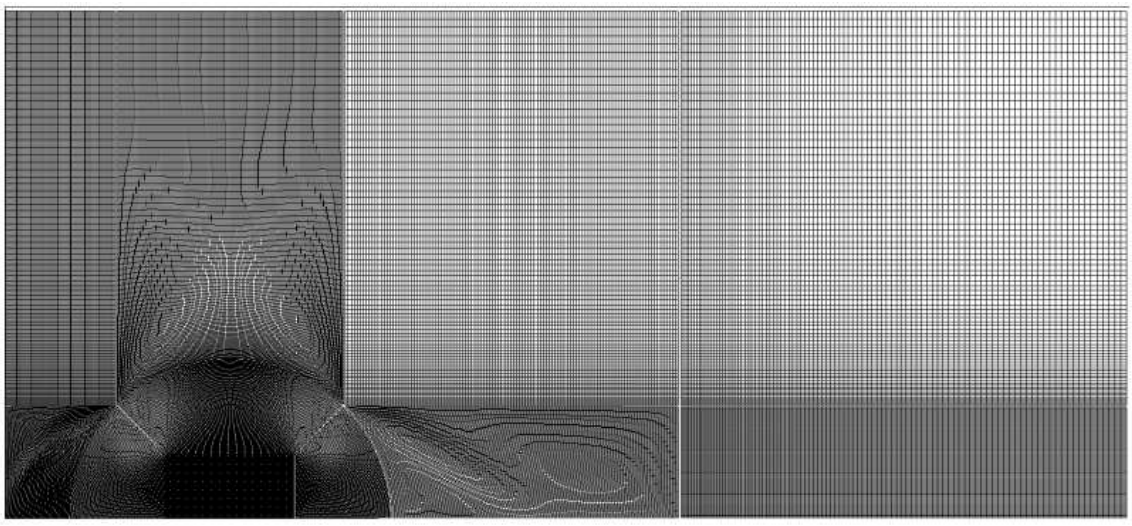

Figure 6.3: Computational mesh used in the simulations of the shock bubble interactions. This is the fine mesh used in the mesh refinement study.

The solution of the flow field was performed in parallel using OpenMP. The computational performance was described in Section 5.3.1.

Mass conservation was solved for the two species in the simulation and their mass fractions were stored for each cell. The properties of the gas in each cell were calculated using mass weighted averages of the gases present in the cell.

The experimental shadowgraphs of Haas and Sturtevant [89] can be compared directly with the simulation results. An experimental shadowgraph represents an integration of the curvature of the density field across the entire width of the shock tube facility used to perform the experiment. Numerical shadowgraph images were constructed from the simulation results using the magnitude of the second derivative of density. The shadowgraph variable, which was plotted, was normalised using the same technique as was used for the numerical Schlieren images described by Quirk and Karni [184].

6.3.1 Helium Bubble

Figures 6.4, 6.5 and 6.6 show the sequence comparing the experimental shadowgraphs produced by Haas and Sturtevant [89] with the numerical shadowgraphs produced from the simulation results, for the Helium bubble case. The experimental shadowgraphs are shown on the left and the numerical shadowgraph images from

the simulation are shown on the right. To make an easy comparison with the experimental results, the incident shock is now shown to be moving from right to left and the original position of the bubble is marked by what looks like a dark circle with a T-shaped support in the experimental images and a fine ring in the simulations.

Frame (a) of the sequence shows the Helium bubble 32 μs after the incident shock contacted the windward edge of the bubble. When the shock contacted the bubble, part of the shock was transmitted through the Helium and part was reflected back upstream. The higher sound speed of the Helium gas means that the transmitted shock is refracted and curves outward, moving ahead of the plane of the incident shock. The reflected shock is curved outwards and is moving upstream from the bubble surface. These shocks meet at a triple point which lies close to the surface of the bubble. The remaining segments of the incident shock, which are passing around the bubble, remain planar. The bubble has already undergone a slight deformation with its windward side being flattened with the movement of the flow. Also visible in the frames is the circular shape of the original supporting ring in the shock tube.

Quirk and Karni [184] comment that the curved reflected wave is not accurately described as a simple shock; it is not a simple expansion either. Its properties near the axis of flow symmetry are those of a weak expansion; however, away from the axis there is little deformation of the bubble surface and this wave has the properties of a reflected shock. Behind the reflected wave is an expansion system which results in the mixed properties.

Frame (b) shows the bubble at 52 μs after the initial shock contact. The transmitted shock, moving through the high sound speed Helium gas, has moved well ahead of the transmitted shock and is over half way through the bubble. As the transmitted shock moves through the leeward half of the bubble, the bubble height is decreasing and part of this transmitted wave has emerged from the top and bottom of the bubble. Due to the geometry of the bubble, this wave has almost converged with the reflected wave at the transmitted wave. Two shocks normal to the bubble surface are formed where this shock meets the bubble interface; these shocks can not be seen in the experimental photographs, but are evident in the simulation results. This point is joined to the bubble surface by Mach stem. This four shock configuration is termed by Henderson et al. [94] as a Twin Regular Reflection-refraction. The windward side of the bubble has continued to flatten. The reflected shock has moved further outward upstream from the bubble.

Frame (c) shows the flow at 62 μs. Around this time the transmitted shock emerges from the leeward side of the bubble. At the instant that the transmitted wave emerged from the leeward side of the bubble, the reflected part of this wave converged on the centreline at the bubble interface. This wave is reflected internally

back upstream inside the bubble. These waves are weak and just show up in the photographs, but are important for the stability of the windward interface.

Frame (d) shows the flow at $72\,\mu$s. Using their simulations, Quirk and Karni (1994) identify in this frame the contact surface that would be expected to emanate from the point of the TRR [184].

Frame (e) (in Figure 6.5), at $82\,\mu$s, shows that the internally reflected waves cross over one another and diverge. These waves extend across the leeward side of the bubble and, as they sweep outwards, converge with the transmitted waves.

In Frame (f), at $102\,\mu$s, along the axis of flow symmetry the side shock and the transmitted shock have almost merged. Meanwhile, both the original reflected wave and the transmitted shock have reflected from the walls of the shock tube and pass back over the bubble. The incident shock can been seen to have started to diffract around the leeward side of the deformed bubble. The internally reflected wave, described in Frame (c), has emerged from the windward side of the bubble. As this shock emerges from the bubble this adds to the instability on the bubble surface. This wave is weak enough not to have appeared on the experimental shadowgraphs.

In Frame (g), at $245\,\mu$s, the windward side of the bubble, which was accelerated the most by the shock, has moved past being flat and begins to fold inwards on itself. The original deformation is caused by the baroclinic vorticity generated at the bubble interface which forces the windward face of the bubble to be pressed in together into the plane of symmetry. As this progresses the face of the bubble folds further, into the less dense fluid. In this Frame, waves which have reflected from the walls of the shock tube can be seen interacting with the bubble. These waves have a significant effect on the pressure field in and around the bubble, but, for the most part, they are weak enough to have a small effect on the actual deformation of the bubble.

In Frame (h), at $427\,\mu$s, the windward side of the bubble has continued to fold through and has formed into a jet that passes through the centre of the bubble. By this stage no waves are visible in the Frame, as all have either moved downstream or, for the waves reflected from the side walls, have been scattered and are now weak enough not to appear.

In Frame (i) (in Figure 6.6), at $674\,\mu$s, the generation of vorticity, which was mentioned in reference to Frame (g), has caused the front of the jet to continue to roll over into a mushroom shape. This effect becomes more pronounced as the jet reaches the leeward side of the jet; it is more difficult for the jet to penetrate into the high density air on the other side of this boundary and so it is spread out laterally.

The only significant difference between the simulations and the experiments is evident in the last frame, Frame (i). As the heavy gas upstream of the bubble penetrates the Helium gas, the stiffness of the interface at the head of this gas appears to be less in the simulations than in the real flow. Additional folding back of the interface is evident in the simulated image. This may be due to limitations in the ability of the simulations to reproduce the actual stiffness of the interface; however, it may also be due to the effect of the nitrocellulose influencing the stiffness of the gas interface.

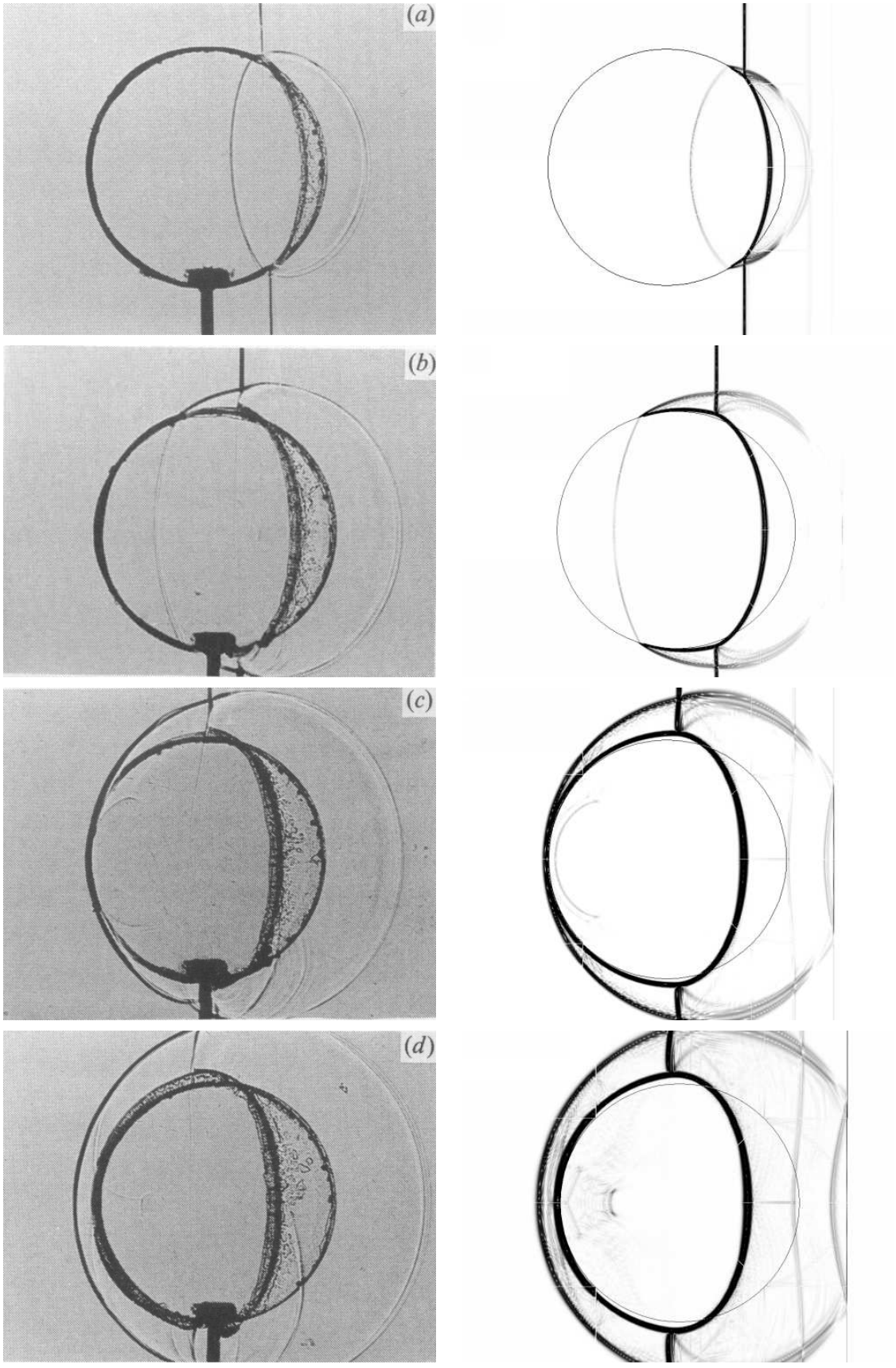

Figure 6.4: Part one of the sequence of frames comparing the experimental shadowgraphs (left) with the numerical shadowgraphs (right) for the helium bubble. The experimental shadowgraphs are reproduced from Haas and Sturtevant [89].

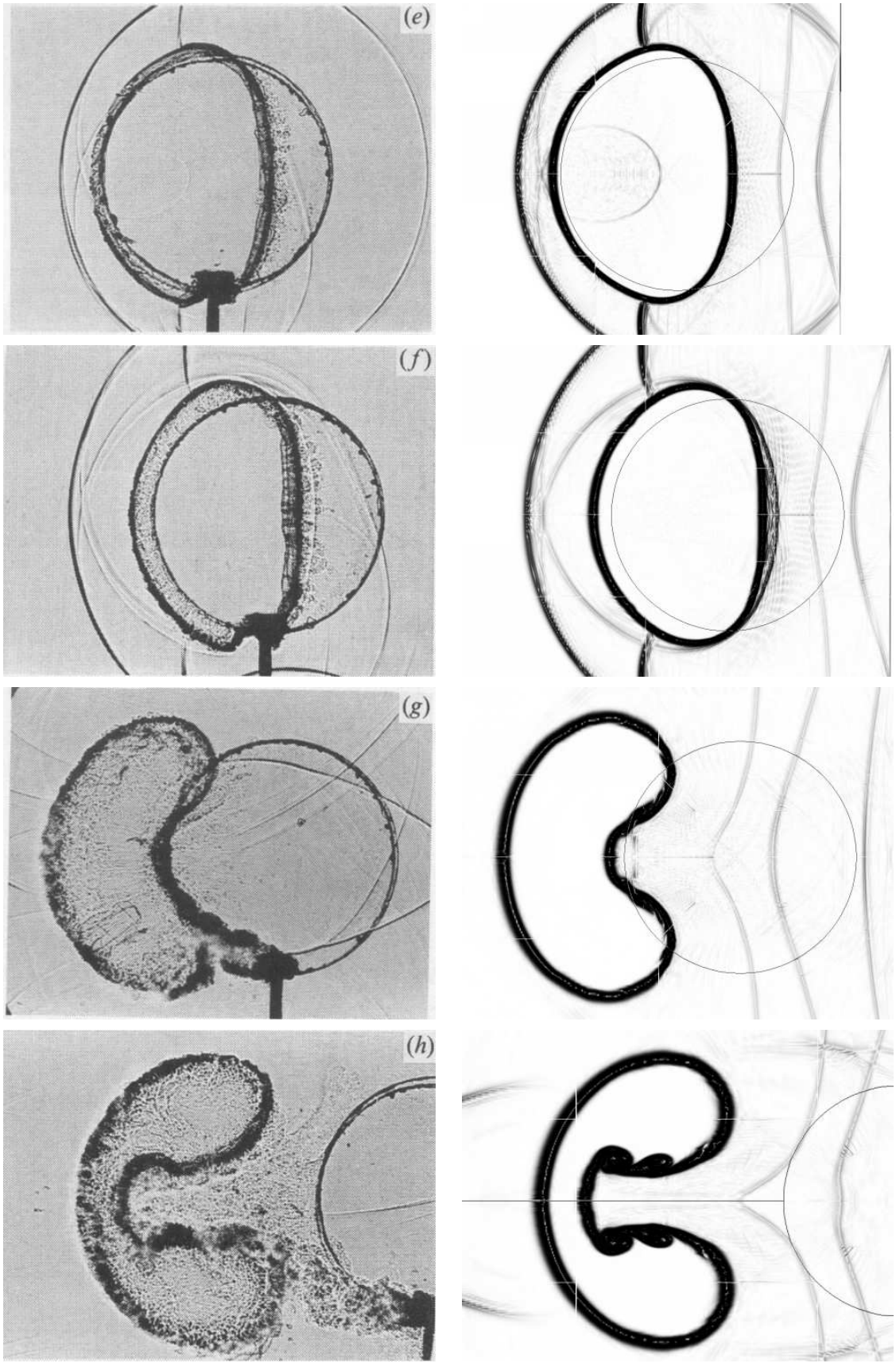

Figure 6.5: Part two of the sequence of frames comparing the experimental shadowgraphs (left) with the numerical shadowgraphs (right) for the helium bubble. The experimental shadowgraphs are reproduced from Haas and Sturtevant [89].

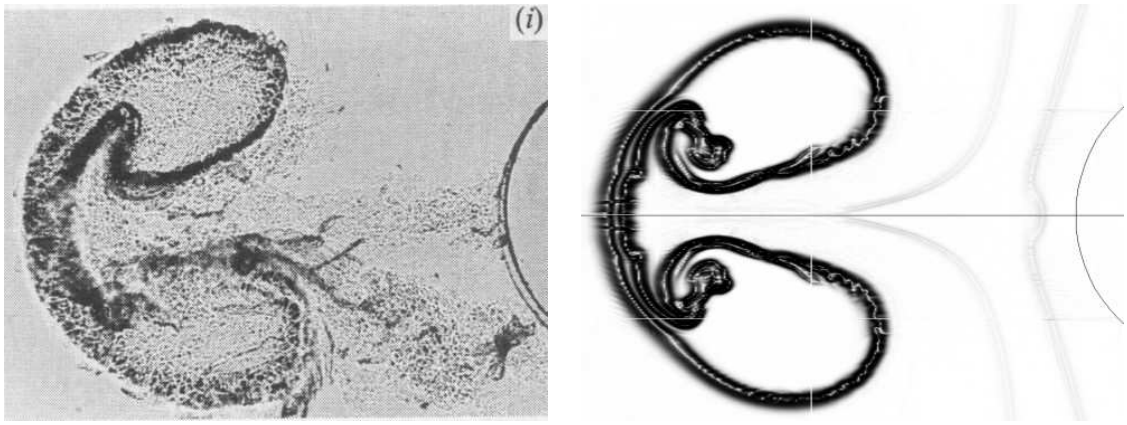

Figure 6.6: Part three of the sequence of frames comparing the experimental shadowgraphs (left) with the numerical shadowgraphs (right) for the helium bubble. The experimental shadowgraphs are reproduced from Haas and Sturtevant [89].

6.3.2 Refrigerant 22 Bubble

for the Refrigerant 22 bubble case, Figures 6.7 and 6.8 show the sequence comparing the experimental shadowgraphs produced by Haas and Sturtevant [89] with the numerical shadowgraphs produced from the simulation results. The experimental shadowgraphs are shown on the left and the numerical shadowgraph images from the simulation are shown on the right. Again, the incident shock is moving from right to left and the original position of the bubble is marked by what looks like a dark circle with a T-shaped support in the experimental images and a fine ring in the simulations.

Frame (a) shows the Refrigerant 22 bubble 55 μs after the incident shock contacted the windward edge of the bubble. The incident shock, passing over the windward side of the bubble, is still planar. As the incident shock contacts the bubble part of the shock is transmitted through the bubble and part is reflected back upstream; Quirk and Karni [184] performed a one dimensional analysis for the moment the incident shock impacts the bubble which suggested that the transmitted component of the shock is 6.4 times stronger than the component that is reflected. The transmitted shock is refracted strongly as it moves through the bubble and lags behind the incident shock. Deformation of the bubble is already evident with it's windward side being flattened into the flow.

In Frame (b), at 115 μs, the two segments of the incident shock have passed over to the leeward side of the bubble and can be seen to be diffracting around the cylindrical shape of the bubble. Unlike with the Helium bubble, the point at which the transmitted shock and the incident shock meet does not move away from the bubble surface. The strong inward refraction of the transmitted shock has meant that material inside the bubble has started to be focused towards the centreline of the bubble and towards the leeward side. Haas and Sturtevant (1987) observed that the refracted shock is thickened at its two endpoints. Quirk and Karni (1994) conclude that this thickening is must be due to some three dimensionality in the shadow photographs and that it is not a two dimensional flow feature; however, in the simulation a series of additional waves can be seen in the flow around this time which could explain this thickening. No explanation of it's origin was given by the Haas and Sturtevant [89]. Quirk and Karni [184] also noted that, at this point in time, the bubble interface shows signs of incipient instabilities, where vorticity has been generated on the interface of the bubble .

In Frame (c), at 135 μs, the incident shock has refracted further around the cylinder and has continued to focus the material in the bubble towards the centreline. The two segments of the incident shock have continued to diffract around the leeward

side of the bubble and meet the bubble interface at with the transmitted shock. Waves can be seen to have reflected from the upper and lower wall of the shock tube and are re-entering the flow field.

In Frame (d), at 187 μs, shows that the transmitted shocks are just about to focus on the bubble centreline. Simulations by Quirk and Karni (1994) show that this focusing results in a peak pressure 2.1 times higher than behind the transmitted Mach 1.22 shock wave. The diffracted segments of the transmitted shock have crossed over and continue to move outward through one another. Weak contact discontinuities, between regions that have been processed by either the diffracted or planar incident shock, are visible in this frame. The reflected shocks from the top and bottom walls of the shock tube have now started to pass through the bubble. Deposited vorticity has caused the rolling up of the bubble interface to continued to grow.

In Frame (e) (in Figure 6.8), at 247 μs, the two segments of the transmitted wave have emerged from the leeward side of the bubble after having focused on the centreline. These waves are cylindrical in shape and are moving outward. The elevated temperature caused by the focusing of the transmitted shocks on the centreline caused a high speed behind the transmitted shocks and causes the leeward interface of the bubble to become wedge shaped.

Frame (f), which is at 318 μs, shows the region downstream of the bubble. This Frame is taken relatively close in time to Frame (g), but is of interest as it records the motion of the transmitted and diffracted waves. Of the waves downstream of the bubble, the two diffracted segments of the transmitted shock can be seen. A Mach stem has formed near the centreline. The weak contact surface behind the transmitted shock can be seen. The strongly curved transmitted shock can be seen close behind these waves.

In Frame (g), at 342 μs, as well as the significant deformation of the leeward side of the bubble, the whole bubble can be seen to have moved downstream from its initial position at the rings on the side windows. The outward motion caused by the vorticity in the bubble interface has caused the bubble to be elongated in the span-wise direction. A complicated wave, reflected internally from the focused transmitted waves, can be seen moving upstream inside the bubble.

In Frame (h), at 417 μs, the bubble has continues to fold under the influence of the interface vorticity. The wave that was internally reflected, following the focusing of the transmitted shock, has partly emerged from the upstream side of the bubble. This Frame is complicated by the waves reflected from the upper and lower walls of the tube. Significant instability is evident in the bubble interface, particularly along

the leeward side.

In Frame (i), much later in time at 1020 μs, shows the bubble as the two large vortex cylinders forming at the upper and lower edges of the leeward side are developing; the whole bubble evolves into two large vortices. The blurring evident in this Frame is due to three dimensionality in the cross section through the flow recorded by the shadowgraph.

The simulations reproduce the experimental flow field accurately. Both the external and internal wave stuctures, and the evolution of the bubble are reproduced accurately. It appears that the simulations have also accurately reproduced the level of instability in the bubble interface. The level of instability in the experimental images is harder to ascertain, as the shadowgraphs are taken through the width of the test section; however, the magnitude of the disturbed regions the windward and leeward sides of the bubble are very similar, between the simulations and the experiments.

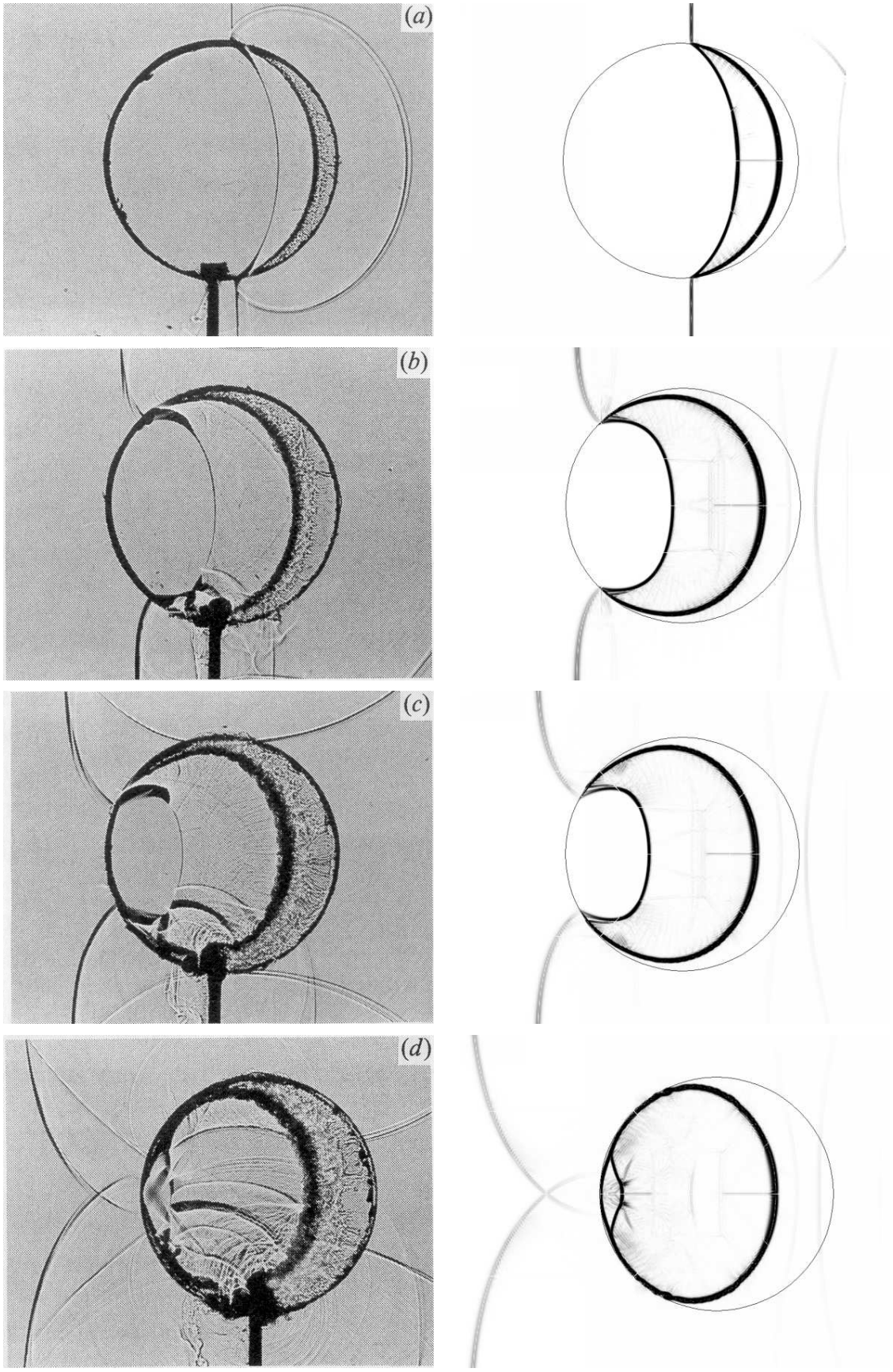

Figure 6.7: Part one of the sequence of frames comparing the experimental shadowgraphs (left) with the numerical shadowgraphs (right) for the Refrigerant-22 bubble. The experimental shadowgraphs are reproduced from Haas and Sturtevant [89].

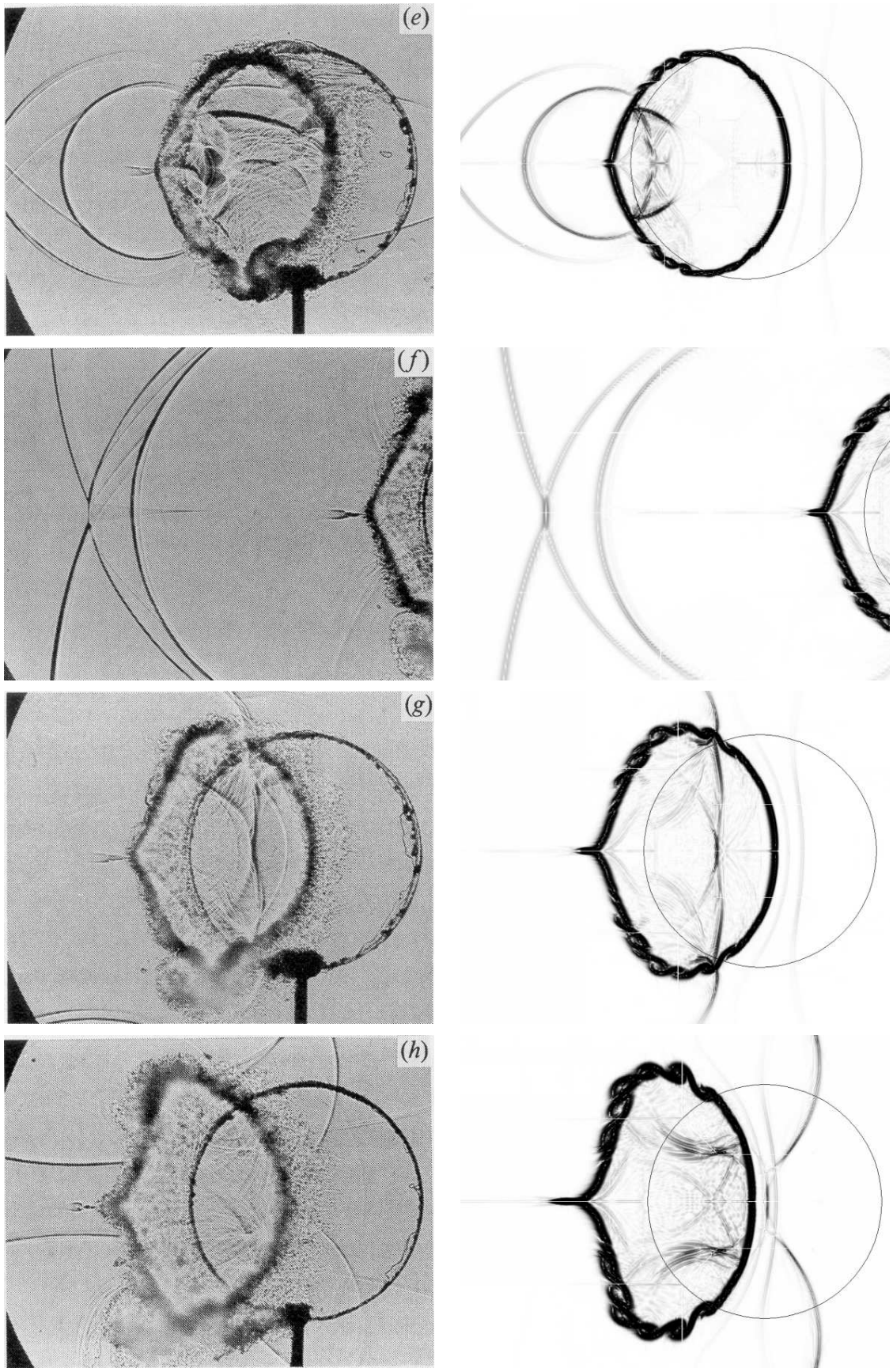

Figure 6.8: Part two of the sequence of frames comparing the experimental shadowgraphs (left) with the numerical shadowgraphs (right) for the Refrigerant-22 bubble. The experimental shadowgraphs are reproduced from Haas and Sturtevant [89].

6.3.3 Discussion of the Results

Effect of Mesh Refinement

Figure 6.9 shows the effect of refining the mesh from the coarse mesh, with 83,520 cells, to the fine mesh, with 334,080 cells, on the simulation results. The fine mesh was shown in Figure 6.3. There only a qualitative difference between the two flow fields. The characteristics of the bubble, and the internal and external waves are all unaffected by the refinement, apart from the effect of resolution. The level of diffusion at the bubble interface is increased, although not by a significant amount.

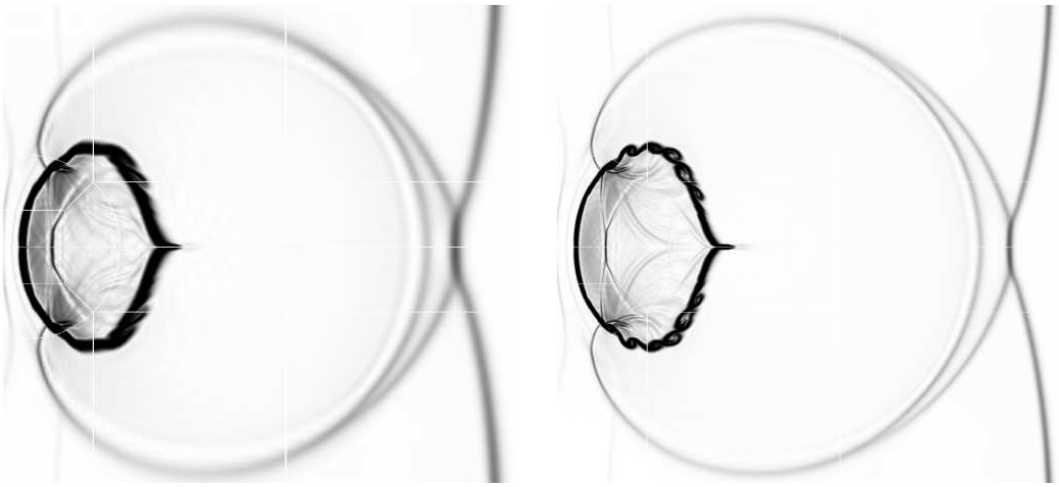

Figure 6.9: Effect of mesh refinement on the simulation of the Refrigerant 22 bubble. The flow field from the coarse mesh is shown on the left and from the fine mesh on the right.

Vorticity Generation

For both the heavy and the light bubbles, the deformation is driven by vorticity impulsively generated at the interface by the passage of the shock wave through it. Vorticity is produced whenever there is a misalignment in the gradients of the density and pressure fields; in this case, the local pressure gradient through the shock. The source of the vorticity is the shown in the baroclinic torque term (on the far right) in the Curl of the Momentum Equation, Equation 6.1, for a compressible fluid:

$$\frac{\partial \omega}{\partial t} - (\omega \cdot \nabla)\, u = -\omega \nabla \cdot u + \frac{\nabla \rho \times \nabla p}{\rho^2} \tag{6.1}$$

where ω is the vorticity, v is the velocity, p is the pressure and ρ is the density [89]. The left side of the equation represents the total change in vorticity following the motion of the fluid. The first term on the right side represents vortex stretching; this

term is zero for two dimensional planar simulations such as this study. The second term on the right side is the baroclinic torque, which results from misalignment of density gradients with pressure gradients. Since the density gradient is only present on the bubble interface, vorticity is generated there.

Figure 6.10 shows contours of baroclinic vorticity generation in the lower half of the frame and accumulated vorticity in the upper half of the frame, during the passage of the shock for the Helium bubble case. This figure shows that once the vorticity has been deposited at the bubble interface, it remains there, continuing to deform the surface.

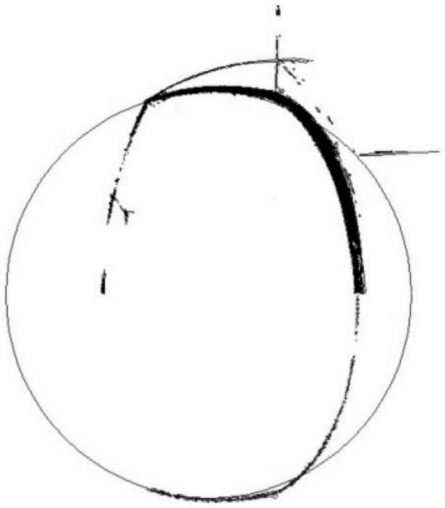

Figure 6.10: Contour plots of baroclinic vorticity generation in the lower half of the frame and accumulated vorticity in the upper half of the frame, during the passage of the shock for the Helium bubble case.

The majority of the vorticity is produced where the shocks intersect the bubble interface due to the baroclinic torque. Since the incident shock is diffracted around the leeward side of the bubble, it is significantly weakened at the point at which it intersects the bubble. This means that it is only the refracted shock that produces a significant amount of vorticity on the leeward side of the bubble and that more vorticity is deposited on the windward side of the bubble than on the leeward side.

In the case of the lower density Helium bubble, the density gradient vector is directed inwards. With the direction of the pressure gradient through the shock, this results in an anti-clockwise baroclinic torque applied to the bubble surface. This results in the windward surface of the bubble being folded in through itself as shown in sequence of the Helium bubble interaction. With the higher density Refrigerant 22 bubble, the density gradient vector is directed outwards from the bubble. This vector combined with the shock pressure gradient results in a clockwise

torque applied to the interface. This results in the outward folding of the Refrigerant 22 bubble evident in the sequence.

Effect of Numerical Instabilities

The simulations performed using MB_CNS were not as fine resolved as the simulations performed in Quirk and Karni [184]. This was because of the adaptive mesh refinement scheme used in the simulations of Quirk and Karni. The fine grid size, and therefore low numerical viscosity, combined with the lack of a real viscosity meant that small numerical disturbances grew quickly in their simulations. In addition, physical flow processes, which would normally be suppressed by the viscosity in the fluid may be over-estimated [184]. The coarser mesh used in the MB_CNS simulations, combined with the implementation of a real viscosity, meant that these simulations were not as susceptible to this type of numerical instability. The simulation of the physical instabilities in the flow were shown to be reproduced accurately by the MB_CNS simulations.

An example of the numerical instabilities present in the late time flow fields of the Helium Bubble as simulated by Quirk and Karni is shown in Figure 6.11. The instabilities on the interface of the bubble, likely Kelvin-Helmholtz instabilities resulting from the shear across the interface, appear to be spurious. This may be caused by the lack of real or numerical viscosity in the simulation. The equivalent frame from the MB_CNS simulations is shown in Frame (i) of Figure 6.6. Although this figure shows an accurately stable representation of most of the bubble interface, the stiffness of the interface is under-estimated by the simulation through the region at the head of the heavy gas penetrating into the bubble.

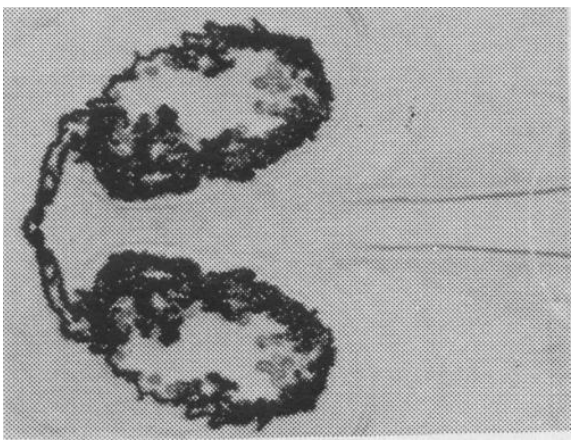

Figure 6.11: Frame from the simulation of Quirk and Karni [184] showing the spurious instabilities that were shown to form at late times.

CHAPTER 7

Simulations of the Drummond Tunnel Facility

Chapter 2 described how the flow development during the operation of a shock tunnel differs significantly from ideal representations, with the test time achieved being significantly shortened and noise being introduced into the flow. A number of non-ideal processes occur throughout the operation of the facility. Some of these processes are understood; however, empirical studies and numerical simulations have not provided an adequate method of predicting driver gas contamination, which has been identified as the primary cause of the termination of test time for high enthalpy operation in the T4 shock tunnel [221]. A method of predicting, and potentially preventing it, would be useful. Also, the quality of the test flow affects the use of experimental data obtained in these facilities to their use in investigating the conditions experienced in flight. The noise levels in shock tunnels are often an order of magnitude higher than those experienced in free flight and this affects many aspects of the experimental flow, including the onset of boundary layer transition.

Numerical simulations provide a method of investigating non-ideal processes occurring in a shock tunnel and predicting their effect on the test flow. In this chapter, simulations of the Drummond Tunnel facility, which was described in Section 2.2.1, are presented. These simulations are aimed at providing a better understanding of the flow in shock tunnels and the mechanisms by which the real flow deviates from the ideal.

The approach taken in these simulations is to model the flow development through the complete facility, from the driver section to the dump tank. An axisymmetric, body-fitted mesh is used to accurately represent the geometry of the facility and the simulations are provided with only the initial conditions reproduced from the experimental operation. These simulations then run from the initiation of the rupture of the primary diaphragm. Such large scale simulations have been made possible through the use of parallel computing, which is described in Chapters 4 and 5. By modelling the development of the flow through the entire facility, the

potential exists for predicting the test flow conditions, quality and duration. These simulations will aim, in particular, to identify the mechanisms that lead to driver gas contamination and the generation of the high levels of noise experienced in the test flow.

The simulations were performed using the multi-block CFD code, MB_CNS [114], which is based on a finite-volume formulation of the compressible Navier-Stokes equations. It has a shock-capturing capability through the use of a limited reconstruction scheme and an adaptive flux calculator that switches from AUSM to the Equilibrium Flux Method (EFM) where large velocity gradients are detected. The numerical techniques used in MB_CNS were described in Chapter 3.

The literature review, in Section 3.3, described the limitations in previously published numerical simulations of shock tunnel operation. These limitations are brought about by assumptions associated with only modelling part of a facility. The simulations performed in this study extend the modelling performed previously by removing these assumptions through the simulation of the complete facility. The models that are used in this study provide sufficient resolution to study each of the processes occurring in a shock tunnel under the influence of the surrounding flow.

Simulating the complete facility, from only the initial conditions complicates the simulation because all of the relevant processes must be modelled with sufficient resolution and any assumptions that are made must be valid. Processes that are accounted for include: the rupture mechanics of the primary diaphragm, turbulence in the boundary layers and the influence of the secondary diaphragm.

Chapter 6 described simulations of the interaction of shock waves with cylindrical bubbles of a light and a heavy gas and their comparison with experimental photographs. This was used as validation of the ability of MB_CNS in modelling transient flows involving instability in interfaces separating different gases. This validation is important to this chapter, since in the simulation of shock tunnel flows, these same unstable interfaces are present; however, there are no experimental photographs of the shock tunnel interfaces which can be compared directly with these simulations.

The simulations of the Drummond Tunnel are compared with the experimental results that were described in Section 2.2. The experiments provide traces of particular properties throughout the experiment. Support is given to the validity of the representation of the whole flow by the simulations being able to provide a close reproduction the traces from the experiments, This comparison of the experimental results with the numerical simulations is discussed in Section 7.2. The three cases that will be simulated are shown in Table 7.1.

Table 7.1: Outline of the different cases simulated in this chapter.

Driver and driven gas	Level of tailoring	Test attachment
Nitrogen driving Nitrogen	over-tailored	Mach 4 nozzle
Nitrogen driving Nitrogen	over-tailored	blanked end
Helium driving Nitrogen	roughly tailored	Mach 4 nozzle

Once the results of the simulations have been validated through comparison with the experimental results, the simulated shock tunnel flow is then used to investigate the flow in the real facility Section 7.3. The effect of mesh resolution on these processes is important and will be examined. The processes occurring in the operation of the shock tunnel are examined in separate discussions, but unlike previous studies, the simulations do not assume that each of these processes is occurring in isolation.

7.1 Simulation Setup

This section will describe the details of the numerical methods used that are specific to the simulation of the Drummond Tunnel. The MB_CNS Scriptit (`.sit`) files for the Drummond Tunnel simulations are provided in Appendix B. These files include the specification of the geometry and mesh, and the initial conditions. The special case files `mb_special_init.inc` and `mb_special_step.inc`, which are used to perform sections of codes specific to these simulations, are provided in Appendix C.

7.1.1 The Computational Mesh

The geometry of the Drummond Tunnel facility is discussed in Section 2.2.1. Body-fitted, finite-volume meshes covering the complete facility are used in the simulations. Each mesh covered the driver section, the length of the shock tube, and where applicable, the Mach 4 nozzle, test section and dump tank. The computational meshes are designed to capture the geometry of the facility as accurately as possible, given the constraints of the computational effort required and the body fitted arrangement of cells.

The meshes exploit axisymmetry, which allows a good representation of the cylindrical geometry of the shock tunnel, but constrains the gas to movement in the axial and radial directions, allowing no circumferential motion or gradients. This approximation affords considerable computational savings from fully three dimensional modelling.

The meshes are decomposed into 24 blocks in order to represent the geometry of the facility and to allow parallelisation of the solution. These blocks are, where possible, maintained with the same number of cells to aid parallel efficiency.

The main 59 mm diameter section of the driver section, with the spike running along its centerline, is included in the mesh. There is an additional driver volume around the pneumatic cylinder, which is also included. Since all of the gases used in the facility are modelled, there are no gas inflow boundaries in the facility.

The zero location for the geometry is set at a machined reference surface where the nozzle attaches to the shock tube section. This allowed the most reliable way of specifying the geometry, with the nozzle geometry being specified relative to this surface. A body-fitted mesh is used primarily so that the geometry of the nozzle could be modelled accurately. The geometry of the nozzle was known to within 0.1 mm from the nozzle design profile.

The full length of the dump tank behind the test section is modelled, along with its end wall. The dump tank extended outwards from the nozzle centreline in two opposite directions. This arrangement can be seen in Figure 2.12. This geometry is modelled as a supersonic outflow condition at the radius of the cylindrical part of the dump tank. The flow detail at this boundary was not considered to be important.

Two different computational meshes are used. The meshes only differ in that the geometry of the Mach 4 nozzle, test section and dump tank are attached to the end of the shock tube in one mesh, while the other has the end of the shock tube blanked off. The wall for the blanked end is sunk 4 mm into the shock tube from the zero location.

Figure 7.2 shows the coarse computational mesh used for the simulations with the Mach 4 nozzle attached. The extent of the mesh is shown at the top of the figure, showing the outline of each of the blocks. Inset 1 shows a closer view the mesh in the region around the nozzle and the test section. Inset 2 shows the mesh through the driver section. Inset 3 shows a closer view of the mesh through the nozzle contraction from inset 1.

With the original mesh being defined, three meshes were used for refinement studies. These three meshes were uniformly refined so that the number of cells across the radius of the shock tube was varied to from 40 cells for the coarse mesh (which is shown in Figure 7.2), 60 cells for the medium mesh and 80 cells for the fine mesh. These three meshes consisted of a total of 80,850 to 181,980 and 323,400 cells respectively. With the mesh being refined, the properties in the simulation should approach the experimentally recorded values with a second order convergence rate. Increasing the resolution of the mesh greatly increases the amount of computation

required in the solution. Ideally, doubling the resolution of the mesh in both axes requires an eight times increase in computational effort. The results of the mesh refinement studies are described in Section 7.3.

The valid resolution of the boundary layers is the primary limitation on the mesh. The mesh was refined towards the wall to focus cells in the boundary layers, whilst still leaving a significant number of cells in the core flow. The ability of the mesh to resolve the boundary layer is quantified in Section 7.3.2. The body-fitted mesh requires that the radial dimension is constant along the length of the tube and nozzle. Three frames of the same axial segment of the mesh through the shock tube is shown in Figure 7.1. The three frames are taken from the coarse (on the left) the medium (in the middle) and the fine resolution mesh (on the right). These frames also demonstrate the refinement of the mesh towards the wall.

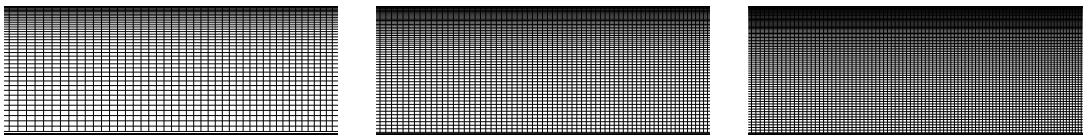

Figure 7.1: Segments of the computational meshes used. The coarse (left), medium (middle) and fine (right) resolution meshes are shown. Segments of the meshes across the 31.1 mm radius of the shock tube are shown, with $N_y = 40$, 60 and 80.

The mesh covers the length of the tube, so that the evolution of the flow along the length of the facility can be modelled. Over the downstream 0.5 m of the shock tube, the mesh is refined axially at towards the end of the shock tube in order to focus the computational effort in the region of the reflected shock interactions that take place. Simulations were run with double and quadruple axial resolution in order to investigate the effect of the axial resolution, and cell aspect ratio, on the evolution of the contact surface along the tube. In these simulations, the shape and characteristics of the contact surface remained essentially the same. From this it was concluded that the axial resolution of mesh was sufficiently converged for atleast the fine resolution mesh.

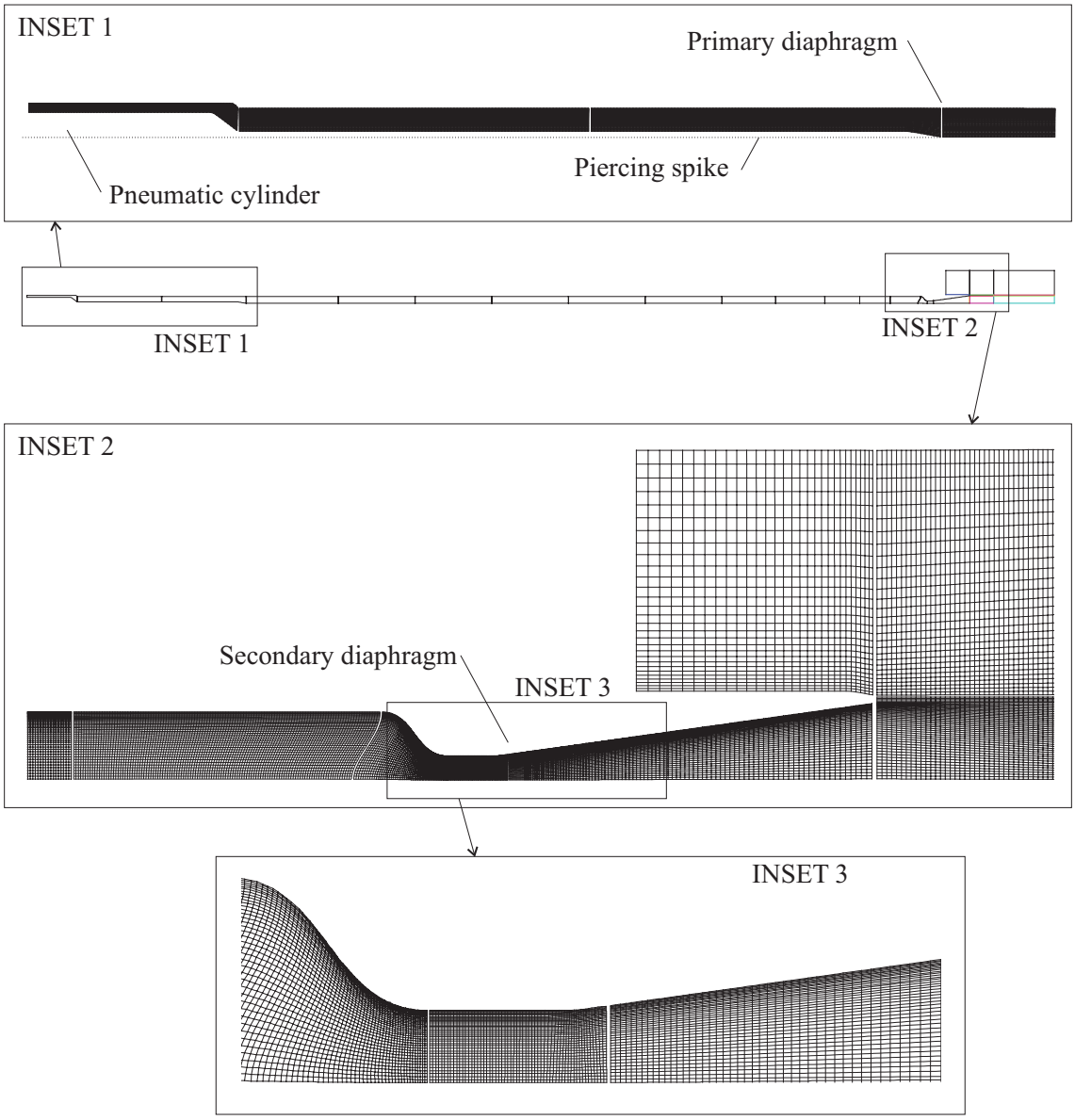

Figure 7.2: The computational mesh used for simulations of the Drummond Tunnel facility with the Mach 4 nozzle attached. The extent of the mesh is shown in the outline of the block boundaries. Three insets are shown: 1. the coarse mesh in the driver region, 2. the coarse mesh in the region around the nozzle, 3. an inset further into the nozzle region.

7.1.2 Initial Conditions

The type of gases used in the experiments, and their fill pressures, were given in Table 2.1. These were specified in the simulation as the initial conditions, along with the temperatures of the driver gases, which shall be further discussed in Section 7.2. As all of the facility is modelled (except for the outer part of the dump tank) all of the gases in the experiment are specified in the initial conditions and there are no inflow boundaries.

The wall temperature was assumed to be constant at 296 K, which was measured as an average ambient temperature in the laboratory containing the facility. It was assumed that the experiment was short enough that the heat transfer would not have been able to change the wall temperature. Simulations also were run using adiabatic walls; however, these simulations could not reproduce the heat transfer recorded in the experiments nearly as well as the constant temperature walls.

The ambient temperature of 296 K was also used as the initial temperature for the driven and dump tank gases and, in the case with the Nitrogen driver, the driver temperature. As was discussed in Section 2.2, the initial temperature of the driver gases were elevated due to the non-ideal filling process. This was demonstrated in the simulations, with shock speeds resulting from an ambient temperature driver gas being around five to ten percent too low for the corresponding post incident shock pressure.

In the real facility the test section is filled with very low pressure air; however, in the simulations Nitrogen is specified in this section. This is done for the simplicity of the gas models used and is not thought to be significant since the gas initially in the test section does not interact in any significant way in the facility and Nitrogen is the primary component of air.

7.1.3 Gas Models

The numerical methods used in MB_CNS require models representing the characteristics of the gases in the simulation. For the Helium driving Nitrogen case, the gas model assumed that the gas was a mixture of perfect gases. For the Nitrogen driving Nitrogen cases, a look-up table, based on the CEA tables [39] was used.

The properties of gas composed of multiple components (or species) are modelled by solving additional equations for conservation of each of these component gases. The properties of the gas mixture in each cell are calculated using mass fraction weighted averages of the component gas properties, as was described in Section 3.1.1. The ability to specify additional gas components, which may be the same as another

gas being used, allows regions of gas to be tagged and tracked over time through a simulation.

The peak temperatures in the stagnation region at the end of the shock tube remain below 1600 K. At these temperatures, vibrational and rotational modes of excitation of the molecules are yet to become significant. For the Helium driving Nitrogen case, the gas model assumes mixtures of perfect gases. Given the relatively low peak temperatures simulated, this is thought to be sufficient. For the Nitrogen driving Nitrogen case, a look-up table is used because it was available. Assuming an ideal gas for this case would have been sufficient as well.

7.1.4 Primary Diaphragm Rupture Model

The simulations being performed in this chapter involve the modelling of both the complete driver sections and the driven sections and, therefore, must account for the rupture of the primary diaphragm. The flow in the real facility is initiated by the rupture of a metal diaphragm separating the driver gas from the driven gas, as was described in Section 2.1.4. It was found through the development of the numerical models in this study that the flow in a shock tunnel could not be reproduced by a simulation of the complete facility, which did not include the effect of the diaphragm rupture process.

A model that assumes that the primary diaphragm opens as an iris is used in these simulations. Support for the appropriateness of this model is based on the experimental observations of Rothkopf and Low [197] and is discussed in detail in Section 2.1.4.

The total rupture time and final rupture diameter are specified in the file `mb_special_init.inc` (which is provided in Appendix C), as well as the block index of the block upstream of the diaphragm position; specifying a block index of `-1` turns off the diaphragm rupture model for the simulation.

The model works by assuming the linear variation in cross sectional area observed by Rothkopf and Low [197]. At each time step the radius of the open diaphragm is calculated based on this area. The cell occupying this radius of the shock tube was identified and was specified as the edge of the diaphragm. Measurements of used diaphragms, following experiments in the Drummond Tunnel, used to obtain an average diameter of the remaining diaphragm material of 57 mm. The radius of the opening diaphragm was increased at each time step, until this radius was reached, at which point it remained.

The diaphragm rupture model is implemented in the inter-block communication in MB_CNS. The block boundary condition would normally copy the data from the

first two rows of cells on the neighbouring block to the two rows of ghost cells used by the present block. This information passed from the neighboring block connects the flow in the two blocks. On the outside of the row of cells specified as being at the edge of the diaphragm, the two rows of the present block are transferred to the ghost cells in the same block instead. This has the effect of a reflected boundary condition, which is a wall between the blocks. This process is repeated for each side of the diaphragm boundary looking at the other side.

The reconstruction scheme used requires information from two neighboring rows of cells. This means that the communication at the block boundaries and, therefore, the diaphragm rupture model uses the two rows of cells on either side of the diaphragm boundary. Figure 7.3 shows the numerical arrangement of the rupture model, with the flow of information shown as arrows between the blocks on either side of the diaphragm, and the ghost cells used in the block communication.

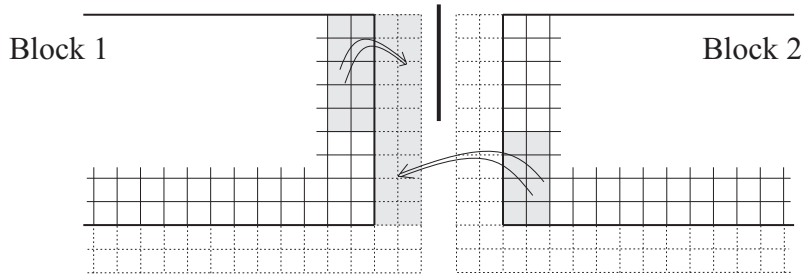

Figure 7.3: The numerical setup of the iris based diaphragm rupture model

This model does not attempt to account for the energy taken out of the flow by the deformation of the diaphragm but, given the relatively less ductile opening observed of aluminium diaphragms, this model is believed to be sufficient.

7.1.5 Secondary Diaphragm Rupture

The Drummond tunnel also has a thin secondary diaphragm, which separates the driver gas from the dump tank section. This diaphragm plays an important role in the operation of the shock tunnel. In these simulations the effect of the secondary diaphragm, in initially separating the shock tube from the test section, is modelled; however, it is assumed to be removed ideally.

The boundary between the computational blocks on either side of the diaphragm position are set at the position of the diaphragm. In the simulations, the effect of the intact diaphragm is modelled by assigning the connections between these two blocks as reflected boundary conditions, closing the connection. As there is no flow in the blocks on the downstream side of the diaphragm (those in the test section

and dump tank) they are assigned as inactive. The code to set up this arrangement was included in the file `mb_special_init.inc`.

When the incident shock arrives at the diaphragm, the pressure on the diaphragm rises rapidly and it bursts, allowing the shock processed driver gas to flow through the nozzle. The pressure of the downstream most cell in the block before the diaphragm boundary is monitored. When this pressure exceeds 100 kPa, the boundary conditions in the blocks up against the diaphragm position are connected to one another and the blocks downstream of the diaphragm position are activated with the initial conditions in the dump tank. The code to monitor the diaphragm pressure and to activate the diaphragm connection and the dump tank blocks was included in the file `mb_special_step.inc`.

The diaphragm burst pressure was varied from 100 kPa to 300 kPa to investigate the effect of the pressure chosen on the resulting flow. Investigations of the resulting supply pressure transducer traces and the test flow pitot pressure traces, showed no difference for this variation in the secondary diaphragm burst pressure. Given the rapid rise of the pressure at the secondary diaphragm resulting from the arrival of the shock, the diaphragm will burst within a very short period for any burst pressures of this order.

7.1.6 Turbulent Boundary Layer Modelling

The high densities and high stagnation enthalpies present in shock tunnel flows lead to considerable heat transfer to the shock tube walls. This heat transfer rate is determined by the behaviour of the turbulent boundary layers [213].

As was described in Section 3.1.1, the shock tunnel simulations in this thesis solve the Reynolds-Averaged Navier-Stokes equations. These equations use methods of statistically averaging the Navier Stokes equations over a time which is long compared to turbulent time scales, but short compared with the time scale of the mean motion.

The Baldwin-Lomax eddy viscosity model [11] is used to account for the effect of turbulent motions in the shock tube wall boundary layers. The Baldwin-Lomax model, is based on the model of Cebeci and Smith [40]. Eddy viscosity models are the simplest turbulence models in that they model turbulent stresses and fluxes by analogy to molecular stresses and fluxes [143]. The models add a component to the viscosity in the boundary layer to account for the effect of turbulent motions. The implementation of the Baldwin-Lomax model in MB_CNS is of the same form as that described by Craddock [53].

For the inner layer, the Baldwin-Lomax model is similar to the Cebeci-Smith model, but it differs significantly in the outer layer; Baldwin and Lomax modified the Cebeci-Smith Model by substituting new relations for conditions at the outer edge of the shear flow. The Baldwin-Lomax model is inexpensive and robust; however, it is an incomplete turbulence model and requires knowledge of the actual flow being modelled in the form of modifiable coefficients. In the original paper, the coefficients were obtained through comparison with the Cebeci and Smith model.

The variation of the outer layer coefficients, C_{cp} and C_{kleb}, is described extensively in the literature. The effect of favourable and adverse pressure gradients on C_{cp} and C_{kleb} is discussed in Granville [84]. The effect of Mach number on these coefficients is described in York and Knight [252]. Kim, Harloff and Sverdup [121] modified the coefficients for hypersonic flow conditions. He and Walker [92] obtained an expression for C_{cp} and showed that the Baldwin-Lomax model can account for the effects of variability in density. The variation of other parameters, such as the Clauser factor and the Coles wake factor are also described in the literature [84].

The outer layer coefficients, C_{cp} and C_{kleb}, were chosen from the relations provided in Kim, Harloff and Sverdup [121]. No instance of the variation of the Karman constant, κ, has been discussed in the literature; however, in these simulations, the use of the value of κ=0.4 provided in the original model [11] resulted in simulations which were definitely incorrect. The value of κ used in the original Baldwin-Lomax model was specified by Coles and Hirst [50], using experimental observations of a wide range of incompressible attached boundary layer flows. The reason for the requirement of varying κ in these simulations is not understood; however, it may be due to either compressibility effects or some aspect of the boundary layer specific to shock tube flows. Due to the effect that it has on the level of viscous attenuation, both the pressure behind the incident shock and the pressure behind the reflected shock depend strongly on the value of κ. The value of κ used in the simulations is selected through the comparison of the results with experimental results, in the context of the specific conditions used in the simulations. The selection of the Baldwin-Lomax coefficients used in the simulations is described further in Section 7.2. The coefficients used in the simulations are outlined in Table 7.2.

The turbulence model was activated for the block boundaries along the shock tube walls and through the nozzle. The flow in the driver section was assumed to be fully laminar since simulations using the turbulence model in the driver section were inaccurate. In the original paper describing the Baldwin-Lomax method a model for boundary layer transition was proposed. This transition model was used in these simulations

In these simulations, the boundary layer was assumed to be turbulent through-

out the nozzle profile. The boundary layer grows significantly in size through the divergent section of the nozzle. This expansion also promotes the rapid transition to turbulence in these boundary layers. The boundary layers on the nozzle of HEG are assumed to be turbulent through the majority of the divergent section [45]. The increased thickness of this boundary layer, due to the effect of turbulence, affects the effective profile that the core of the flow, that is the useful test flow, passes through. This has been found in the simulations to improve the accuracy of the simulated test low pitot pressure traces.

The implementation of the Baldwin-Lomax model through the separated flows in the region of the bifurcated foot of the reflected shock is questionable [122]. In order to investigate the effect that the model in this region has on the simulation, the turbulence model was switched off throughout the shock tunnel flow as the shock reflected from the end of the shock tube. This showed no effect on the traces produced, in comparison with simulations in which the turbulence model was left as active.

7.1.7 Recording Experimental Traces

Data can be recorded at the geometric location of data acquisition equipment in the real facility using *history cells*. These history cells are specified in the Scriptit files that are provided in Appendix B. The gas properties of interest at this location can then be extracted from this data. This simulated data can then be used in direct comparisons with the experiments. These comparisons provide the validation of the simulations in Section 7.2. The data can be recorded at a high sampling rate, providing the appearance of a continuous trace.

History cells were specified as the outer most cells in the rows corresponding to the axial locations of the supply pressure transducer and the heat flux gauge. For the experiments with the blanked end, history cells are assigned at the radial locations of the probes and the axial stations at which the rake is positioned. No attempt is made to model the effect of the rake and its sting on the flow. In the Nitrogen driver case with the nozzle attached, the pitot probe was positioned 14mm from the exit plane of the nozzle. For the Helium driver case it was positioned on the exit plane. This was accounted for by adjusting the history cells used for these two cases.

As an example, the trace of wall heat flux is calculated using these history cell traces. Knowing the temperatures at the outer row of cells, the gradient of temperature at the wall can be calculated by approximating a linear profile at the wall. For the comparison of the simulation results with the experimentally recorded

heat transfer data, the heat flux to the wall (Q_w) was calculated using:

$$Q_w \approx k\frac{\Delta T}{\Delta y} = k\frac{T_1 - T_w}{y_1 - y_w} \tag{7.1}$$

Where k is the thermal conductivity of the gas near the wall, T is the temperature and y is the radial position. The subscript 1 denotes the properties at the centre of the first row of cells from the wall and the subscript w denotes the properties at the wall.

7.2 Validation of the Simulations using the Experimental Results

The simulations that were run in this chapter reproduced the initial conditions of three sets of experiments that were conducted in the Drummond Tunnel facility. The data sets that were recorded during these experiments, were discussed in Section 2.2.4. Simulated data traces were also recorded throughout the simulations and, in this section, this simulated data will be compared with the corresponding experimental data, to provide a validation for the simulations. The simulations are given only the initial conditions from the experimental operation. This means that if the simulations can provide a close reproduction of the traces from the experiments, then it is likely that they are producing a valid representation of the flow through the complete shock tunnel.

The case was initially modelled assuming laminar boundary layers. The accuracy of the simulations were improved by accounting for the effect of turbulence in the shock tube wall boundary layers with the Baldwin-Lomax eddy viscosity model which was described in Section 7.1.6. This is an algebraic turbulence model and it has coefficients that are modifiable for particular flow conditions. The variation of the two outer layer coefficients C_{cp}, C_{kleb} is described widely in the literature [121, 84, 92]. The values of these coefficients were obtained from Kim and Harloff [121]. Kim and Harloff provide graphs of the variation of these two coefficients with the Mach number, which were used to obtain the values used for the cases simulated, based on the incident shock Mach number. The variation of C_{cp} and C_{kleb} was found to not have a significant effect on the attenuation of the incident shock. The effect of these values on shock tunnel flows is primarily seen in the evolution of the contact surface along the shock tube, which will be discussed in Section 7.3.6. In order to produce accurate simulations, the model was also modified through the Karman constant (in the inner layer of the model). In the simulations, the value that would reproduce the experimental results was found to be between 0.15 and 0.20. The exact value was found by modifying this coefficient in conjunction with the driver temperature; this process will be discussed in detail. The results in this section will first be presented for the simulations assuming laminar boundary layers and then for the simulations incorporating the turbulence model. The original Baldwin-Lomax coefficients [11] provided results that are indicative of more than a 10% over estimation of the attenuation of the incident shock.

The experimental gas fill pressures were known and were not varied in the simulations, but the initial gas temperatures in the experiments were not known exactly. The investigation of the filling process, described in Section 2.2, indicated that the

driver temperatures were elevated to between 30^{o}C and 40^{o}C. It was also found that the driver temperature can vary between experimental shots depending on the rate of filling and the delay between filling and the firing of the shot.

The driven gas temperatures were assumed to be ambient, and equal to the wall temperature. The driven gases were not filled to the same high pressure as the driver gases and, therefore, were not believed to be subject to the same heating effect. Any variation in this temperature would only have been due to variation in the ambient temperature in the laboratory.

The implementation of the primary diaphragm rupture model was described in Section 7.1.4. All of the parameters used in the model were characteristics of the diaphragms used or were obtained from the literature. The primary diaphragm was assumed to rupture with a linear profile of opened cross-sectional area versus time. This assumption, and the total rupture time was obtained from Rothkopf and Low [197] and was discussed in Sections 2.1.4. The resulting diameter of the ruptured diaphragm was obtained from measurements of used diaphragms taken from the facility following experiments. The effect of the diaphragm rupture model will be discussed in Section 7.3.1.

The initial conditions used in the simulations reproduced the facilities operating conditions that were given in Table 2.1. Along with these conditions, the modifiable characteristics of the simulation that most closely reproduced the flow are given in Table 7.2.

Table 7.2: Modifiable simulation parameters that best reproduced the experimental results.

Driver Temperature	305K (He) and 310K (N_2)
Driven Temperature	296K
Wall Temperature	296K
Baldwin-Lomax (Turbulence Model):	
κ	0.18
C_{cp}	1.74
C_{kleb}	0.47
Primary Diaphragm Rupture Model:	
rupture time	200 μs
rupture diameter	57 μs

The values used for the driver gas temperature and the Baldwin-Lomax model

coefficient κ were obtained by varying them and comparing a characteristic value obtained from the simulation to the corresponding value obtained in the experiment. This characteristic value was the pressure behind the incident shock for the Helium driver case and the pressure behind the reflected shock for the Nitrogen driver cases. An increase in driver temperature results in an increase in the initial shock strength. On the other hand, an increase in κ resulted in an increase in the level of the turbulent contribution to the viscosity in the inner layer of the simulated boundary layer ($\mu_{t(inner)}$). This increased the viscous attenuation of the shock as it progressed along the tube and, therefore, caused a decrease in the pressure behind the shock as it passed the supply pressure transducer.

The speed of the incident shock was measured using the time of flight between the heat flux gauge and the supply pressure transducer (which were separated by 217 mm). The shock speed was not noted during the modification of the two simulation parameters which affected the shock pressures and it was found that the shock strength required to reproduce the post shock pressure would reproduce the shock speed. Support is given to the validity and potential predictive capacity of the simulations in that the simulations, being calibrated for particular simulation values, can predict other values.

Various combinations of initial conditions and levels of viscous attenuation could be used to reproduce the desired shock characteristics; however, there are other characteristics of the simulation which, if measured in combination with these, could only be reproduced by a single set of simulation parameters. The other parameters used in the comparison were: the delay between the arrival of the incident shock and the reflected shock at the supply pressure transducer, and the time of the arrival of the reflection of the tailoring wave. The delay before the arrival of the reflection of the tailoring wave is dependent on the interaction of the reflected shock and the contact surface and the conditions in the stagnated region at the end of the shock tube. Also the behaviour of the driver gas following its interaction with the reflected shock is used as a qualitative constraint. These constraints led to the single set of parameters which were used in the simulations.

This set of parameters was the same for the two different operating conditions, apart from the driver temperatures (which were, in all probability, different from one another in the experiments). This is an important result for the validity of the simulations: even though the operating conditions were very different, and given the possible ranges of the parameters, the parameters that reproduced experimental traces the most closely were the same for the three cases.

In this section, the simulations and the experimental traces used the same time scale. The times are aligned at the arrival of the incident shock at the supply

pressure transducer. Table 7.3 shows the time offsets between the times used in this section, and the time from the initiation of the primary diaphragm rupture, which is used in Section 7.3.

Table 7.3: Time offsets relative to the initiation of diaphragm rupture used in this section.

Driver and driven gas	Attachment	Time offset used
Nitrogen driving Nitrogen	Mach 4 nozzle	+3.716 ms
Nitrogen driving Nitrogen	blanked end	+3.716 ms
Helium driving Nitrogen	Mach 4 nozzle	+2.528 ms

Two of the experimental transducers, the supply pressure PCB transducer and the heat flux gauge, are mounted in the wall of the shock tube. This means that the transducers measure the flow inside the boundary layer, and as a result, any comparison of simulation data with them is sensitive to the accurate modelling of the boundary layer. This is shown to be particularly important for measurements recorded during the passage of the bifurcated foot of the reflected shock.

The simulation results presented in the following comparisons use the fine resolution mesh. In addition, the fine mesh resolution simulations are further discussed in Section 7.3. The modifiable parameters used in the simulations were chosen so that the result would converge for the fine resolution mesh simulations. As the mesh was refined from the coarse and medium meshes to the fine mesh, there were no significant deviations from the traces presented in this section. This was true for the three cases that were modelled.

The experimentally recorded traces are described in Section 2.2.4. The features of these traces are described in that section and so the discussion of the traces in this section will be restricted to the comparison of the simulated and experimental traces.

Problems with the Simulations

The accuracy of the simulations depends on careful comparison with the experiments. The simulations, in their current form, cannot be used in a predictive way with any certainty. There were some simulations, which through small changes in the conditions, or the modifiable parameters, would produce entirely incorrect traces.

The most obvious of these is evident in the supply pressure transducer traces from the Helium driving Nitrogen case. This case used roughly tailored operating

conditions, and the axial location at which the reflected shock reaches the contact surface is between the supply pressure transducer and the heat flux gauge (which are separated by only 217 mm). This means that any inaccuracies in either the initial conditions, or the level of viscous attenuation along the tube can result in a slightly incorrect level of tailoring. This can result in incorrect movement of the contact surface following the interaction and as a result, the contact surface can impinge on the supply pressure transducer. Driver gas coming into contact with the transducer can produce large variations in the trace recorded by the transducer.

These types of dips were also evident in the over-tailored Nitrogen driving Nitrogen case. This resulted from a slightly incorrect representation of the over-tailored interaction for the same reasons described in the Helium driving Nitrogen case.

Amongst the problems observed in the traces were the entirely incorrect traces produced in simulations using the original value of κ used in the original Baldwin-Lomax model [11]. This variation may result from a coding problem in the implementation of the model in the code; however, the implementation of the model has been used successfully in the past and careful examination of the code did not find such an error. The simulations were continued using the modified value of κ since these values, through calibration, could be used to reproduce the traces.

7.2.1 Over-Tailored Operation: Nitrogen Driving Nitrogen

This section will compare the traces obtained from the simulations of the Nitrogen driving Nitrogen cases with the equivalent experimental traces. The two cases are with the nozzle attached and with the end of the shock tube blanked off where the nozzle would be attached. Nitrogen is used as the driver gas and the driven gas, using the operating conditions given in Table 2.1; the resulting operation is over-tailored. The other parameters used in this simulation were outlined in Table 7.2.

The convergence study was produced similar results for the two Nitrogen driving Nitrogen cases. As a result of this, the convergence study is shown for the combined case. This similarity in the convergence would be expected given the similarity of the operating conditions. The x-t diagram that is discussed for these cases is also assumed to be the same for the two cases. Separate traces from the experiments and simulations using the blanked end and the Mach 4 nozzle will be considered.

Grid Convergence

As the mesh is refined, the solution should converge towards the experimental flow field. The convergence of the solution, with refinement of the mesh, is examined using characteristic flow values that can be used for reference. The most distinct measurement values were used for comparison: shock speeds for both cases, and the incident shock pressure for the Helium driver case and the reflected shock pressure for the Nitrogen driver case.

Various levels of approximation interact throughout the simulation affecting the level of errors present in the solution. The unlimited reconstruction scheme is 3rd order accurate; however, the use of limiters affects the convergence, as do the boundary conditions, the solution of viscous fluxes, the turbulence model and the selective refinement of the mesh towards the wall. This means that no definite order of convergence is expected to be observed.

Given the convergence of the solution with mesh resolution, the parameters used in the simulations were chosen such that the values obtained from the fine mesh resolution simulation converged to the experimental values. This was believed to provide the best representation of the flow possible with these meshes. This means that the coarse and medium meshes will under predict these values; however, the fine mesh is used in the discussion and analysis. It will be seen in Section 7.3 that the processes occurring in the flow are resolved sufficiently, for even the coarse mesh simulations.

The convergence plots for the Nitrogen driving Nitrogen cases are shown in Figure 7.4. The convergence based on the shock speed is shown on the left and the

convergence based on the pressure behind the incident shock shown on the right. The values obtained from the simulation are plotted versus the inverse of the number of cells used across the radius of the shock tube. This is done so that a solution with an infinite number of cells is at the left edge of the plot. The experimentally measured value is shown as the dashed line across the plots. The convergence of the solution towards the experimental values are evident. The value for the fine mesh (which is the left most point) is slightly lower than the experimental values and would be expected to converge closer to the experimental value for even finer mesh solutions.

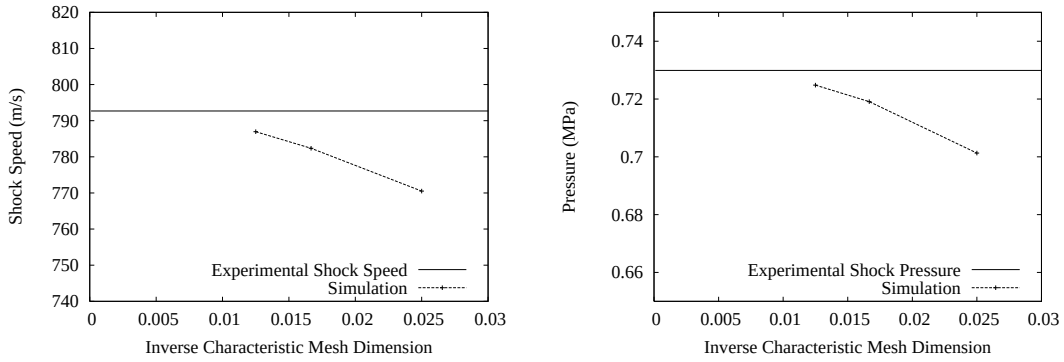

Figure 7.4: Grid convergence for the blanked end Nitrogen driving Nitrogen case simulations. The grid convergence based on the shock speed is shown on the left and the convergence based on the pressure behind the reflected shock is shown on the right.

x-t Diagram

An x-t diagram can be constructed from the MB_CNS simulations by sampling the centreline positions of the shock and the contact surface through time. This x-t diagram, for the over-tailored case, is shown in Figure 7.5. This x-t diagram is essentially the same for the two over-tailored cases simulated (the case with the Mach 4 nozzle and the case with the blanked shock tube end). The x-t diagram shown in Figure 7.5 is taken from the blanked end case. The main difference between the two cases, the expansion of the gas through the nozzle, does not have a significant effect on this diagram. The extent of the contact surface is defined as being between the most upstream and downstream part of the contact surface, not inside the boundary layer, and not detached from the main contact surface.

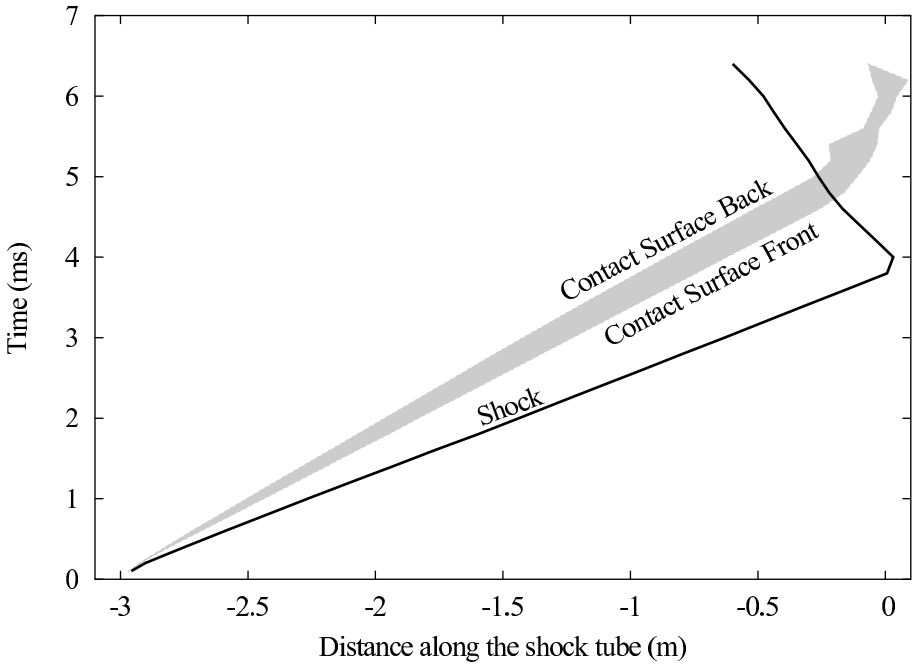

Figure 7.5: MB_CNS simulated x-t diagram for the Nitrogen driving Nitrogen with the blanked shock tube end case. The **turbulent** simulation was used.

In the x-t diagram, early in time the trajectories of the incident shock (solid line) and contact surface (grey region) can be seen moving towards the downstream end of the shock tube. The incident shock is shown to reflect from the downstream end of the shock tube and travel back upstream. The reflected shock interacts with the contact surface, sending both a transmitted wave upstream and a reflected wave back downstream. The wave reflected downstream, which is not shown, is the over-tailoring wave. This wave ends the test time.

The expansion fan emanating from the primary diaphragm position is not shown in this x-t diagram as its arrival at the end of the shock tube is late in time and

so it is not important to the discussion with the operating conditions used in these cases.

Wave Speed Diagram

The points sampled from the blanked end case simulations and used in Figure 7.5, were used to obtain the wave speed diagram shown in Figure 7.6. As with the x-t diagram, the shock is shown as the solid black line and the contact surface is shown as the grey region. This figure demonstrates the initial strong acceleration of the shock as the waves that form during the opening of the primary diaphragm coalesce. Following this, the gradual attenuation of the shock, caused by the viscous boundary layers, is evident.

The front of the contact surface moves along the shock tube with approximately constant velocity, although a slight deceleration is evident. The effect of the diaphragm rupture model on the initial deformation of the contact surface is evident. The stability of the contact surface is shown, in that as the contact surface moves along the shock tube, the front and the back of the contact surface approach the same speed.

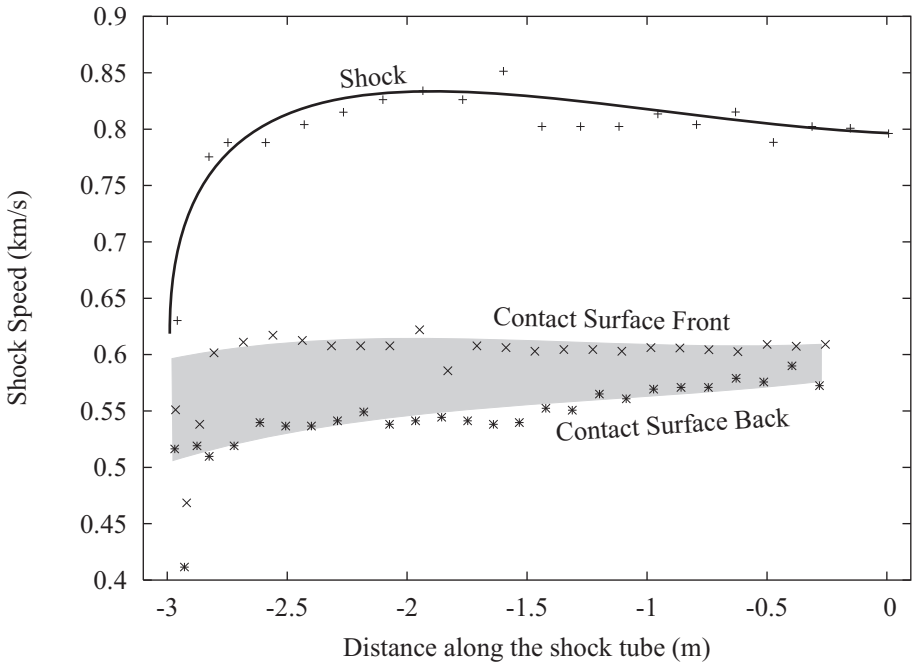

Figure 7.6: Evolution of waves speeds along the tube from the **turbulent** MB_CNS simulations for the Nitrogen driving Nitrogen case. The speed of the shock and the front and the back of the contact surface are shown.

The figure actually shows the variation of du/ds along the tube; however, this leads to the acceleration by considering the relation: $a = du/dt = du/ds \cdot ds/dt =$

$du/ds \cdot v$. This means that, since the sign of the velocity is always positive, the sign of du/dt is the same as the sign of du/ds.

The delay in the arrival time of the incident shock and contact surface at the heat flux gauge and the pressure transducer, being separated by 217 mm, can be used to identify the wave speeds, both in the experiment and the simulations.

For the blanked end case, the experimentally measured shock speed was 792.7 m/s. Using the fine mesh simulation, the shock speed predicted by the laminar simulation was 893.1 m/s (12.7% high) and the turbulent simulation was 787.0 m/s (0.7% high). This shows a 12.0% improvement associated with the implementation of the Baldwin-Lomax model. The experimental shock speed from the Mach 4 nozzle case could not be determined with any accuracy due to the high level of noise in the experimental trace.

Supply Pressure Transducer: Blanked End Case

1. Laminar Simulation (Figure 7.7)
The laminar simulation significantly over estimates both the incident and the reflected shock pressures. These results are indicative that there has not been enough viscous attenuation of the shock. In the experimental trace, the rise due to the arrival of the reflected shock shows a curved profile, resulting from the interaction of the reflected shock with the boundary layer. This effect is not reproduced in this simulation. The tailoring waves arrive late, leading to an over estimate of the steady period. In the laminar simulations, the arrival of tailoring waves at the transducer are too sharply defined, appearing on the trace as steps, rather than the more tapered rise observed in the experimental trace. The final pressure, following the passage of the contact surface is too high.

2. Turbulent Simulation (Figure 7.8)
With the calibrated turbulence model, the pressure behind the incident shock and the arrival time of the reflected shock are reproduced accurately. The most significant difference between the turbulent simulation and experimental traces is at the arrival of the reflected shock. The foot of the reflected shock has bifurcated due to its interaction with the incident boundary layers. The traces resulting from this interaction are discussed in detail in Section 7.3.4. The arrival times of the tailoring waves are accurately reproduced. The pressure rises are more tapered and are significantly more like the experimental trace than the laminar simulation; however, the tailoring waves are still sharper and more defined than in the experimental trace. This is believed to be due to limitations in the simulation of the contact surface, which are described in detail in Section 7.3.6. The final pressure, as the driver gas passes the transducer is accurately reproduced and the increased level of noise associated with the arrival of driver gas is also reproduced.

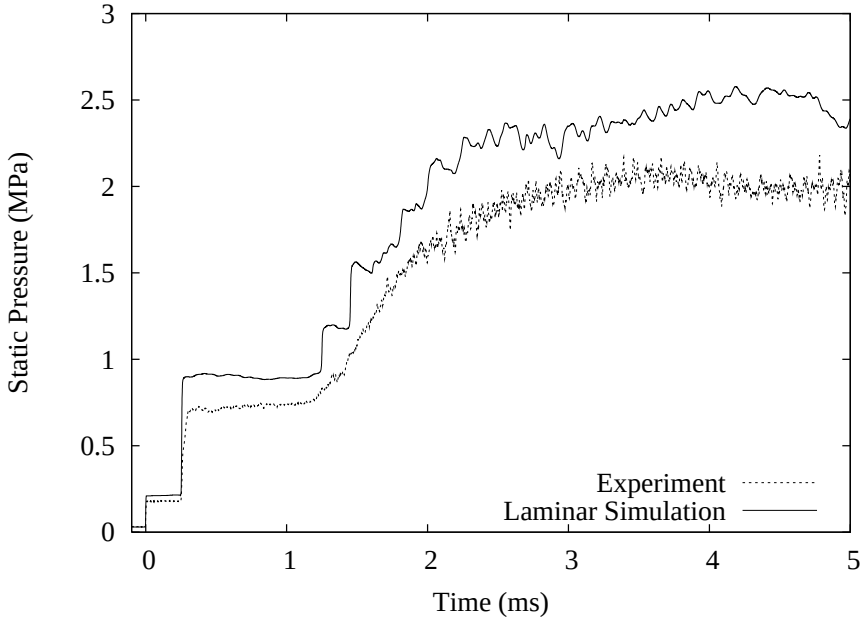

Figure 7.7: Static pressure recorded by the supply pressure PCB transducer during the Helium driver case for the **laminar** simulation compared to the experiment.

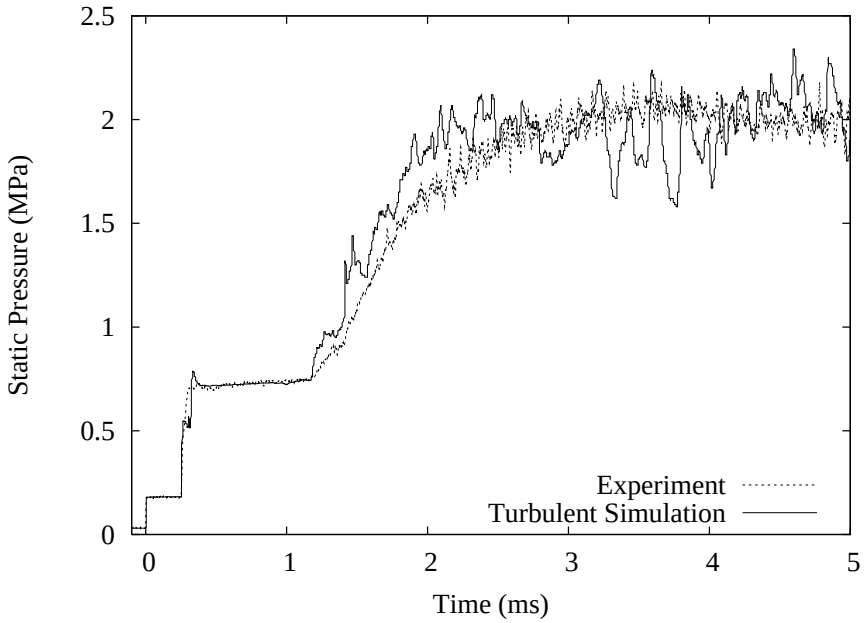

Figure 7.8: Static pressure recorded by the supply pressure PCB transducer during the Helium driver case for the **turbulent** simulation compared to the experiment.

Supply Pressure Transducer: Mach 4 Nozzle Case

1. Laminar Simulation (Figure 7.9)
As with the blanked end case, the laminar simulation significantly over estimates both the incident and the reflected shock pressures. With the Mach 4 nozzle case, the interaction of the reflected shock with the boundary layer is more significant than with the blanked end case. The effect of this interaction is not evident in the laminar simulations, with the trace showing a sharply stepped profile. The tailoring waves arrive late, leading to an over estimate of the steady period. In the laminar simulations, the arrival of tailoring waves at the transducer are too sharply defined, appearing on the trace as steps and the final pressure is too high.

2. Turbulent Simulation (Figure 7.10)
With the calibrated turbulence model, the pressure behind the incident shock and the arrival time of the reflected shock are reproduced accurately. In the simulation, the passage of this shock foot past the pressure gauge has an oscillating stepped profile, whereas the experimental trace is curved following a small step. The pressures in front of, and behind, this interaction are the same for the simulation and experiment, and the gradual rise in the experiment and the stepped profile in the simulation both take the same amount of time to pass the transducer. The foot of the reflected shock has bifurcated due to its interaction with the incident boundary layers. The differences in these traces are discussed in Section 7.3.4. The pressure behind the reflected shock in the simulation is within the range of fluctuations the pressure in the experimental trace, or slightly below it. The arrival time of the tailoring wave is accurately reproduced. As with the blanked end case, these tailoring waves are stronger in the simulations than in the experiment. The trend in pressure as these waves arrive at the transducer is predicted accurately; however, small deviations from the experimental trace are evident. The final pressure, as the driver gas passes the transducer is accurately reproduced and the increased level of noise associated with the arrival of driver gas is also reproduced.

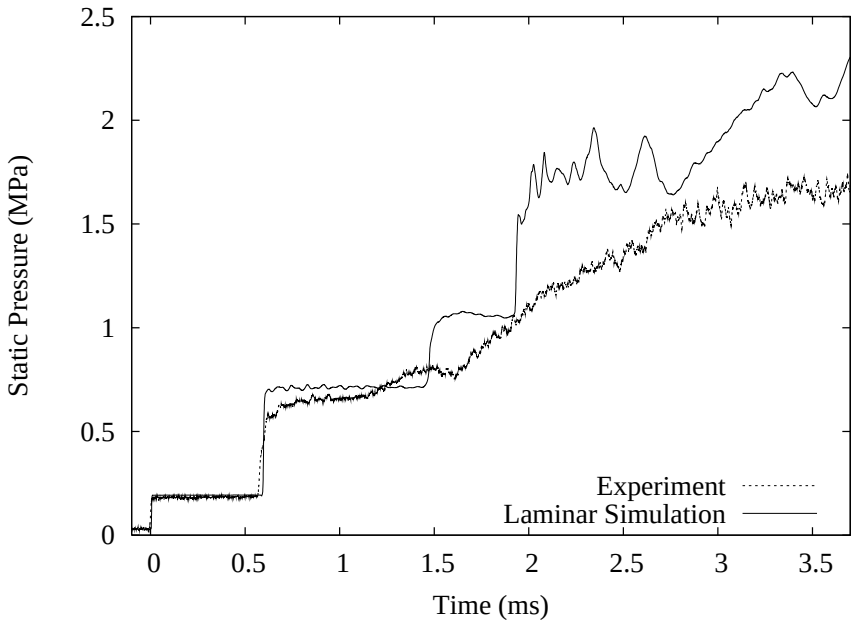

Figure 7.9: Static pressure recorded by the supply pressure PCB transducer during the Helium driver case for the **laminar** simulation compared to the experiment.

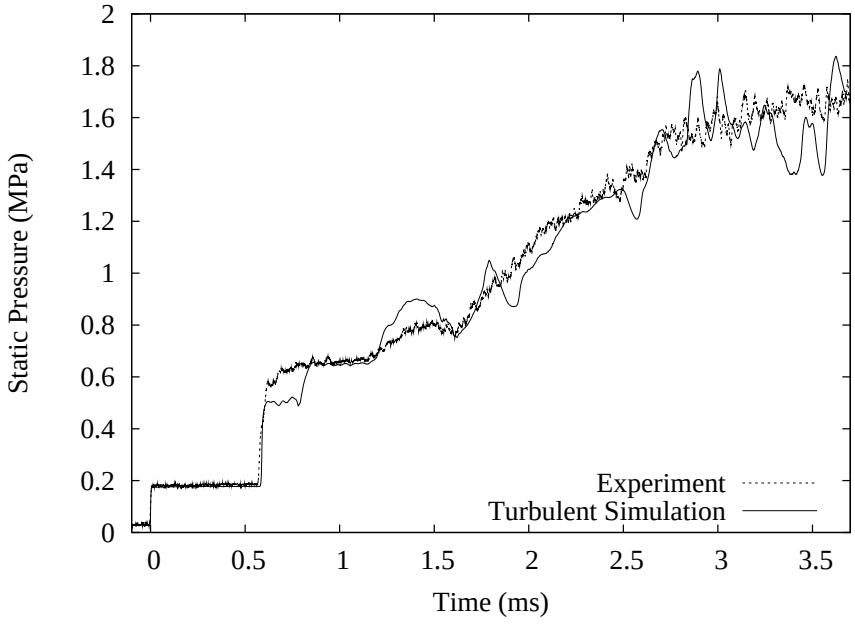

Figure 7.10: Static pressure recorded by the supply pressure PCB transducer during the Helium driver case for the **turbulent** simulation compared to the experiment.

Heat Flux Gauge: Blanked End Case

The heat flux traces are only compared for the blanked end case, as the measurements taken during the experimental with the Mach 4 nozzle suffered from excessive experimental noise. The magnitude of the heat flux was not known during the experiments. This means that no quantitative comparison of heat fluxes could be used, and the level of the experimental heat flux in the figures was aligned with the simulated magnitudes. The simulated heat flux was calculated using a linear relation given in Section 7.1.7.

1. Laminar Simulation (Figure 7.11)
The initial jump in heat flux is caused by the passage of the incident shock. It can be seen in the experimental trace that the heat flux to the wall from the gas in the region behind the incident shock is steady; however, in the simulation, this heat flux decreases over time (with distance from the shock). This decrease in heat flux is caused by the variation in heat flux through the growing boundary layer which is increasing in thickness with distance from the shock. The second rise in heat flux is the arrival of the reflected shock. The heat flux remains high in the region behind the reflected shock. The sharp decrease in heat flux is caused by the arrival of the contact surface following its interaction with the reflected shock. For the remainder of the trace, the waves are dominated by the complex interaction between the reflected shock and the contact surface and the resulting pseudo-shock train.

2. Turbulent Simulation (Figure 7.12)
The only significant difference between the laminar and turbulent heat flux traces is during the passage of the bifurcated foot of the reflected shock. This appears as the dip in heat flux immediately following the arrival of the reflected shock. This dip is not evident in the experimental trace, indicating that the temperature of the gas in the shock foot is not as low as it is modelled in the simulation. The other features in the trace are essentially the same as were described for the laminar simulation comparison. Given the difficulty in reproducing these traces, it is believed that both the laminar and turbulent simulations provide a reasonable reproduction of the experimental heat flux over time.

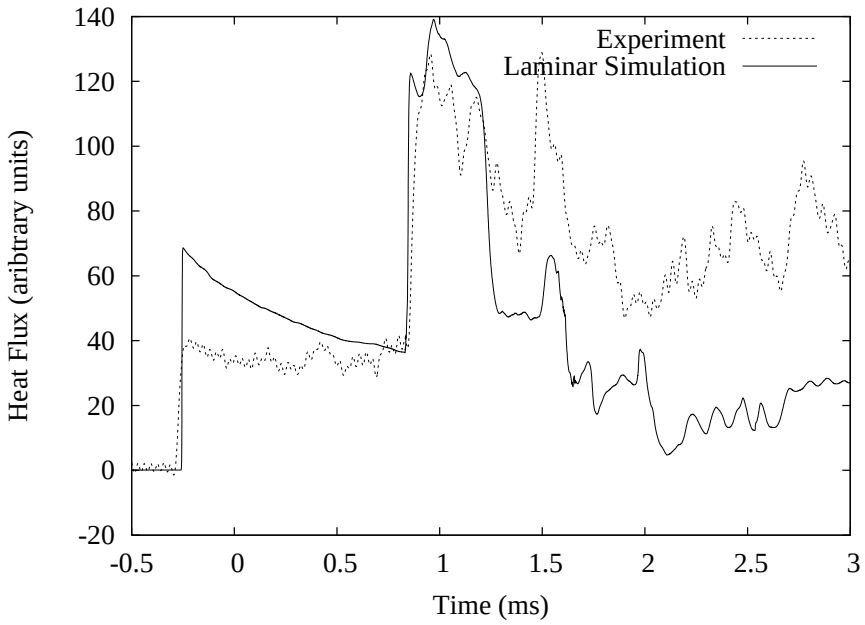

Figure 7.11: Heat flux recorded by the thin film heat flux gauge during the Helium driver case for the **laminar** simulation compared to the experiment.

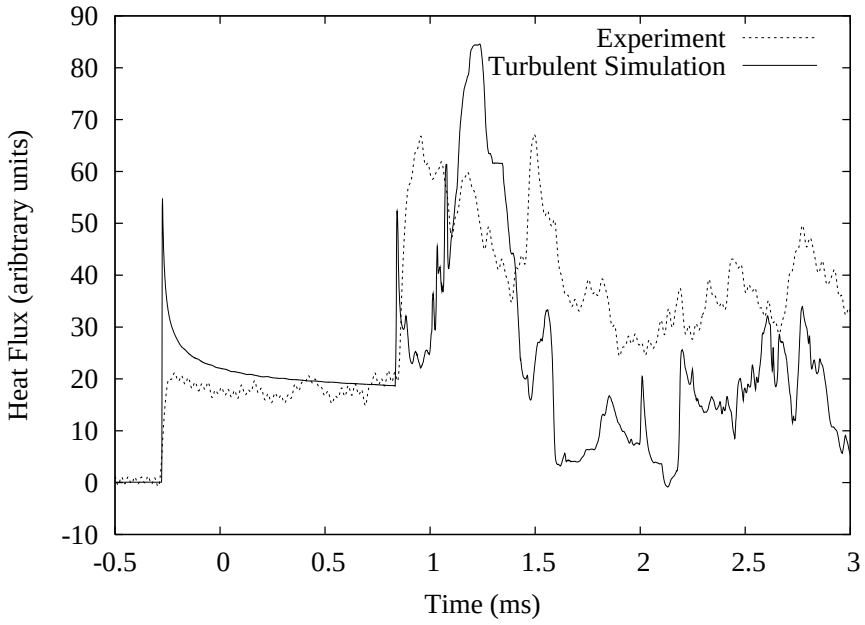

Figure 7.12: Heat flux recorded by the thin film heat flux gauge during the Helium driver case for the **turbulent** simulation compared to the experiment.

Shock Tube Heat Flux Rake: Blanked End Case

The rake of heat flux probes described in Section 2.2.3 was used to detect the arrival of the contact surface at stations at 524 mm and 1015 mm from the blanked end of the tube. The diaphragm rupture model had a significant effect on the shape of the contact surface in the simulations. For this reason, the results for simulations with, and without, the diaphragm rupture model will be discussed.

1. Without the Diaphragm Rupture Model (Figure 7.13)
The profile measured at the 1015 mm position is shown on the left and the at the 524 mm position is shown on the right. The experimentally measured profile is shown in grey and the simulated profile, without the diaphragm rupture model, is outlined in black. The general shape of the contact surface from this simulation is similar to the experimental shape, but the level of mixing at the contact surface is significantly lower than is evident in the experimental measurements. The real contact surface is expected to be turbulent, which would promote mixing and diffusion at the interface, and may account for this difference.

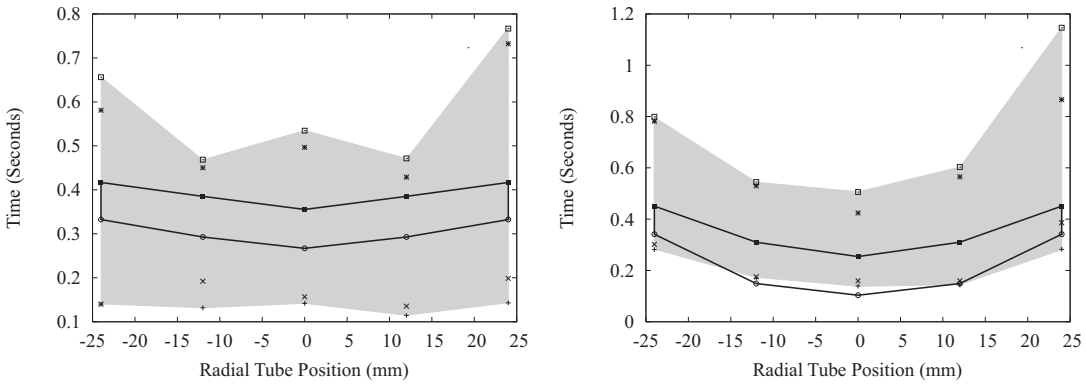

Figure 7.13: Contours of driver gas mass fraction, showing the contact surface shape at the 1015mm station (on the left) and the 524mm station (on the right). The diaphragm rupture model was not included in the simulation. The experimentally measured contact surface shape is shaded in grey and the simulated contact surface shape is outlined in black.

2. With the Diaphragm Rupture Model (Figure 7.14)
The profile measured at the 1015 mm position is shown on the left and the at the 524 mm position is shown on the right. Again, the experimentally measured profile is shown in grey and the simulated profile, with the diaphragm rupture model, is outlined in black. The simulation with the diaphragm rupture model does not reproduce the contact surface shape from the experiment correctly. An important consideration is that the mixing length of the contact surface is about the same between the two. The circulation of material caused by the diaphragm rupture model

has not resulted in full mixing at the interface. Also it is known that the contact surface is turbulent; this turbulence would accelerate the mixing at the interface and is not modelled in the simulations. The result is that the mixing in the simulated profile is predominantly along the centreline of the shock tube, whereas in the experiment it is across the full radius. The characteristics of the contact surface are discussed in more detail in Section 7.3.6. These issues may have an effect on the interaction of the contact surface with the reflected shock, discussed in Section 7.3.7, although it is known that the interaction process is modelled sufficiently to accurately predict the arrival of driver gas in the test flow from the Nitrogen driving Nitrogen (Mach 4 nozzle) case. This comparison is shown in Figure 7.16.

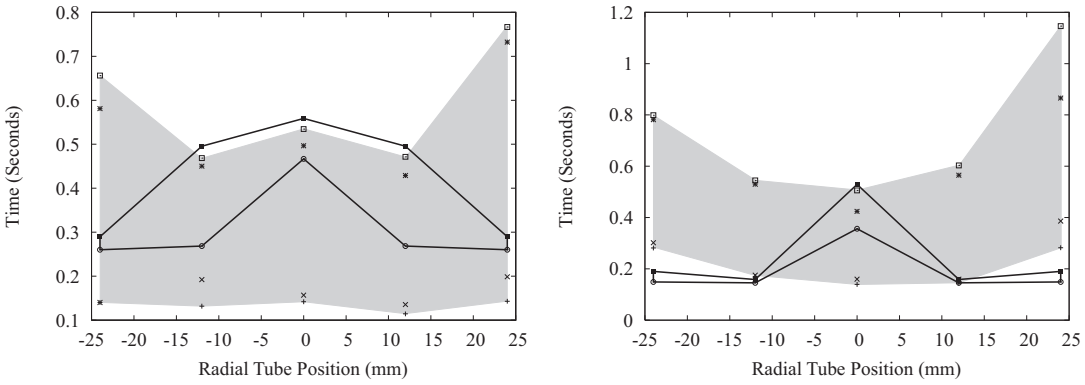

Figure 7.14: Contours of driver gas mass fraction, showing the contact surface shape at the 1015mm station (on the left) and the 524mm station (on the right). The diaphragm rupture model was included in the simulation. The experimentally measured contact surface shape is shaded in grey and the simulated contact surface shape is outlined in black.

Nozzle Exit Pitot Pressure: Mach 4 Nozzle Case

Figure 7.15 shows the experimental trace of pitot pressure measured at the centreline of the nozzle exit plane compared to the trace from the turbulent simulation. In addition to accounting for the effect of turbulence along the length of the shock tube, the use of the Baldwin-Lomax model on the nozzle walls improved the accuracy of the simulations. In the trace shown, the duration over which these waves arrive at the probe is accurately reproduced. The experimental trace has been filtered, removing high frequency fluctuations; this includes much of the magnitude of the startup waves. In the unfiltered experimental trace, the magnitude of the startup waves was of the same magnitude as the simulated waves. The steady test time and the magnitude of the pitot pressure during the test time are accurately reproduced. The over-tailoring wave causes the increase in pressure that ends the test time. This wave is too strong in the simulated trace; however, the general trend in the pitot pressure rise, following the initial arrival of the tailoring wave is close to that seen in the experimental trace. The final pitot pressure, as driver gas flows through the nozzle, is reproduced accurately. The results from the laminar simulations are not shown for the test flow properties as they are signficantly different.

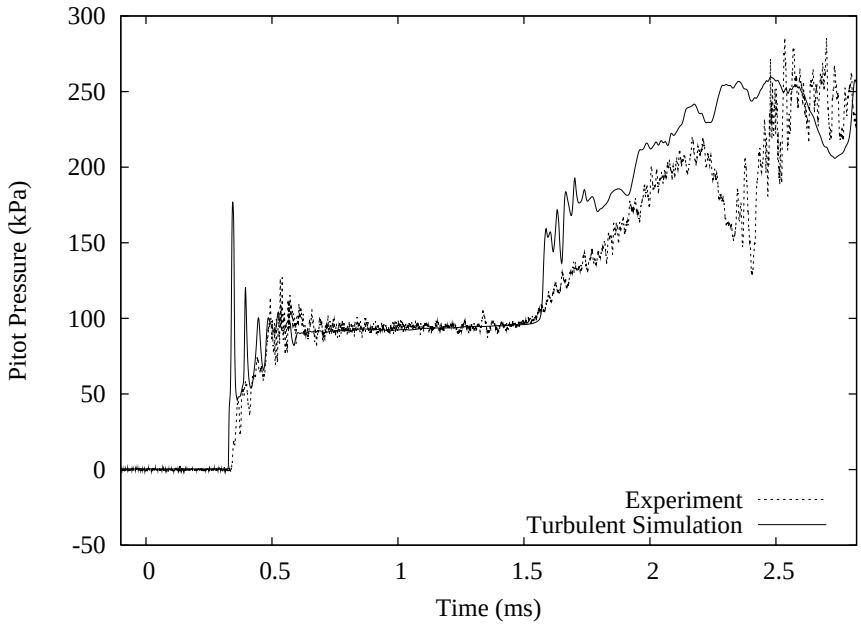

Figure 7.15: Comparison of the simulated and experimental pitot pressure at the nozzle exit centreline. The simulated trace is from the **turbulent** simulation of the Nitrogen driving Nitrogen case.

Nozzle Exit Stagnation Temperature: Mach 4 Nozzle Case

Figure 7.16 shows the experimental trace of stagnation temperature measured 14 mm from the centreline of the nozzle exit plane compared to the trace from the turbulent simulation. The change in stagnation temperature resulting from the arrival of driver gas is evident. This traces shows that the turbulent simulations accurately reproduce the time of the arrival of driver gas in the test flow for the over-tailored case. The trace shows that the simulations reproduce the stagnation temperature with reasonable accuracy, both before and after the arrival of the driver gas. This driver gas arrives in the test flow prematurely, because of the acceleration of driver gas resulting from the interaction of the contact surface with the reflected shock. This interaction is discussed in more detail in Section 7.3.7.

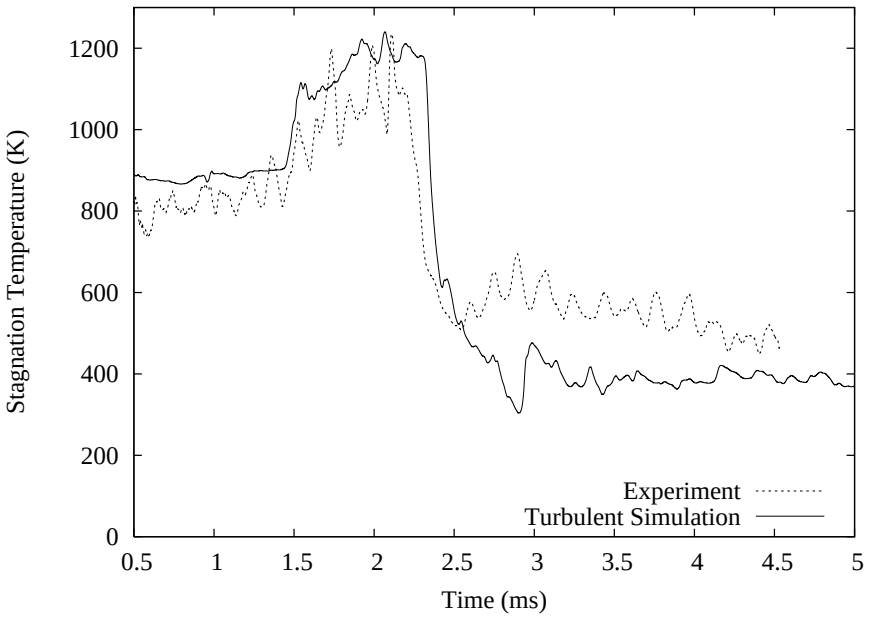

Figure 7.16: Comparison of the simulated and experimental nozzle exit stagnation temperature at the nozzle exit centreline. The simulated trace is from the **turbulent** simulation of the Nitrogen driving Nitrogen case.

7.2.2 Tailored Operation: Helium Driving Nitrogen

This section will compare the traces obtained from the simulation of the Helium driving Nitrogen case. The Mach 4 nozzle is attached and the operating conditions result in roughly tailored operation.

Grid Convergence

The process used to examine the convergence study in the Nitrogen driving Nitrogen cases is followed here. The simulations aim for a converged simulation at, or near, the fine resolution simulation. The incident shock pressure and the shock speed between the supply pressure transducer and the heat flux gauge are used as the characteristic simulation values.

The convergence plots for the Helium driving Nitrogen case is shown in Figure 7.17. The convergence using the incident shock speed between the heat transfer gauge and supply pressure transducer is shown on the left and using the pressure measured 0.1 ms after the arrival of the incident shock at the supply pressure transducer is shown on the right. The values obtained from the simulation are plotted versus the inverse of the number of cells used across the radius of the shock tube. The experimentally measured value is shown as the dashed line across the plots. It can be seen that the convergence is set to a level that is too high and the convergence over-shoots the experimental value. The values from the fine mesh simulation are slightly higher that the experimental value, for both of the characteristic values; however, they only exceed the experimental value by a small amount, possibly within the limit of experimental error, and this convergence was thought to be sufficiently accurate.

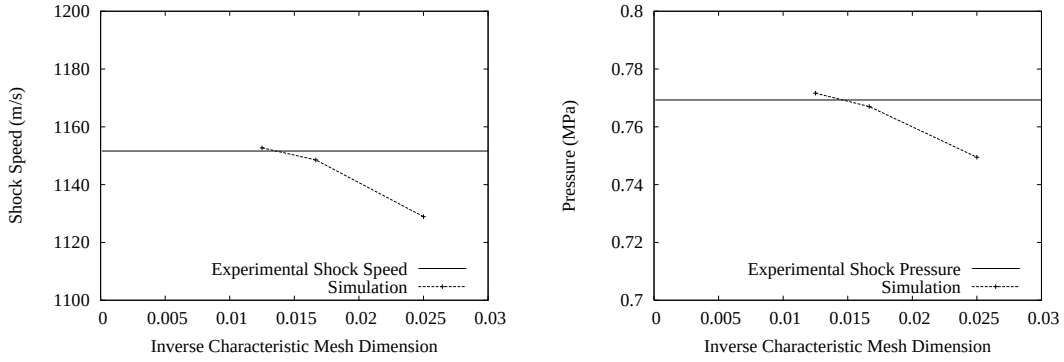

Figure 7.17: Grid convergence for the Helium driving Nitrogen case simulations. The grid convergence based on the shock speed is shown on the left and the convergence based on the pressure behind the incident shock is shown on the right.

x-t Diagram

An x-t diagram was also constructed from the MB_CNS simulation for this case. This x-t diagram is shown in Figure 7.5.

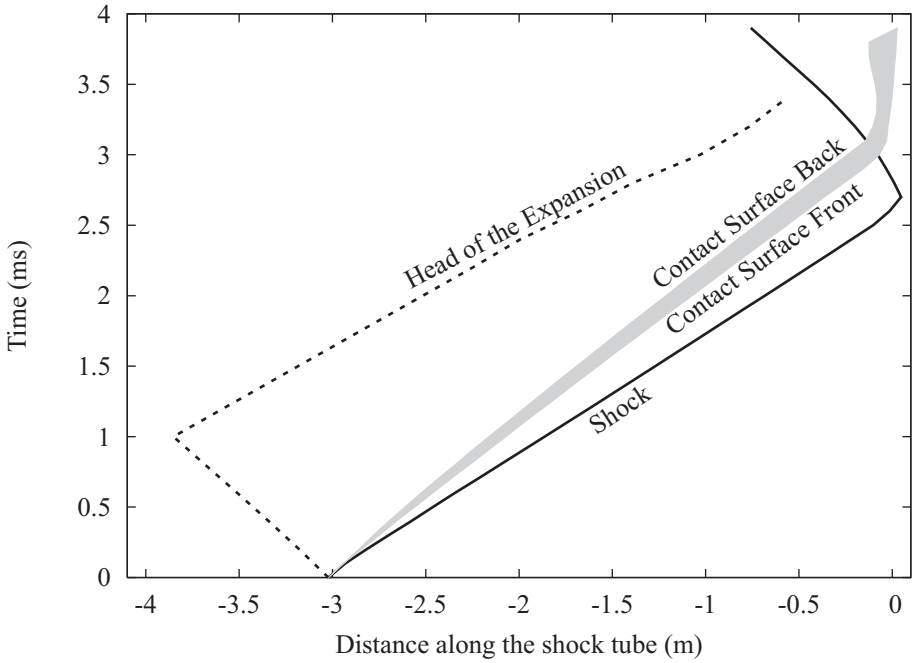

Figure 7.18: MB_CNS simulated x-t diagram for the Helium driving Nitrogen with the blanked shock tube end case. The **turbulent** simulation was used.

In the x-t diagram, the trajectories of the incident shock (solid line) and contact surface (grey region) can be seen moving towards the downstream end of the shock tube. The incident shock is shown to reflect from the downstream end of the shock tube and travel back upstream. The reflected shock interacts with the contact surface. Being a tailored mode of operation, the contact surface should be stopped by the reflected shock. The driver gas can be seen to continue to move slightly downstream. No discernible tailoring waves emanated from this interaction as would be expected. The expansion fan emanating from the primary diaphragm position can be seen to have reflected from the upstream end of the driver section and moved along the shock tube. In this case, the reflected expansion reaches the test flow and is responsible for ending the test time.

Wave Speed Diagram

The points sampled from the simulation, and used in Figure 7.18, were used to obtain the wave speed diagram shown in Figure 7.19. As with the x-t diagram, the shock is shown as the solid black line and the contact surface is shown as the grey region. This figure demonstrates the initial strong acceleration of the shock as the

waves that form during the opening of the primary diaphragm coalesce. Following this, the gradual attenuation of the shock, caused by the viscous boundary layers, is evident.

The viscous boundary layers also cause the acceleration of the contact surface. This acceleration is clearly evident in this case, more so than in the Nitrogen driving Nitrogen cases. This acceleration has a particularly important effect on the stability properties of the contact surface and will be discussed in Section 7.3.6. The effect of the diaphragm rupture model is evident in the initial axial thickness of the contact surface.

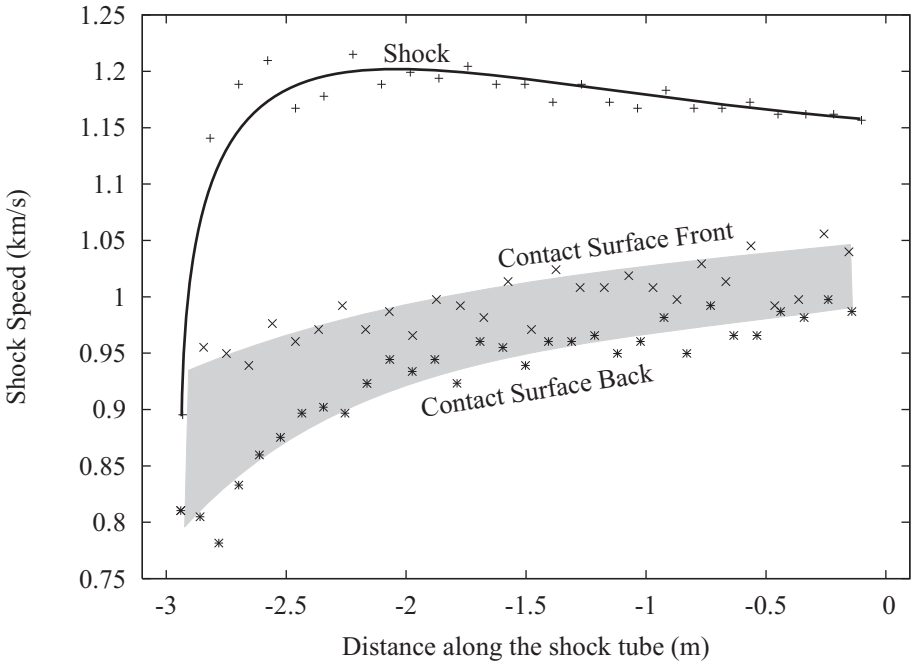

Figure 7.19: Evolution of waves speeds along the tube from the **turbulent** MB_CNS simulations for the Helium driving Nitrogen case. The speed of the shock and the front and the back of the contact surface are shown.

The delay in the arrival time of the incident shock between the heat flux gauge and the pressure transducer was used to identify the shock speed, both in the experiment and the simulations. The experimentally measured shock speed was 1151.6 m/s. The shock speed predicted by the laminar simulation was 1225.3 m/s (6.4% high) and the turbulent simulation on the fine mesh was 1152.7 m/s (0.1% high). This shows an 6.3% improvement with the implementation of the Baldwin Lomax model.

Supply Pressure Transducer

1. Laminar Simulation (Figure 7.20)
The laminar simulation significantly over estimates both the incident and the reflected shock pressures. The arrival time of the reflected shock is close to the experimental value. In the experimental trace, the rise due to the arrival of the reflected shock shows a curved profile, resulting from the interaction of the reflected shock with the boundary layer. This effect is not reproduced in the laminar simulation, which has a sharp profile at the arrival of the reflected shock. The expansion arrives at the correct time, but the pressure from the time of the arrival of the reflected shock onwards is significantly over estimated.

2. Turbulent Simulation (Figure 7.8)
With the calibrated turbulence model, the pressure behind the incident shock is reproduced accurately, but the arrival of the reflected shock is slightly early. The most significant difference between the turbulent simulation and experimental traces is at the arrival of the reflected shock. The foot of the reflected shock has bifurcated due to its interaction with the incident boundary layers and the effect of the bifurcated shock foot is even more significant here than in the Nitrogen driving Nitrogen traces. In contrast, the properties of the flow following the arrival of the reflected expansion are accurately reproduced, with the pressure being only slightly low throughout.

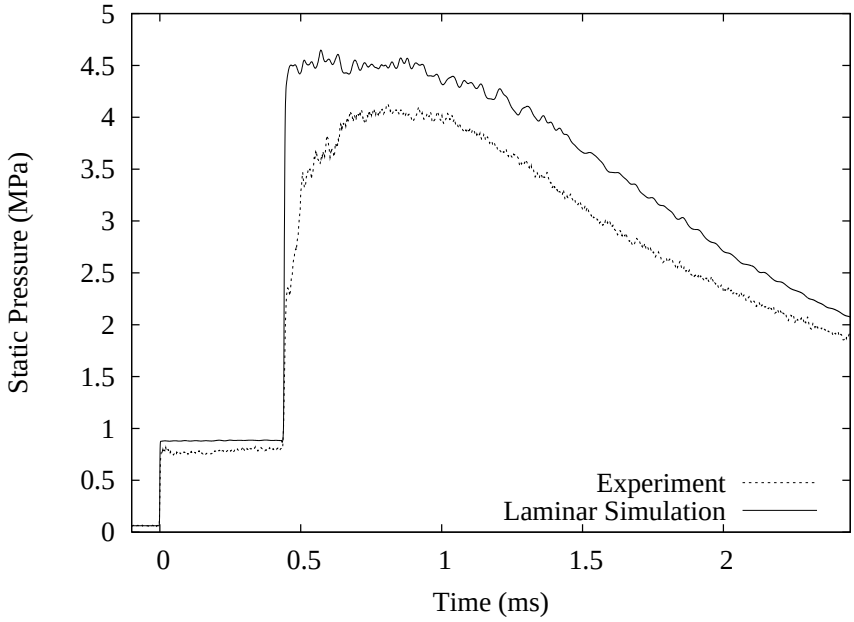

Figure 7.20: Static pressure recorded by the supply pressure PCB transducer during the Helium driver case for the **laminar** simulation compared to the experiment.

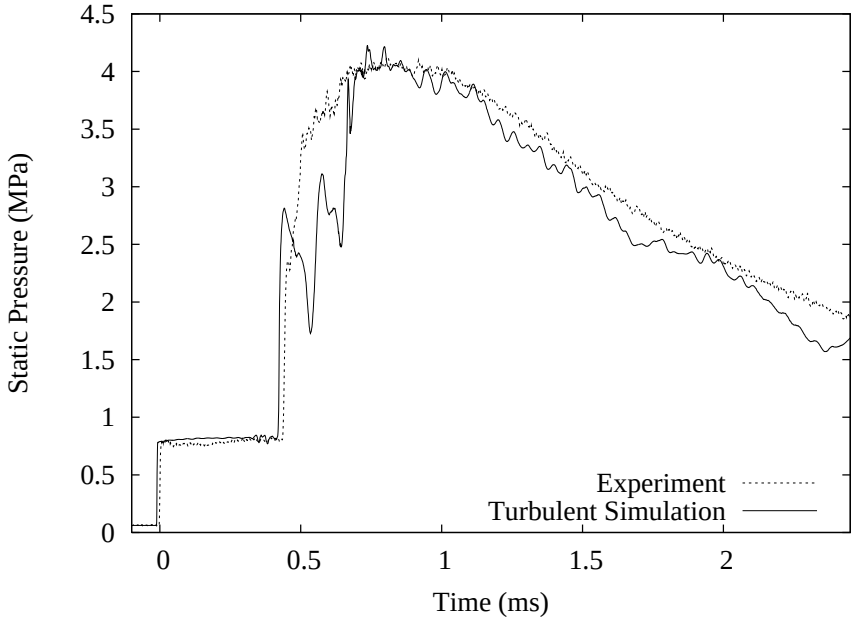

Figure 7.21: Static pressure recorded by the supply pressure PCB transducer during the Helium driver case for the **turbulent** simulation compared to the experiment.

Heat Flux Gauge

As with the Nitrogen driving Nitrogen cases, the magnitude of the heat flux was not known so the level of the experimental heat flux trace was simply aligned with the simulated trace. Again, the simulated heat flux was calculated using a linear relation given in Section 7.1.7.

1. Laminar Simulation (Figure 7.22)
The initial jump in heat flux is caused by the passage of the incident shock. It can be seen in the experimental trace that the heat flux to the wall from the gas in the region behind the incident shock is approximately constant for 600 μs, although there are significant fluctuations during this period. In the simulation, the heat flux decreases slighly during this period. This decrease in heat flux is caused by the variation in heat flux through the increasing boundary layer which is increasing in size with distance from the shock. The relatively sudden decrease in heat flux is caused by the arrival of driver gas at the gauge. The instantaneous rise in heat flux, in the middle of this decrease, is caused by the arrival of the reflected shock, which has already passed through the contact surface. The arrival time of this shock is accurately estimated in this simulation, but additional strong waves are evident in the simulated trace following the passage of the reflected shock. These are not evident in the experimental trace.

2. Turbulent Simulation (Figure 7.23)
A sharp peak in heat flux through the boundary layer immediately behind the reflected shock is evident. The profile from the turbulent simulation has roughly the same characteristics as the laminar profile, except in the region around the reflected shock foot. The width of the material in the shock foot is evident in the simulated trace, being responsible for the longer duration of high heat flux at the arrival of the reflected shock than in the experimental trace. This is due to the high temperature material being carried at the foot of the shock in the simulation. No additional waves are evident in the trace at late times, as in the experimental trace. The simulated trace ends at 1.5 ms.

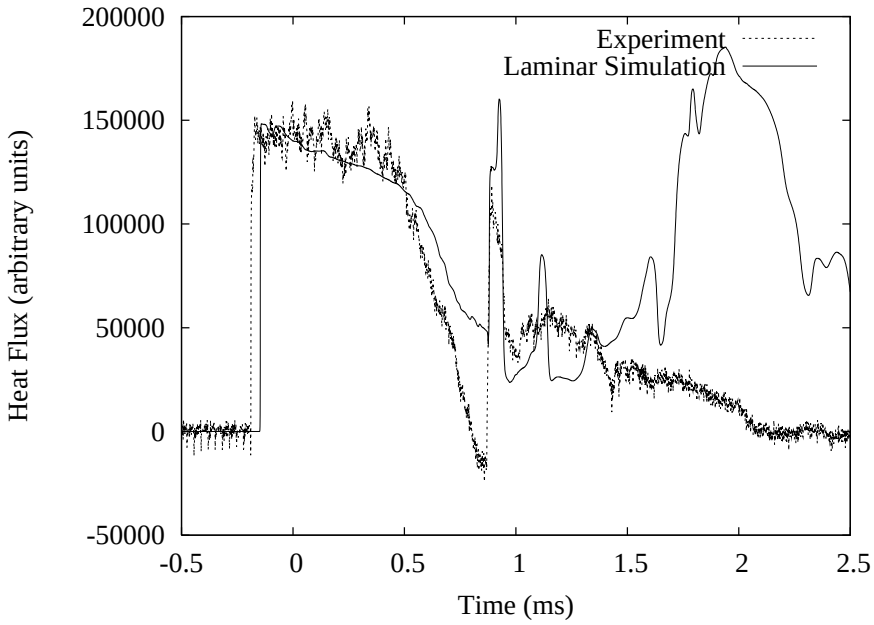

Figure 7.22: Heat flux recorded by the thin film heat flux gauge during the Helium driving Nitrogen case for the **laminar** simulation compared to the experiment.

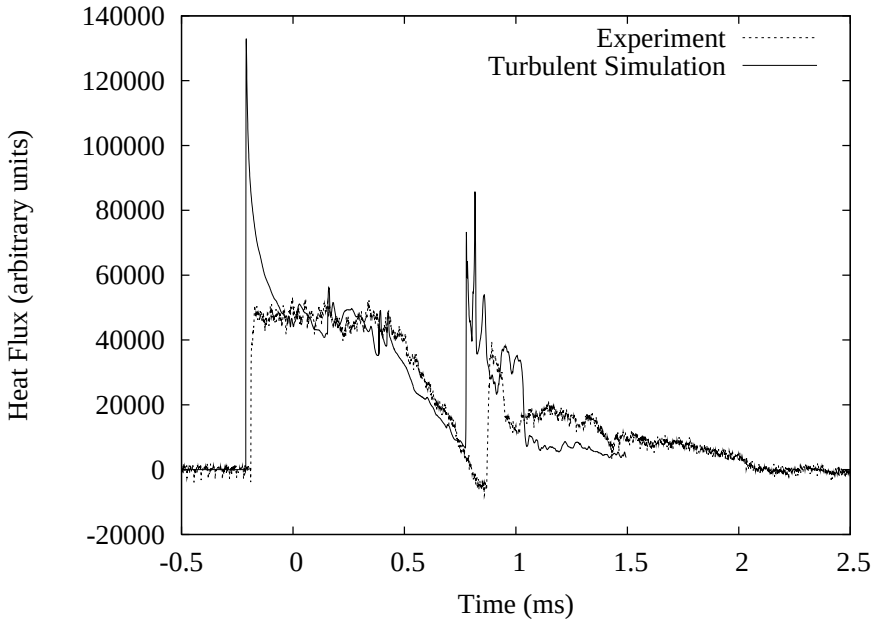

Figure 7.23: Heat flux recorded by the thin film heat flux gauge during the Helium driving Nitrogen case for the **turbulent** simulation compared to the experiment.

Nozzle Exit Pitot Pressure

Figure 7.24 shows the experimental trace of pitot pressure measured at the centreline of the nozzle exit plane compared to the trace from the turbulent simulation. In addition to its implementation along the length of the shock tube, the use of the Baldwin-Lomax model on the nozzle walls improved the accuracy of the simulation of the test flow. As with the Nitrogen driving Nitrogen cases, the characteristics of the startup waves were accurately reproduced. The steady test time and the magnitude of the pitot pressure during the test time are accurately reproduced, as is the arrival of the reflected expansion in the test flow. For this case, the simulated trace close enough to the experimental trace to be within the levels of the fluctuations, throughout the trace shown. Also, the levels of fluctuations during the test flow period are of the same magnitude in each of the traces. This will be discussed further in Section 7.3.10.

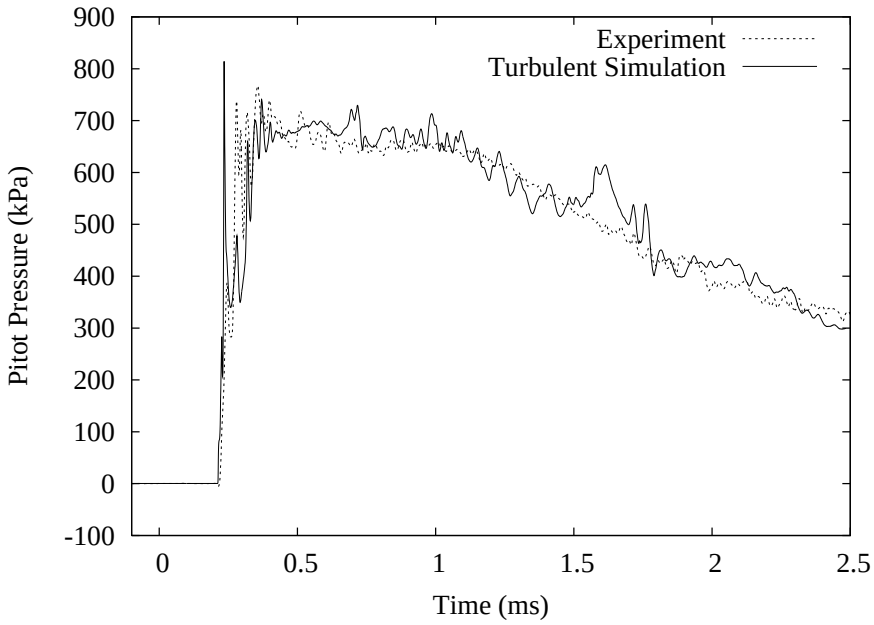

Figure 7.24: Pitot pressure recorded by the nozzle exit centreline pitot probe during the Helium driving Nitrogen case for the **turbulent** simulation compared to the experiment.

7.3 Discussion of the Simulation Results

The comparison of experimental and simulated traces in Section 7.2 demonstrates that the simulations provide a good reproduction of the traces recorded during experiments. The simulations are provided with only the initial operating conditions used in the experiments and the geometry of the facility. This provides strong support as to the accuracy of the simulations and to the modelling assumptions that have been made. There were some differences between the experimental results and the simulation results. These differences will be further addressed in this section.

This section discusses important fluid dynamic processes in the operation of a shock tunnel as illustrated through the simulations. The visualisation of the development of the flow throughout the whole facility is an important feature of these simulations. The evolution of flow structures and the interaction of flow features that have evolved properly is shown to be an important aspect of the overall flow.

The discussion in this section uses simulations performed on the fine resolution mesh, the details of which were described in Section 7.1.1. Apart from Section 7.3.2, which discusses the resolution of the boundary layers, the effect of the refinement of the mesh is addressed in the context of each process being investigated. Through refinement of the mesh from the coarse and medium resolution meshes, to the fine resolution mesh, it is shown that the primary features of the solution are sufficiently independent of mesh resolution.

The times quoted in Section 7.2 were all set with the zero time aligned with the arrival of the incident shock at the supply pressure transducer. This was done so that the simulated traces could be aligned with the experimental traces. Since this section only deals with the simulation, all times discussed in this section will be quoted from the time of the start of the simulation (at the initiation of diaphragm rupture). These time offsets are shown in Table 7.3.

7.3.1 Simulations of Diaphragm Rupture Mechanics

Simulations were run assuming an ideal removal of the diaphragm. These simulations were unable to reproduce the experimental results, even through variation of the simulation parameters. A diaphragm rupture model was implemented, which enabled the simulations to be able to reproduce the experimental results, as shown in Section 7.2.

An iris based model is used to account for the effect that the mechanics of the rupturing primary diaphragm has on the flow. The rupture is assumed to be a steady process, with the opening area increasing linearly with time. The total rupture time is assumed to be 94% (to account for the slow initial opening) of a 200 μs rupture time. These assumptions were based on the experimental observations of Rothkopf and Low [197], which were discussed in Section 2.1.4. These assumptions are believed to be valid since the diaphragm material is aluminium and the rupturing of the diaphragm is initiated by being pierced by the spike. From observations of used diaphragms, the simulated diaphragm is assumed to rupture to a diameter of 57 mm. The implementation of the rupture model in the code was discussed in Section 7.1.4.

The analysis in this section uses the Nitrogen driving Nitrogen case with the blanked end, unless otherwise stated. Since the rupture process is the same for the three cases simulated, the resulting flow at early times is similar for the cases considered.

Sequence of the Rupture Process

Figures 7.25 and 7.26 show a sequence of numerical Schlieren (in the upper half of the frames) and driver gas mass fraction (lower half) frames of the initial flow resulting from the implementation of the diaphragm rupture model. The frames are taken at 20 μs intervals starting at the instant of the start of the simulation.

The gradual opening of the diaphragm can be seen through the frames. Due to the initially small opening, the shock is spherical as it emerges from the opening. In the frame at 60 μs, this spherical shock front reaches the side wall. Following this it reflects back into the flow.

In these early frames, the driver gas can be seen emerging through the opening. This gas expands outwards, but moves predominantly downstream rather than outwards from the diaphragm. It is not until the frames as late as 200 μs that the driver gas can be seen to impinge on the side wall. In the frame at 100 μs, the segment of the shock reflected from the side wall can be seen to reach the contact surface. It continues to pass through the driver gas in later frames. These transverse shocks

bounce back and forth across the tube, repeatedly passing through, and processing the contact surface. In this interaction, both the contact surface and the shocks are significantly deformed, with vorticity being generated at the contact surface. This vorticity results in the curling up of the outward edge of the contact surface seen in later frames.

The forward moving segment of the spherical shock moves along the tube. This part of the shock undergoes a regular reflection at the wall, until, as the angle of incidence of the shock against the wall is such that the regular reflection is no longer possible, a Mach stem forms. The Mach stem forms what becomes the primary shock, as the triple point moves along the incident shock towards the tube centreline. This Mach stem can first be seen to form in the frame at 120 μs and the shock is essentially planar by 240 μs. As this planar shock forms, the transverse waves continue to weaken. This weakening can be seen by comparing the strength of the transverse waves in the frames at 220, 240 and 260 μs. This process demonstrates the inherent stability of the shock wave, as it forms into a roughly planar form within 240 μs, and less than two shock tube diameters from the diaphragm.

It was assumed that the diaphragm opens to 57 mm of the 62.2 mm diameter of the shock tube. The remaining area behind what is left of the diaphragm results in a pocket of stagnant gas. As the gas is forced around this gas, two oblique shocks and a Mach disk form. Throughout the frames, the expansion of the driver gas can be seen moving back upstream into the driver section.

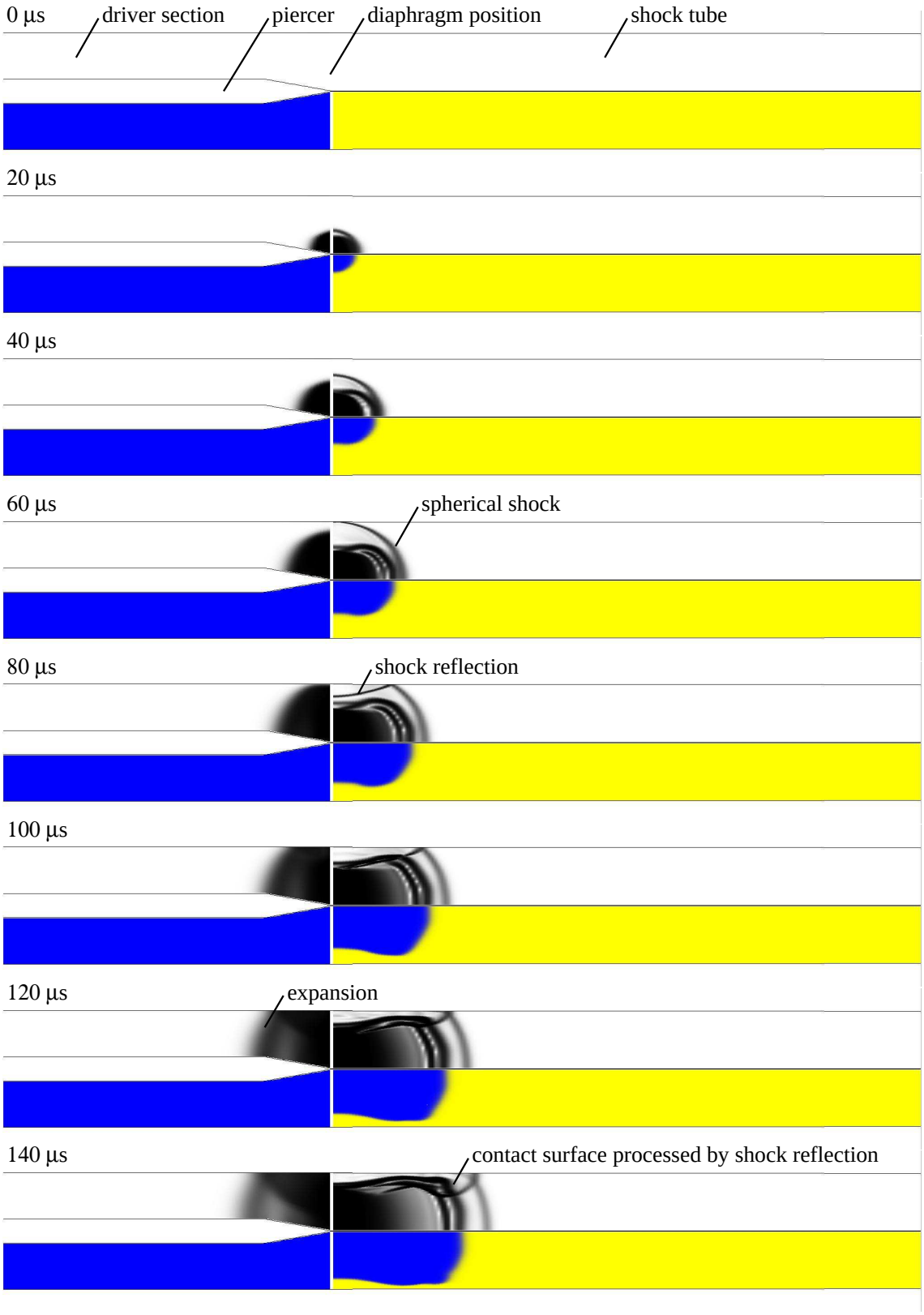

Figure 7.25: Part one of a sequence of frames showing the initial flow that results from the implementation of the diaphragm rupture model. Numerical Schlieren is shown in the upper half of the frames and mass-fractions are shown in the lower half, with driver gas in blue and driven gas in yellow. The frames are at 20 μs intervals starting at the instant of the start of the simulation.

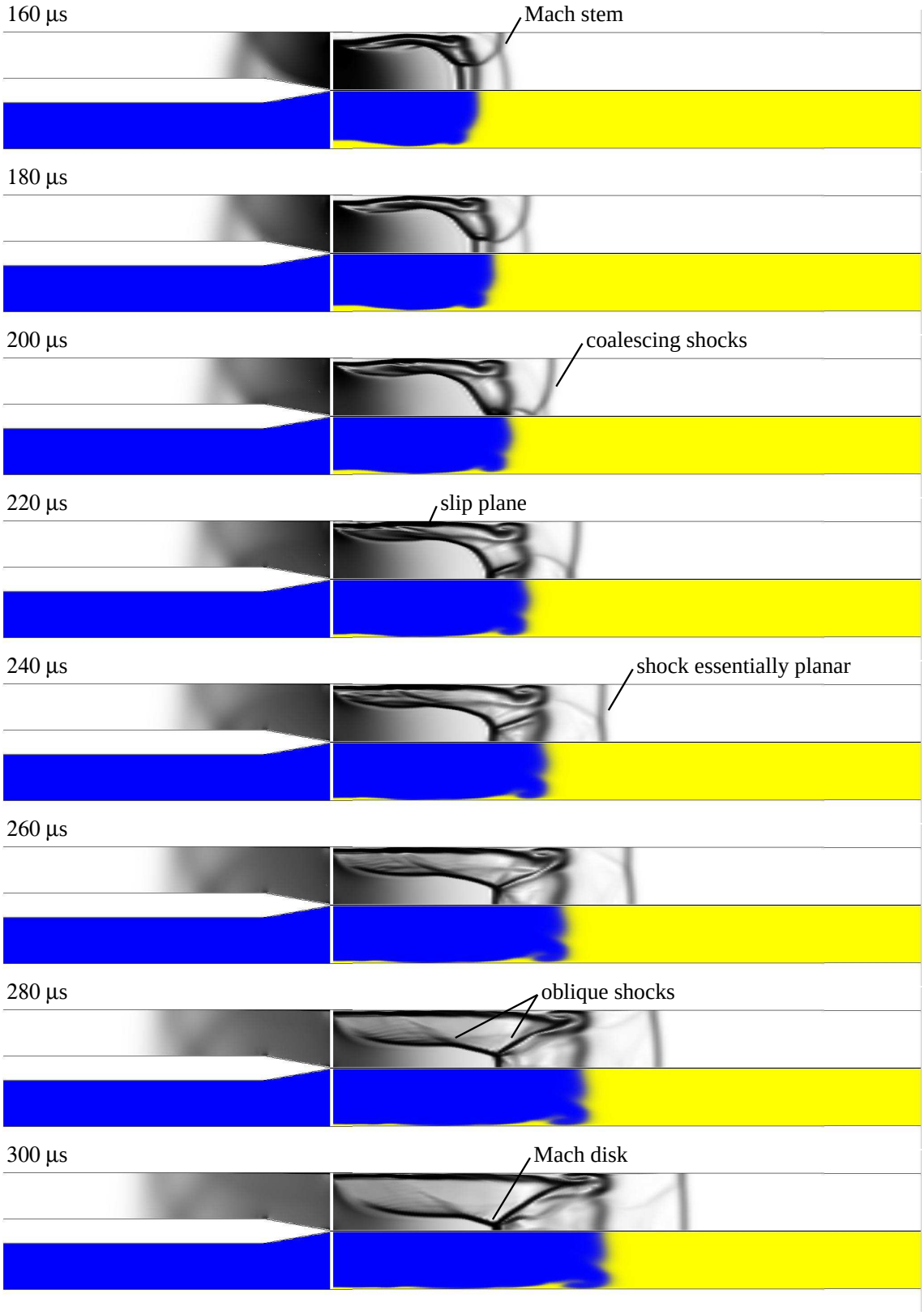

Figure 7.26: Part two of a sequence of frames showing the initial flow that results from the implementation of the diaphragm rupture model. Numerical Schlieren is shown in the upper half of the frames and mass-fractions are shown in the lower half, with driver gas in blue and driven gas in yellow. The frames are at 20 μs intervals starting at 160 μs.

Effect on the Contact Surface

The rupture mechanics of the primary diaphragm has a significant effect on the characteristics of the contact surface. Unlike the shock, which is inherently stable, the contact surface evolves along the tube under the influence of how it is affected by the rupture of the primary diaphragm.

Cambier et al. [35] and Petrie-Repar [176] observed that the diaphragm rupture process results in an increased axial speed of the driver gas near the walls, relative to gas near the centreline. Figure 7.27 shows a numerical Schlieren (upper half) and axial velocity (lower half) frame of the flow resulting from the use of the diaphragm rupture model. This frame is at 300 μs after the start of the simulation.

As the driver gas enters the shock tube, it emerges from the contraction in area through the diaphragm. This gas passes through an oblique shock, and a weaker re-attachment shock, which straightens the flow. Following these shocks, the flow converges slightly towards the centreline, before passing through a second set of oblique shocks, which again straighten the flow. This shock structure remains from the transverse moving shock resulting from the initial emerging spherical shock. The gas which moves straight along the centre of the tube passes through a Mach disk, which raises the pressure of the gas to the level of the gas that passed through the two oblique shocks. This gas emerges with a higher velocity than the gas which passes through the Mach disk.

The axial speed of the driver gas, shown in Figure 7.27, ranges from over 600 m/s near the wall to the nearly stationary gas behind the Mach disk. This gradient in velocity results in the jetting of driver gas near the walls, which imposes an outward mixing of the gas at the contact surface as it moves along the shock tube. This will be discussed further in Section 7.3.6.

This gradient in velocity is compounded by the generation of vorticity, which results from the interaction of the contact surface with the transverse moving shocks. This vorticity is a result of a baroclinic torque, of the same type as will be discussed in Section 7.3.7 in the context of the interaction of the contact surface with the reflected shock. This vorticity acts to promote the deformation and mixing at the contact surface.

The assumption of an ideal primary diaphragm rupture process would result in well defined, planar features forming. This may be appropriate for the shock, given its inherent stability, but in the case of the contact surface, it would result in an inaccurate representation.

Skinner [212] believed that the gradually opening diaphragm, resulted in the jetting of driver gas along the centreline of the tube. It has been observed experi-

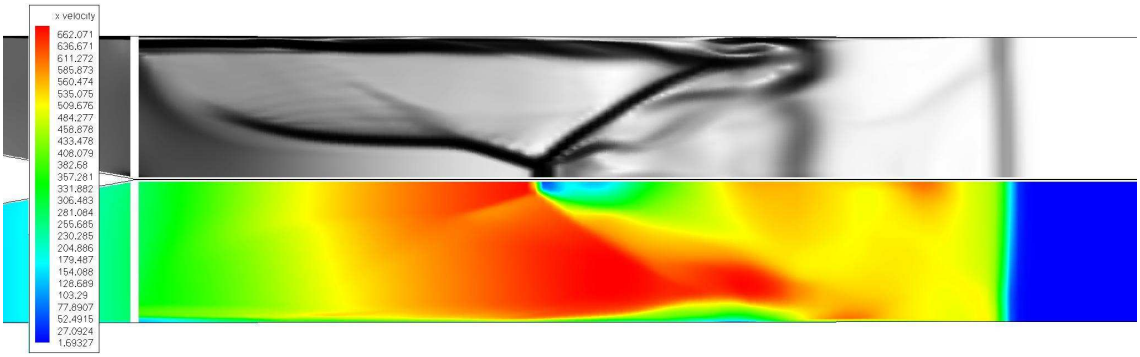

Figure 7.27: Contours of numerical Schlieren (upper half of frame) and axial velocity (lower half of frame) resulting from the diaphragm rupture model opening in 200 μs in the Nitrogen driving Nitrogen case simulations. The frame is at 300 μs from the start of the simulation.

mentally [197] that the initiation of the diaphragm rupture process is a slow process. It was thought that as a result the driver gas was allowed to penetrate into the test gas through this initially small opening. This effect was believed to be the primary cause of driver gas contamination in the experiments conducted [212]. The oblique shock structure observed in these simulations acts against this process, and is now believed to be the more dominant of the two mechanisms.

Effect of the Opening Time

In addition to the analysis of the flow field resulting from the 200 μs rupture time, the total time taken for the diaphragm to rupture was varied from instantaneous to 100, 200 and 400 μs. This allowed variation around the time observed by Rothkopf and Low [197] in order to ascertain the sensitivity of the flow to this time.

Figure 7.28 shows numerical Schlieren (upper half) and driver gas mass-fraction (lower half) frames from the flow fields resulting from these different opening times. The effect of the rupture time on the flow can be seen in what is essentially a delay in the arrival of flow features at positions along the tube, an increase in the strength of the oblique shock structure and a significant increase in the early deformation of the contact surface. The shock and contact surface speeds far downstream of the diaphragm are relatively unaffected by this process; the only significant effect far downstream is on the characteristics of the contact surface. In the 400 μs opening time frame, the diaphragm has not yet fully opened.

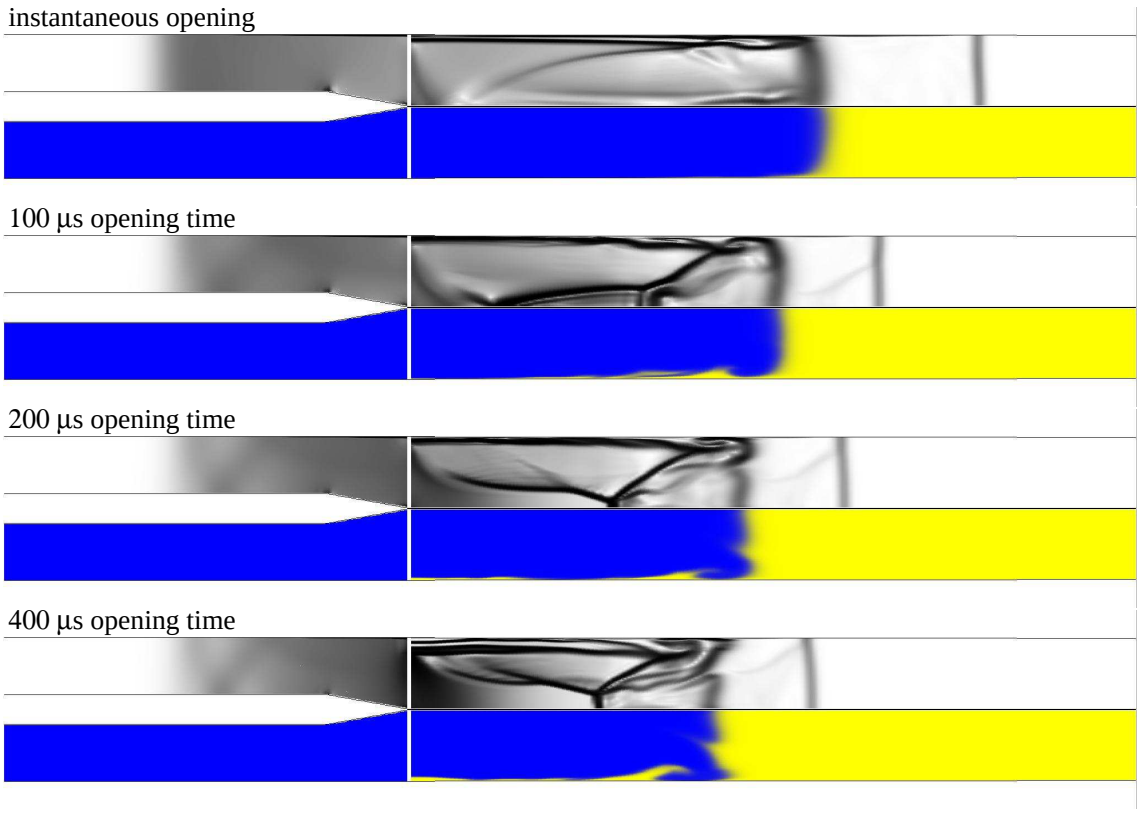

Figure 7.28: The effect of diaphragm opening time on the resulting flow. Numerical Schlieren (upper half of frames) and driver gas mass-fraction (lower half of frames) 300 μs after the diaphragm rupture was initiated. The instantaneous (top), 100, 200 and 400 μs (bottom) opening time flows are shown.

Figure 7.29 shows the shock trajectories versus distance from the diaphragm position, resulting from the use of the diaphragm rupture model. The simulations were run for the instantaneous, 100, 200 and 400 μs opening times. The initially low shock speeds, followed by the acceleration of the shocks as the spherical shocks coalesce to form the planar shock is evient for all four of the traces. The variation in the early centre-line speed of the shock results from the formation of the planar shock from the spherical shock front. This process is also described by by Rothkopf and Low [196], Zeitoun et al. [254] and Petrie-Repar [176].

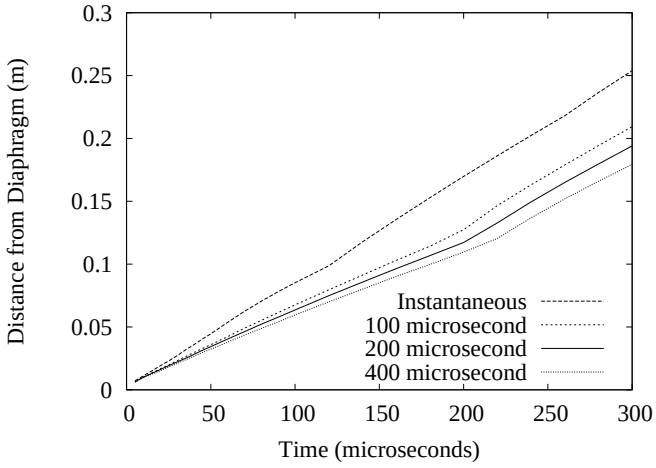

Figure 7.29: Shock trajectories versus distance from the diaphragm position, resulting from the use of the diaphragm rupture model. Trajectories are shown for the model with instantaneous opening, 100, 200 and 400 μs opening times.

Effect of Mesh Resolution

In order to determine whether the rupture process is being simulated in a mesh resolution independent manner, the results are compared with lower resolution simulations. Figure 7.30 shows frames from the coarse, medium and fine meshes, demonstrating the effect of mesh refinement. These frames are taken from the Helium driving Nitrogen case, instead of the Nitrogen driving Nitrogen case as with the other frames. Comparison of these frames show that the simulated flow is essentially the same in the three frames, with sharper resolution being achieved with the medium and fine meshes; this is particularly seen in the resolution of the shocks. It is believed that the fine resolution simulations are not affected by mesh resolution issues.

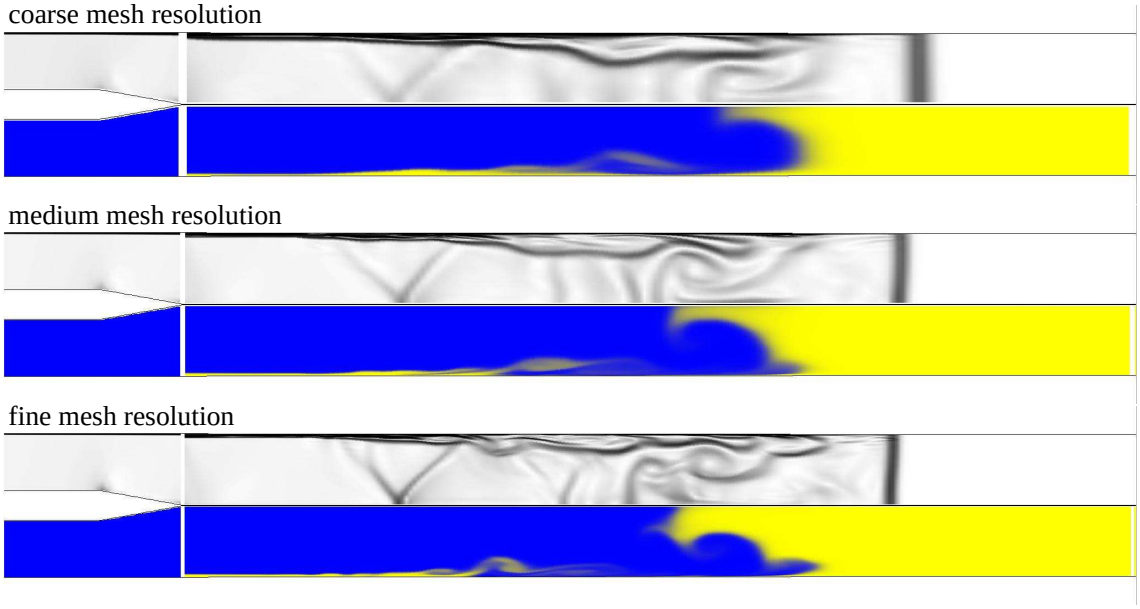

Figure 7.30: The effect of the mesh resolution on the modelling of the diaphragm rupture process. Numerical Schlieren (upper half of frames) and driver gas mass fraction (lower half) contours are shown, comparing the diaphragm rupture induced flow for the coarse (top), medium (middle) and fine (bottom) meshes. The frames are shown at 300 μs.

7.3.2 Resolution of the Boundary Layers

The proper resolution of the boundary layers is vital to the accuracy of the simulations. Properly resolving boundary layers places one of the most significant limitations on the mesh. The shock that processes the test gas has been attenuated along the length of the tube by the influence of the boundary layers. The flow of material into the boundary layers also results in the acceleration of the contact surface, further decreasing the test time.

Given the difficulty in resolving the boundary layers, the mesh is refined towards the wall as was shown in Figure 7.1, in order to focus computational effort in this region. Qualitative inspection of the flow in the boundary layers indicate that sufficient cells are allocated inside them. The good correlation achieved between the simulated heat flux to the wall, which is based on the gradients of properties through the boundary layer, to the experimentally recorded traces from the thin film heat flux gauge, indicates that the boundary layers are being modelled adequately.

The boundary layers from the Nitrogen driving Nitrogen case were examined in detail. It is believed that the analysis conducted on these simulations are applicable to the other cases, given that they use the same mesh, the same modelling assumptions and have similar flow conditions.

u+ versus y+

In order to provide a quantitative assessment of the quality of the grid near the wall, the grid spacing parameters u+ and y+ will be used. The two values u+ and y+ are calculated using the equations:

$$u^+ = \frac{\overline{u}}{u_\tau} \tag{7.2}$$

$$y^+ = \frac{y u_\tau}{\nu} \tag{7.3}$$

where:

$$u_\tau = \sqrt{\frac{\tau_w}{\rho_w}} \tag{7.4}$$

ρ_w is the local density near the wall, τ_w is the shear stress at the wall, y is the distance to the centre of the first row of cells, ν is the kinematic viscosity and $\overline{u}$ is the tangential velocity at the first row of cells.

A study of the simulation of flat plate boundary layers was conducted by Dilley [58]. The grid spacing parameters required to adequately resolve surface heat transfer in that study were discussed in Cockrell Jr., Auslender, White and Dilley [49]. It was stated that a y+ value of 2 is required to ensure adequate resolution

of the boundary layer. This was not achieved in this study by more than an order of magnitude; however, this study does not depend on the quantitative prediction of heat transfer rates. The prediction of heat transfer through the boundary layer discussed in Section 7.2 appeared to be predicted sufficiently for this study. It is believed that the requirements on y+ are not as strict as in other studies, with different requirements, such as Cockrell et al. and Dilley. This study is primarily concerned with the boundary layers for their effect on the flow along the shock tube. With the correct level of the attenuation of the shock, and the critical aspects of the interaction of the reflected shock with the boundary layer being accurately reproduced, it is believed that the boundary layers are sufficiently resolved for this study.

By varying $\overline{u}$ and y, as the cells move away from the wall, profiles of u+ versus y+ can be produced. These figures provide information on the quality of the simulated boundary layer, including the number of cells inside the boundary layer. If the number of cells inside the boundary layer is too low then the numerical techniques used cannot be expected to resolve the boundary layer gradients properly.

A theoretical profile of u+ versus y+ through a boundary layer is shown in Figure 7.31. Comparison of this plot with the profiles obtained from these simulations demonstrates the difficulty in modelling all aspects of a boundary layer.

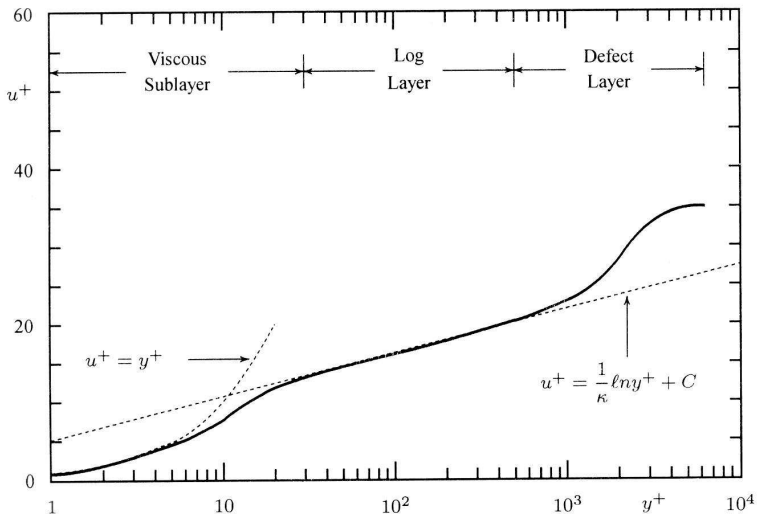

Figure 7.31: A plot of u+ versus y+ based on a theoretical representation of a boundary layer. Reproduced from Wilcox [244].

Figure 7.32 shows the u+ versus y+ plots for the boundary layers growing behind the shock on the coarse resolution mesh. Separate profiles were extracted across the radius of the tube, at 100 mm intervals from the shock, through the growing boundary layer. The laminar boundary layer is shown on the left and the boundary layer utilising the turbulence model is shown on the right.

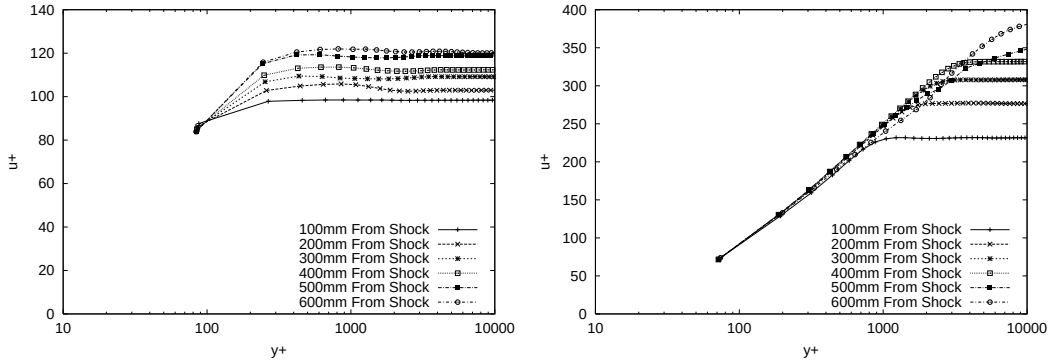

Figure 7.32: Profiles of u+ versus y+ for the laminar (left) and turbulent (right) growing boundary layers behind the shock on the coarse resolution mesh.

The left most point shows the values from the inner most cell in the boundary layer. These profiles all follow the same line through the boundary layer. This would be expected since each profile is through a larger section of the same boundary layer, modelled on the same mesh. The points corresponding to cells outside the boundary layer all have the same axial speed and therefore lie on a flat line of u+. The effect of the growing boundary layer is seen in the movement away from the wall of the points in the flat line.

The use of the Baldwin-Lomax eddy viscosity model [11], accounting for turbulence in the boundary layers, has a significant effect on the simulated boundary layers. The model makes the boundary layers significantly thicker, thereby increasing the number of cells that are inside the boundary layers. This has the effect of increasing the ability of the numerical schemes to resolve the boundary layers. For example, in the profile closest to the shock, which is at the thinnest part of the boundary layer, the turbulent simulation has seven cells inside the boundary layer. This number increases along the growing boundary layer. Across all of the profiles, the simulated laminar boundary layers have between one and two points inside the boundary layer; too little to be able to provide adequate resolution.

Figure 7.33 compares the u+ versus y+ plots for the boundary layers growing behind the shock on the coarse and fine resolution meshes, both incorporating the turbulence model. The improvement in the resolution of the mesh, with more cells in the boundary layers and across the radius of the tube is seen between the two

meshes. The same characteristics are evident in the plots in that the points through the boundary layer are co-linear.

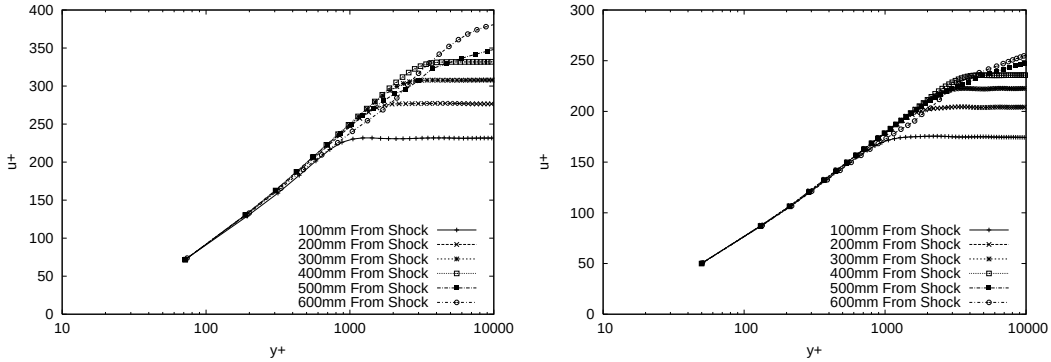

Figure 7.33: Comparison of the profiles of u+ versus y+ for the growing boundary layer behind the shock for the coarse (left) and fine (right) resolution meshes.

7.3.3 Shock Reflection Process

When the shock reaches the end of the shock tube it reflects from the downstream end of the tube. The shock reflection process has an important impact on the resulting flow, influencing the interaction between the resulting reflected shock and the boundary layer, and introducing significant fluctuations into the test flow gas.

In the blanked end case, this reflection is simple. With the Mach 4 nozzle cases, the shock reflection is from the curved surface of the nozzle mouth and the secondary diaphragm. This difference in the reflection process contributes to the differences observed in the experimental results from the blanked end case and the nozzle case, which, for the Nitrogen driver, both use the same conditions.

Sequence of the Reflection

Figures 7.34 and 7.35 show a sequence of numerical Schlieren frames of the reflection of the shock from the Mach 4 nozzle throat. The frames are taken at 10 μs intervals starting at 2.69 ms. The Helium driver case simulation is used.

The first two frames show the incident shock approaching the nozzle. The boundary layer on the shock tube walls is also evident. In the frame at 2.71 ms, the shock first reaches the nozzle surface. The shock gradually reflects from the nozzle surface over the next two frames, as the shock reflection moves along the convergent section of the nozzle.

The part of the incident shock near the centreline reflects from the secondary diaphragm in the throat of the nozzle. As the pressure against this diaphragm increases to 300 kPa, the simulated diaphragm is assumed to be instantaneously removed. This diaphragm is ruptured so quickly that no upstream reflection of the shock is evident. As this diaphragm is removed the flow expands into the nozzle.

The reflection of the shock from the nozzle surface moves outwards from the nozzle surface, in an upstream direction and across the flow. A Mach disk forms on the nozzle centreline as this reflected shock crosses its matching reflection from the other wall. This is shown in the figure at 2.75 ms. The resulting triple point moves upstream and away from the nozzle centreline. As this triple point moves through the flow it leaves a contact surface between the gases processed by the Mach stem, or by the two segments of the reflected shock. An important feature, noted by Jacobs [111], is the formation of a vortex at this contact surface. This vortex moves slowly upstream and remains present in this region until it is swept out of the flow by the nozzle expansion or by the arrival of the contact surface.

The reflection of the shock reflects from the opposing side of the nozzle mouth for a second time. This shock, labelled in the frame at 2.78 ms, moves quickly upstream

and, by the frame at 2.81 ms, coalesces with the main reflected shock. Further reflections continue this trend, but the shocks decrease in strength as they continue.

While the reflected shock moves upstream, it continues to coalesce with other shocks and the triple point moves outwards towards the wall. Through this process the reflected shock becomes planar. The inherently stable nature of the shock is demonstrated in this reflection process.

The significant fluctuations in the nozzle supply region left by this process are evident in the later frames in the sequence. The steady shocks through the nozzle are also beginning to form by the later frames.

Similar nozzle reflection processes are described in more detail by Tokarcik-Polsky and Cambier [234] and Lee [128].

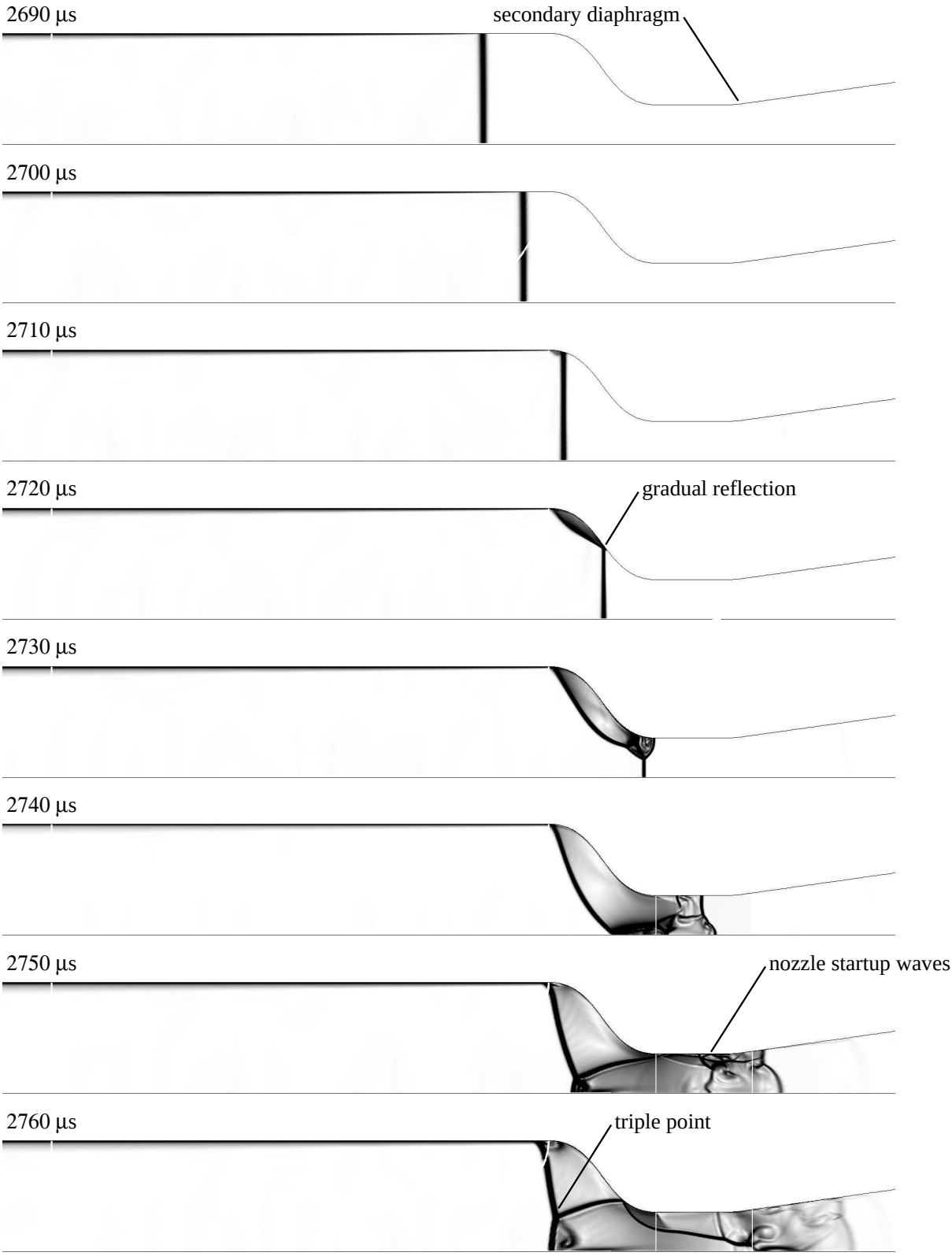

Figure 7.34: Part one of the sequence of numerical Schlieren images showing the shock reflection process from the nozzle throat. The Helium driver case simulation is used. The frames start at 2.69ms and are at 10μ s intervals.

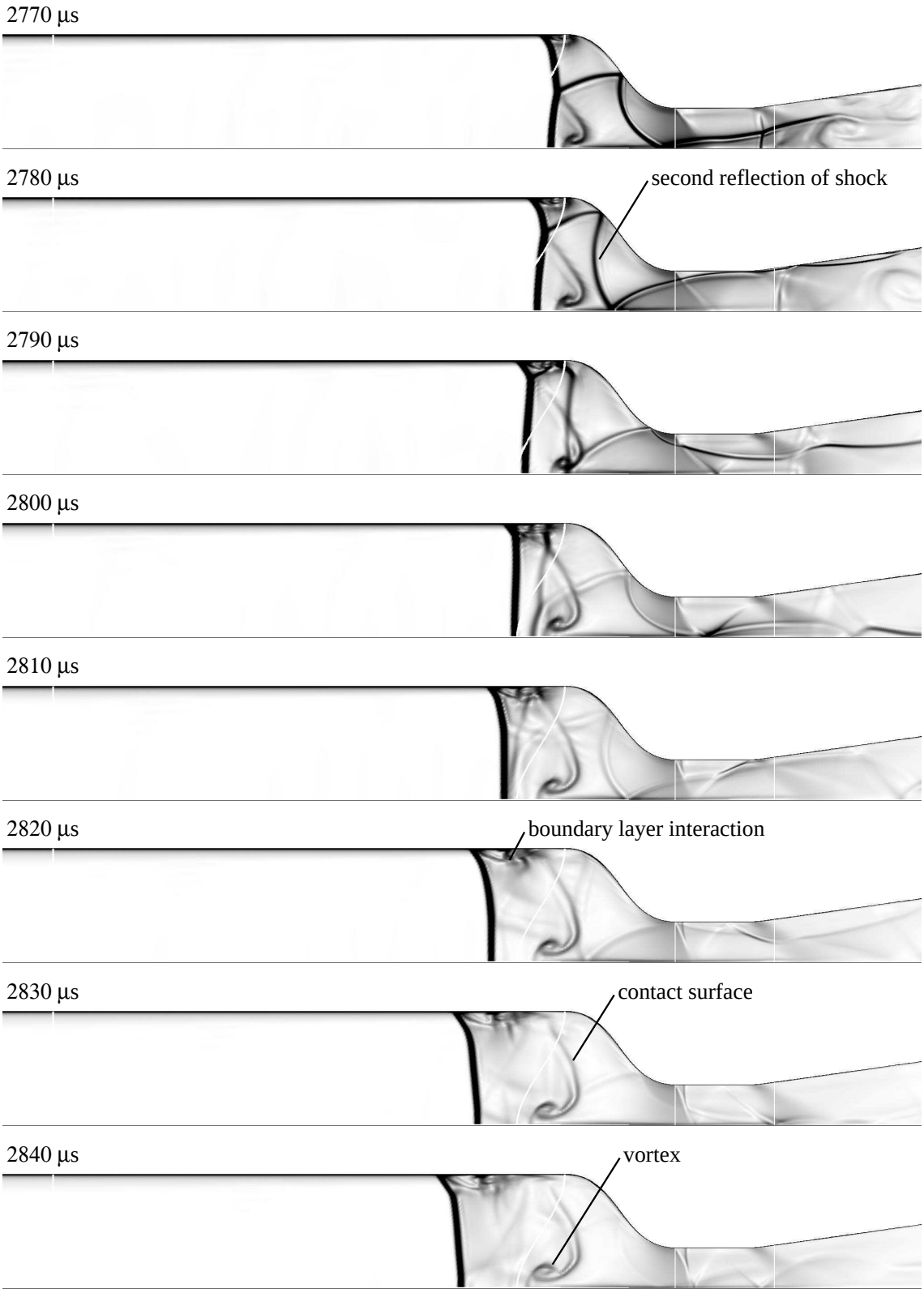

Figure 7.35: Part two of the sequence of numerical Schlieren images showing the shock reflection process from the nozzle throat. The Helium driver case simulation is used. The frames start at 2.77ms and are at 10μ s intervals.

Effect of Mesh Resolution

Figure 7.36 shows the effect of mesh refinement on the simulation of the shock reflection process. Differences between the three frames is minimal. The position of the contact surface, and therefore the locus of the triple point appears to be unchanged. The most obvious of differences between the frames is in the position of the second reflection of the shock. This is a fast moving shock and the differences in the position of this shock are almost certainly due to the time misalignment of the frames. The physics of the shock reflection process is not believed to be influenced by mesh refinement.

Refinement of the mesh resulted in a slight increase in the speed of the shock along the shock tube. This difference in shock speed meant that these frames were realigned to account for the difference in shock arrival time. The frames were attempted to be aligned with the fine resolution of mesh at 2700 μs. The resulting coarse and medium meshes are shown at 2715 and 2708 μs respectively. Differences in the frames are largely due to time misalignment still present between these three frames.

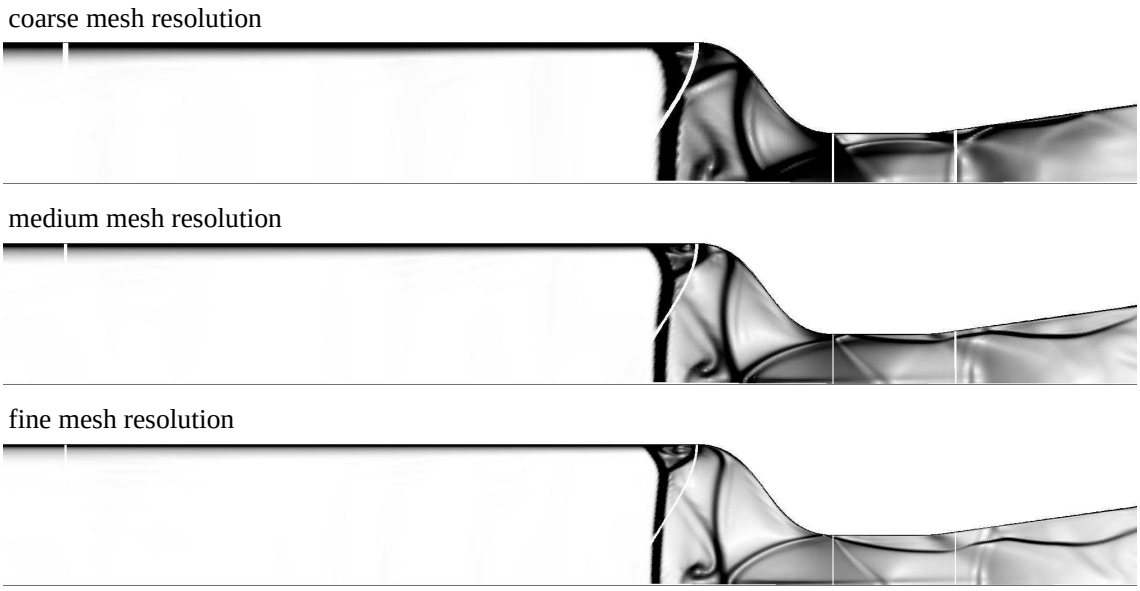

Figure 7.36: The effect of the mesh resolution on the modelling of the diaphragm rupture process. Numerical Schlieren contours are shown, comparing the shock reflection process for the coarse (top), medium (middle) and fine (bottom) meshes. The frames are shown at 2715, 2708 and 2700 μs respectively.

7.3.4 Interaction Between the Reflected Shock and the Boundary Layer

After the primary shock reflects from the end of the shock tube, it must move back upstream through the boundary layer. As was discussed in Section 2.1.6, the momentum deficient boundary layer material does not have enough monentum to cross the normal shock. Since it cannot cross the shock, boundary layer material builds up at the foot of the reflected shock and is carried upstream with it. The gas passing around this foot material passes through two oblique shocks, rather than the normal shock across the centre of the tube, and emerges from these shocks with a higher velocity than the core flow. This is combined with a venturi effect as the gas is accelerated through the contraction in area between the boundary layer material. The result is a jetting of gas near the walls of the tube which has, in the literature, been believed to be the cause of driver gas contamination, by jetting driver gas material as the contact surface reaches the reflected shock [228, 46].

This jetting process has a number of effects on the operation of the shock tube, introducing significant levels of fluctuations into the stagnation region, and contributing to the process of driver gas contamination. The process also leads to fluctuations of the properties behind the shock structure, and results in test flow noise.

Figure 7.37 shows contours of pressure in the region of the reflected shock interaction with the boundary layer. In the region behind the normal shock, the flow is stagnated; however, the flow that passes through the oblique shocks emerges from these shocks at a lower pressure. This gas influences the region between the ring of shock foot material by lowering the pressure of the gas in towards the centreline. The lower pressure gas in this region is stagnated by a series of shocks across the tube at the downstream end of the shock foot. These shocks are referred to in the literature as the pseudo shock train [144, 167]. This shock train means that as the structure moves upstream it leaves the gas essentially stagnated. The figure also shows that the pressure of the gas in the shock foot is in equilibrium with the surrounding gas.

Figure 7.38 shows contours of axial velocity in the region of the reflected shock interaction with the boundary layer. The stagnation of the gas through the normal shock is evident. The gas which moves through the oblique shocks emerges with a positive velocity. It continues to move with this velocity around the shock foot, until it is stagnated by the shocks at the tail end of the shock foot. This figure seems to suggest that, although gas is projected around the shock foot by the oblique shocks, any sustained jetting of the gas would be limited by the shocks at the tail end of

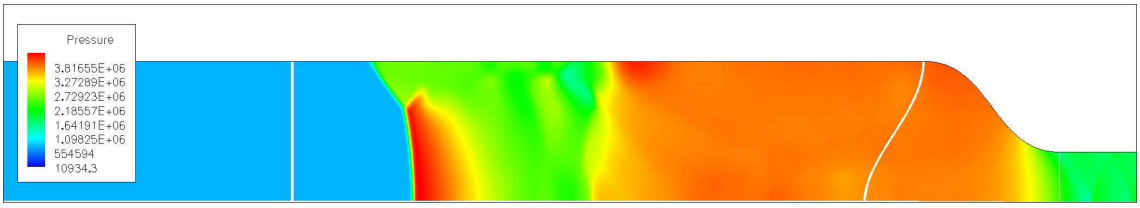

Figure 7.37: Contours of pressure in the region of the reflected shock interaction with the boundary layer.

the shock foot. The gas near the centreline of the tube has a positive velocity due to the influence of the gas from the oblique shocks, but this gas is also stagnated by the tail end shocks.

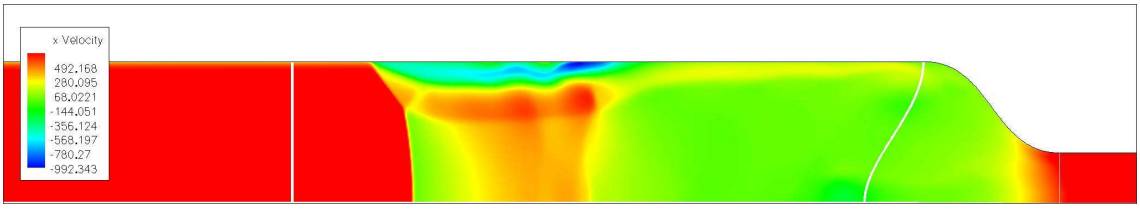

Figure 7.38: Contours of axial velocity in the region of the reflected shock interaction with the boundary layer.

The normal segment of the reflected shock is curved across the radius of the tube. This is due to the gradient of pressure on the downstream (towards the nozzle) side the shock. This gradient results from influence of the lower pressure gas which emerges from the oblique shocks.

Comparison of the Simulated and Experimental Interaction

The effect of the interaction between the reflected shock and the boundary layer is evident in the experimentally recorded supply pressure transducer traces. The leading arm of the lambda shock and the boundary layer material at the foot of the shock pass over the supply pressure transducer as they move upstream. The initial sharp rise in pressure is due to the passage of the leading arm of the lambda shock foot. Following this there is a gradual rise in pressure through the material in the shock foot to the post shock pressure level.

Figure 7.39 shows supply pressure transducer traces in the region of the interaction process. The Helium driving Nitrogen case simulations are shown, assuming laminar boundary layers on the left and with turbulent boundary layers on the right. Both the simulated and experimental pressure trace are shown for comparison.

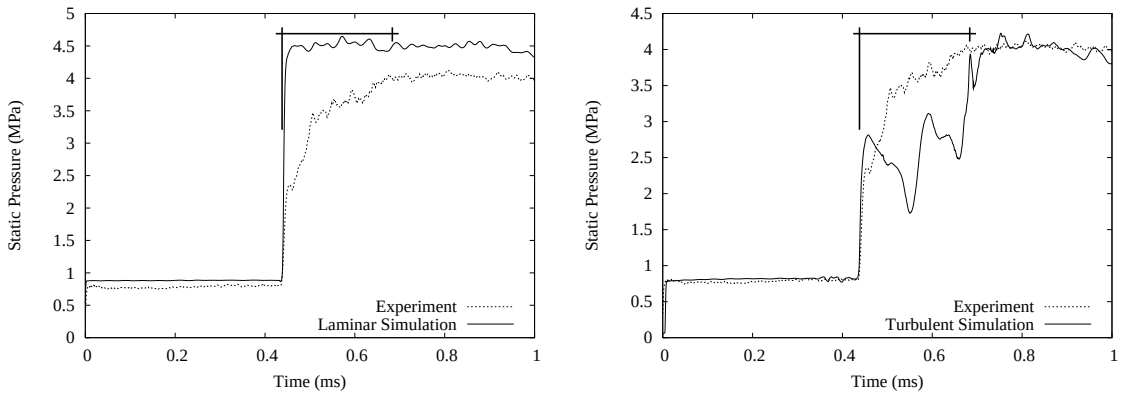

Figure 7.39: Supply pressure transducer traces in the region of the interaction between the reflected shock and the boundary layer. The period of time for which the shock foot is passing over the transducer is labelled on the figure. The Helium driving Nitrogen case is shown, with the laminar simulation shown on the left and the turbulent case shown on the right. The experimental and simulated traces are shown.

It can be seen in the pressure profile obtained using the simulation that assumes laminar boundary layers, that the boundary layer is not thick enough to cause significant bifurcation of the foot of the reflected shock. The laminar has a sharp profile with the pressure rising rapidly to the post shock pressure. This indicates that insufficient boundary layer material has been entrained at the foot of the shock. The over-estimation of the final shock pressure that is evident in the trace is unrelated to these aspects of the boundary layer interaction process.

In the trace from the turbulent simulation, the same sharp in rise resulting from the passage of the front arm of the lambda shock is evident; however, it is then followed by a sharp plateaued profile through the material at the foot of the shock, rather than the gradual rise in the experimental trace. This difference is caused by a difference in the pressure profile through the material in the foot of the shock. The Baldwin-Lomax eddy viscosity model [11] adds a component onto the viscosity to account for the turbulent fluctuations near the walls. This increased viscosity causes the boundary layers to be much thicker and, as a result, a thicker layer of gas cannot cross the normal shock, instead building up at the shock foot.

Despite the large deviations in the pressure traces, there are aspects of the interaction process that are reproduced much more accurately with the turbulent simulations. The pressure histories are recorded, in both the simulation and experiment, inside the boundary layer. This makes this trace particularly susceptible to the details of this interaction process and to the properties of the gas inside the shock foot. Even though the simulated pressure trace diverges from the experimental trace for the period while the foot material is passing over the transducer, there is still significant support given to the simulation being an accurate representation

of the flow:

1. the duration for which the shock foot passes over the transducer is accurately predicted by the simulation

2. the pressures before and after interaction are predicted accurately.

This difference in the traces is also evident in the literature, as was shown in Section 3.3. No simulations in the literature show a gradual rise in pressure through the shock foot material and all simulations show the same type of plateaued profile seen in these simulations.

The effect of the 5 mm width of the pressure transducer was investigated by using multiple history cells, covering the width of the transducer, in a simulation. The values from these history cells were averaged to obtain an average value for the area of the gauge. This had very little effect on the pressure trace.

An error in mounting height could also cause inaccuracy in recording the pressure trace. The mounting of the transducer was inspected and was found to have been mounted with a fine tolerance, making it unlikely that mounting height would have an influence on the trace.

The response time of the transducer was estimated to be of the order of 2 μs [211, 54]. This is unlikely to be responsible for an inaccurate profile of pressure through the shock foot as this takes a total of 250 μs to pass the transducer.

Effect of Turbulence on the Interaction

As was discussed in Section 2.1.3, the boundary layers on the shock tube walls are almost completely turbulent. Turbulence in the boundary layers It is known from other experimental studies, including the original paper on this process by Mark [141], that the interaction changes significantly with transition in the boundary layer; this was discussed in Section 2.1.6. It is believed that the differences observed between the experimental and simulated wall pressure profiles are caused by approximations regarding the simulation of turbulence.

The simulations solve the Reynolds Averaged Navier-Stokes (RANS) equations. These equations time average the fluid motion, removing any motion in the flow resulting from turbulent fluctuations. This is an important aspect of simulations, meaning that turbulent fluctuations in the boundary layers are not directly included in the simulations. The real turbulent boundary layer is composed of fluctuating turbulent eddies. Figure 2.6 demonstrates the magnitude of these fluctuations that would be expected. These turbulent eddies would enter the shock foot in an unsteady manner, unlike with the laminar boundary layer material. The additional

energy that these eddies carry may even allow boundary layer material to cross the shock intermittently. Comparison between the simulated and experimental pressure traces shows that the amount of material in the shock foot is predicted accurately; however, it is likely that the turbulent fluctuations in the boundary layer would have a significant effect of the properties of the gas in the shock foot. With turbulent boundary layer material entering the shock foot, the shock foot would be unsteady, and the resulting profile of pressure through this shock foot would, in detail, be different to that predicted by a simulation assuming a thick, but steady boundary layer.

In addition to the RANS approximation, the movement of gases in the simulations are restricted by the imposed axisymmetry, which does not allow any out of plane motion of the gases. At later times, large vortices are seen to be generated by the reflected shock interaction process. In the real flow these vortices would degenerate into turbulence through the process of vortex stretching.

Sequence of the Interaction

Figure 7.40 shows a sequence of numerical schlieren frames of the flow in the region of the flow in which the interaction of the reflected shock with the boundary layer is occurring. These frames follow on from the sequence of shock reflection process in Figures 7.34 and 7.35.

In the first frame of the sequence, material can be seen building up at the foot of the reflected shock almost immediately. As the reflected shock moves upstream, additional material builds up at the shock foot. The length of the separated region can be seen to increase along through the sequence. At later times vortices can be seen to be shed from the entrained material at the foot of the reflected shock. These vortices are shed as the mass reaches some critical level.

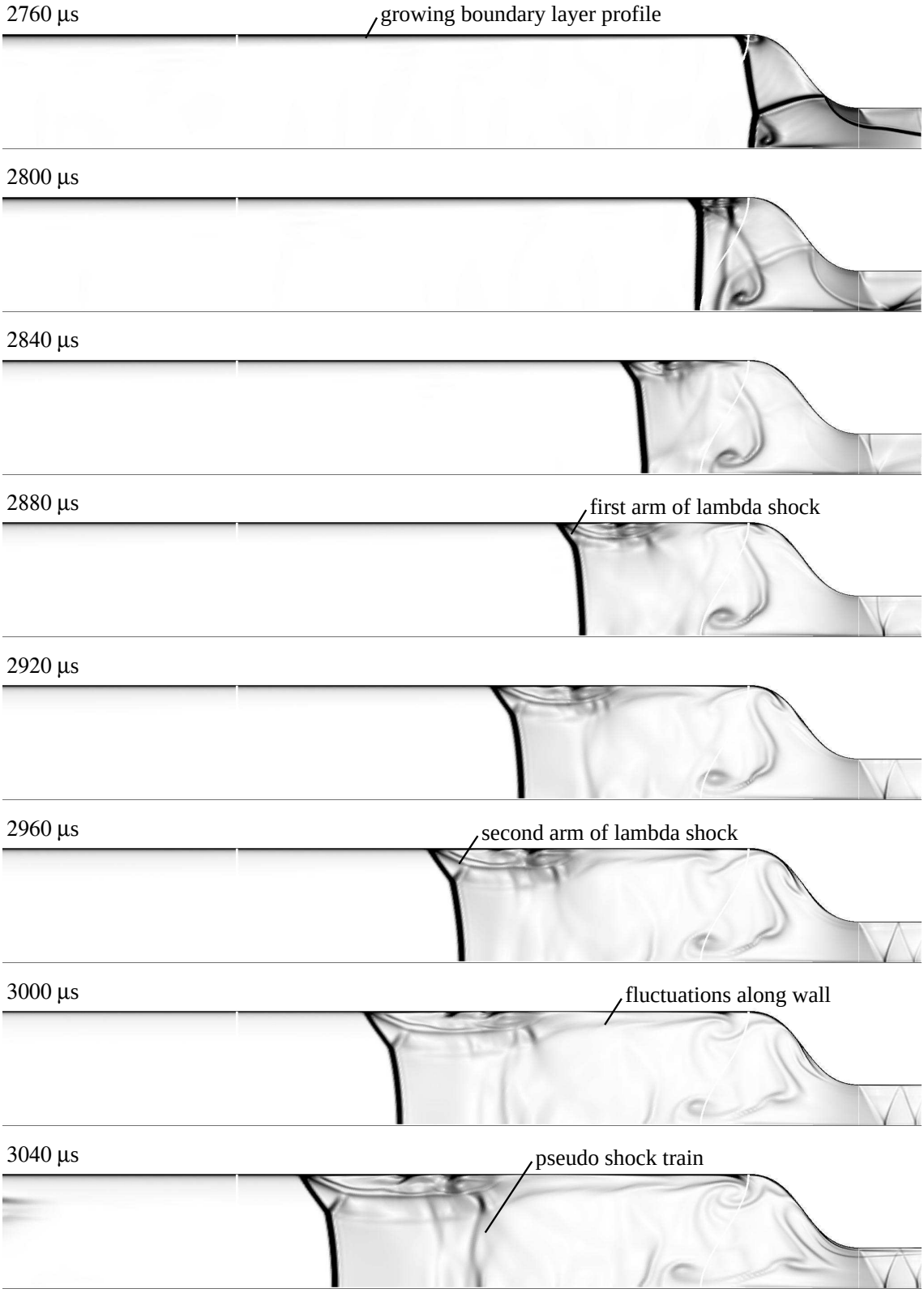

Figure 7.40: Sequence of numerical Schlieren images showing the interaction of the reflected shock with the boundary layer. The Helium driver case is considered. The frames start 2.760ms after the arrival of the incident shock at the supply pressure transducer and are at $40\,\mu$s intervals.

Influence of the Shock Reflection Process

The size of the interaction region recorded by the pressure traces is significantly different between the blanked end and the Mach 4 nozzle cases (with the Nitrogen driver). This difference is evident in both the experimental and simulated traces, in Figure 7.41 Apart from the detail of the pressure rise through the region, the simulations reproduce other important aspects of the experimental traces.

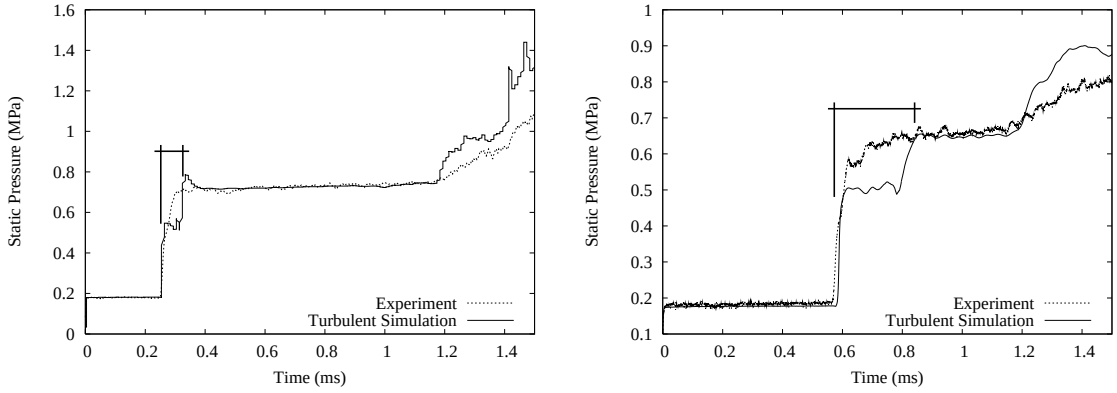

Figure 7.41: Supply pressure transducer traces for the Nitrogen driving Nitrogen cases with the blanked shock tube end (left) and the Mach 4 nozzle (right). The experimental and simulated traces are shown. The traces are focused on the region in which the shock boundary layer interaction is evident.

This is a significant result since the operating conditions are almost identical and so must result from the geometry differences. The shock reflection in the blanked end case is simple; however, in the nozzle case the reflection is complicated and the reflected shock is coalesces from at least three reflections of the shock from the nozzle surface. This shock formation occurs as the boundary layer interaction process has already begun. The reflection process from the nozzle surface is shown in Figures 7.34 and 7.35. Another difference caused by the geometry is the flow of material over the structure as the test gas is expanded through the nozzle mouth. This means that the flow, and the boundary layer with it, is expanded towards the nozzle, increasing the rate of flow into, and over, the shock foot. In addition to these differences, the distance that the reflected shock has travelled upstream before it reaches the transducer is greater in the Mach 4 nozzle case.

Figure 7.42 shows numerical Schlieren images of the interaction of the reflected shock with the boundary layer. The interaction resulting from reflection from the blanked end is shown on the top and resulting from reflection from the nozzle is shown on the bottom. The separation length is slightly shorter for the blanked end case. This does not completely account for the difference in the traces observed in Figure 7.41. The reflected shock travels upstream at least 10% faster in the

blanked end case. This is believed be a result of the flow of material into the nozzle not occurring in the blanked end case. The combination of the increased shock speed and the slightly shorted separation distance is responsible for the differences observed in Figure 7.41.

Figure 7.42: Comparison of numerical schlieren contour plots comparing the interaction of the reflected shock with the boundary layer following the reflection of the shock from the blanked shock tube end (top) and the nozzle (bottom)

Effect of Mesh Resolution

Figure 7.43 shows frames from the coarse, medium and fine meshes, demonstrating the effect of mesh refinement on the interaction process. These frames are from the Helium driving Nitrogen case. Comparison of these frames shows that there are no significant differences in the characteristics of the flow. As expected, sharper images are produced with the medium and fine meshes.

Figure 7.43: The effect of the mesh resolution on the modelling of the interaction of the reflected shock with the boundary layer. Numerical Schlieren images are shown, comparing the simulated flow for the coarse (top), medium (middle) and fine (bottom) meshes. The frames are shown at 300 μs

7.3.5 Jetting of Gas Through the Reflected Shock

The jetting of gas through the bifurcated foot of the reflected shock is an important feature of the operation of a shock tunnel. This process results in the introduction of significant noise into the test flow and plays a role in the contamination of the test flow with driver gas. Sudani and Hornung [228] state: "It is widely accepted that the driver gas (normally Helium or a mixture of Helium and Argon) passing through the bifurcated foot of the reflected shock causes early contamination of the test gas in the reservoir condition".

It is known, from sources in the literature and from this study, such as Figure 7.38, that the gas passes through the oblique shocks at the foot of the shock with a higher axial velocity than the gas moving along the centre of the tube, through the normal shock. The degree of sustained jetting of gas cannot be determined directly from these sources, and the conclusions reached in the past, on the degree of sustained jetting has largely been based on indirect evidence of the jetting, such as streamline plots.

Temperature contours, as used in the visualisation of the interaction by Wilson, Sharma and Gillespie [247], are sensitive to the jetting of the test gas, before the contact surface interaction. In the figures showing this interaction, the jetting of gas showing temperature differences is shown, both before and after the contact surface interaction. As a result the interaction resulting from the arrival of the contact surface is difficult to distinguish from the previous jetting. Streamlines and vectors, as used by Weber, Oran, Boris and Anderson Jr. [240], are difficult to interpret for late time evolution of the transient flow fields studied here. In addition, in many previous studies, the generation of vorticity in the simulations will be underestimated due to the planar contact surface profile used in those simulations.

It is not certain whether this jetting persists far past the shock structure, as when the gas passes behind the foot material, it passes through the oblique shock train and reduces in speed. In order to provide a direct measure of the jetting of driver gas through the bifurcated foot of the reflected shock, two strips of gas across the radius of the shock tube were tagged, one ahead of and one behind the reflected shock. These two strips of gas were then allowed to evolve in time as the reflected shock structure moves upstream. This part of the investigation does not include the interaction of the reflected shock with the contact surface; it is only concerned with the jetting of test gas material as the reflected shock moves through the test gas slug. The interaction of the reflected shock with the contact surface, and its subsequent effect on the jetting of gas through the bifurcated shock foot, will be discussed later in Section 7.3.7.

Sequence of the Jetting

Figures 7.44 and 7.45 show the sequence of the flow, starting with the reflected shock part way moving upstream through the test gas slug. Numerical schlieren is shown in the upper half of the frame and the mass fractions of the two tagged gases (in blue) is shown on the lower half the frame. This sequence is taken from one of the full shock tunnel simulations of the Helium driving Nitrogen case.

The initial strips of gas can be seen in the first frame at 3000 μs. The strips are slightly curved due to the body fitted mesh that they were aligned with.

In the frame at 3020 μs, the front strip can be seen to enter the shock structure. A slight relative movement of the gas that passes through the oblique shocks, to the gas that passes through the normal shock is evident; however, as the frames progress, no sustained jetting of any part of the strip is evident. By the frame at 3140 μs, the front strip of gas has completely passed through the shock structure and has emerged from the pseudo shock train. The section of the strip near the wall is slightly downstream of the rest of the strip, but there is little remaining jetting motion.

As the sequence progresses to later times, the movement of the wall gas towards the nozzle is largely driven by the flow of the test gas into the nozzle. The movement of the back strip of gas is more strongly influenced by the drainage into the nozzle than the front strip.

In this simulation a slight relative movement of the gas near the wall, relative to gas near the middle is evident; however, this is not a sustained jetting of the degree that would cause this effect to be responsible for driver gas contamination. The start of the interaction of the reflected shock with the contact surface is evident in these frames. It will also be shown in Section 7.3.7 that the generation of vorticity at the interaction of the reflected shock with the contact surface acts to stop the jetting. These two observations mean that it is highly unlikely that jetting of gas through the shock foot is mainly responsible for driver gas contamination.

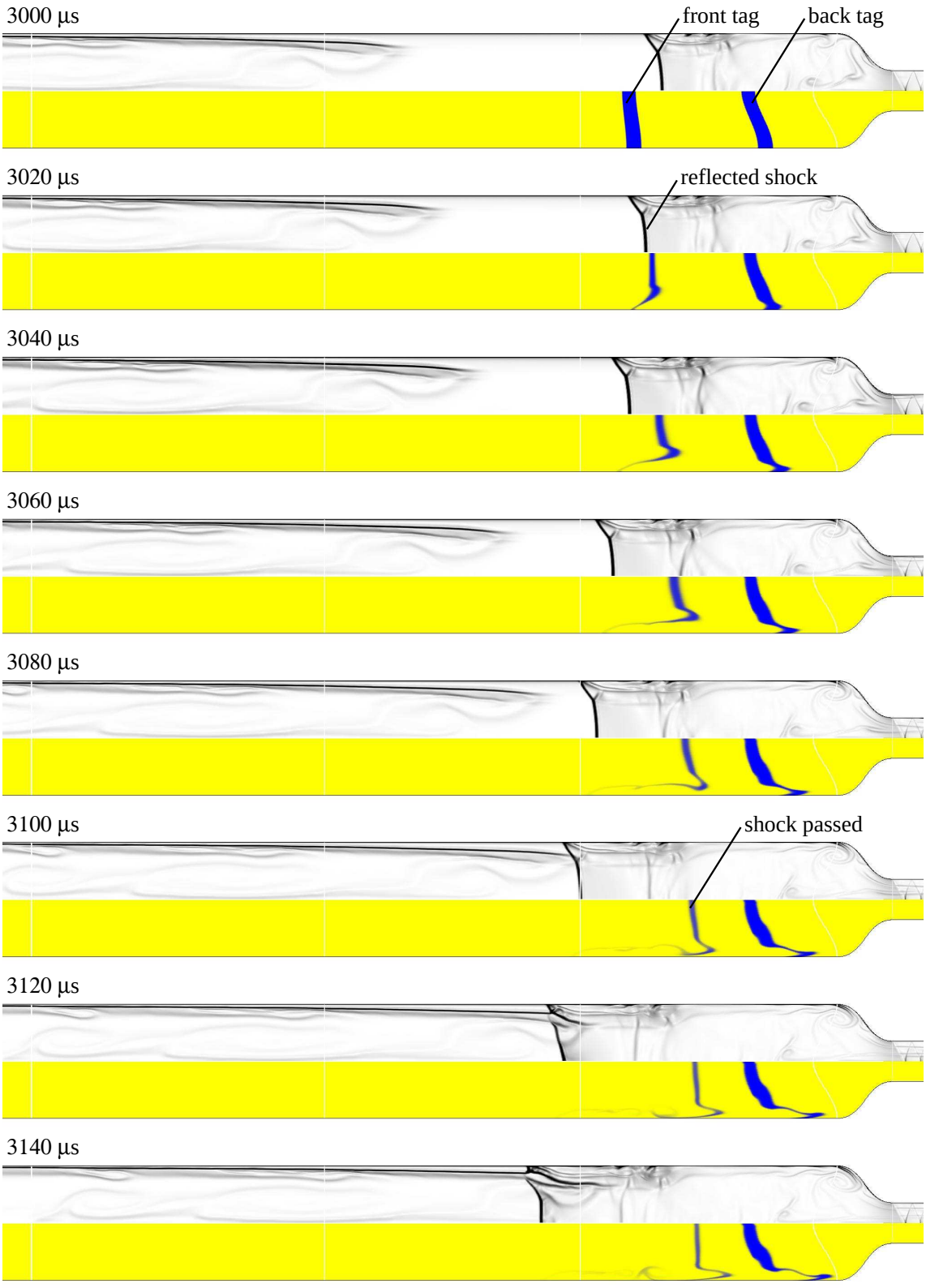

Figure 7.44: Part one of the sequence of numerical Schlieren images (upper half of frames) and tagged mass fractions (lower half of frames) showing the jetting of test gas caused by the reflected shock with the boundary layer. Helium driver with the Mach 4 nozzle.

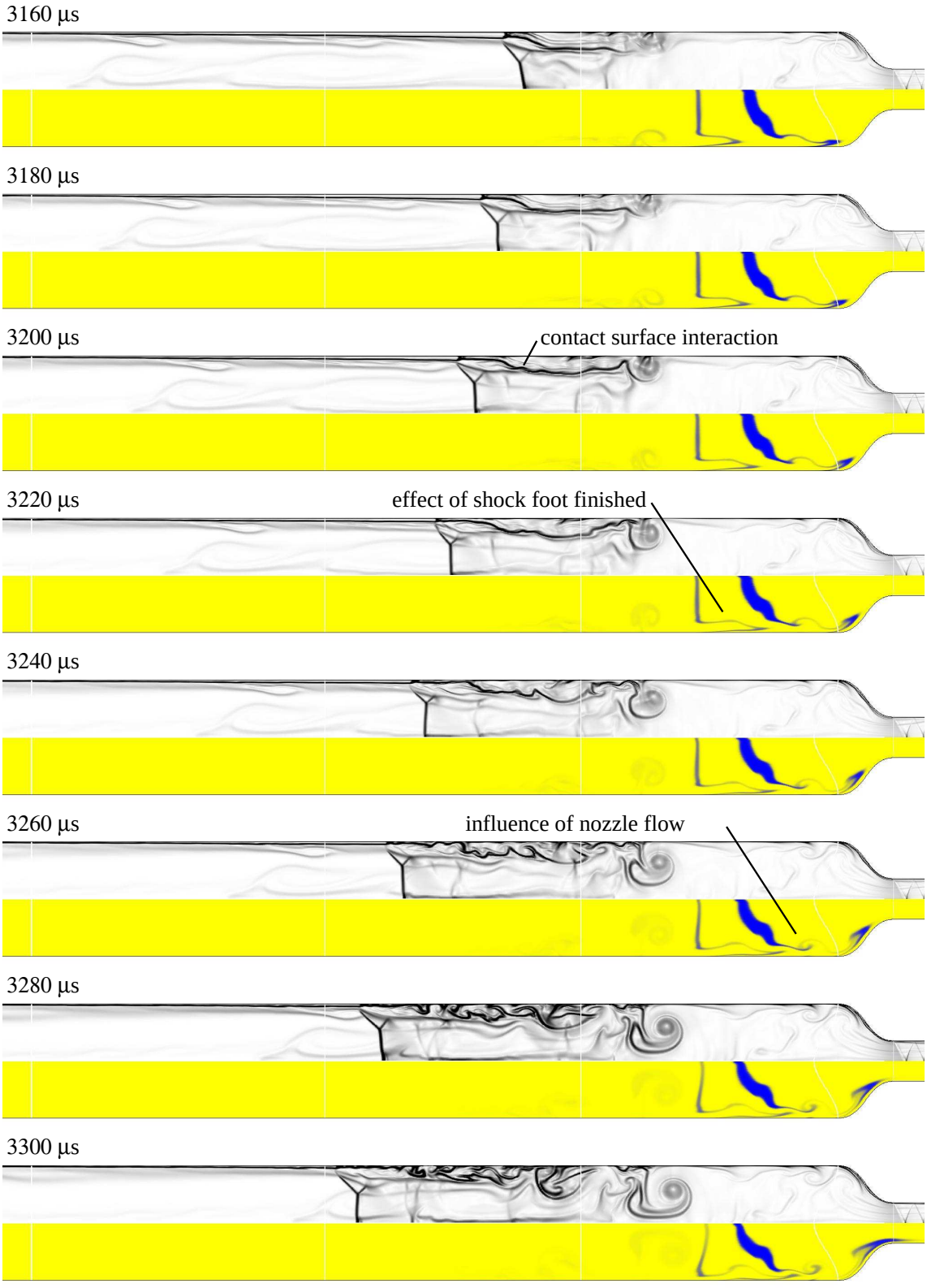

Figure 7.45: Part two of the sequence of numerical Schlieren images (upper half of frames) and tagged mass fractions (lower half of frames) showing the jetting of test gas caused by the reflected shock with the boundary layer. Helium driver with the Mach 4 nozzle.

7.3.6 Contact Surface Evolution and Characteristics

Unlike the shock, which rapidly becomes planar, the contact surface is not inherently stable and does not become planar; the shape of the contact surface continues to deform as it propagates along the shock tube. This evolution is driven by viscous drag from the boundary layers and by its stability properties. The characteristics of the contact surface play an important role in simulations aiming to predict driver gas contamination.

Effect of Diaphragm Rupture

The contact surface evolution is driven initially by the rupture mechanics of the primary diaphragm, as was discussed in Section 7.3.1. In the diaphragm rupture process, the contact surface is repeatedly affected by transverse waves, depositing vorticity with each pass. The resulting gradient in the velocity of the driver gases across the tube, with the gas near the wall moving significantly faster than gas near the centreline, causes an outward mixing motion at the contact surface. Both of these effects combine to promote deformation at the contact surface.

Figure 7.14 shows the experimentally recorded contact surface profile across the tube with the iris based rupture model and Figure 7.13 shows the profile without this model. The shape of the contact surface recorded by the heat flux rake shows that it is more curved than would be caused by the boundary layer. If the contact surface was influenced by only the boundary layers, then the profile across the tube would be rounded through the width of the boundary layers, with a planar profile across the rest of the tube. This is not the profile that is recorded. It is the combined action of the initial deformation and the mixing along the tube that is believed to result in the contact surface shape recorded in the experiments.

Stability of the Contact Surface

The flow of test gas into the boundary layers on the walls of the shock tube can, under some conditions, cause the contact surface to accelerate as it moves along the length of the shock tube [207]. It was shown for the two operating conditions considered in this thesis, in Figures 7.6 and 7.19, that the contact surface accelerates along the length of the shock tube. This is true more for the Helium driving Nitrogen case.

This acceleration means that the contact surface will be unstable, due the compressible form of the Rayleigh-Taylor instability, for flow configurations in which the density of the driver gas is less than the density of the driven gas at the contact

surface [26]. This has an effect the same as a heavy gas being on top of a light gas under a gravitational acceleration.

Figure 7.46 shows profiles of density along the shock tube centreline from the primary diaphragm to past the incident shock for the Nitrogen driving Nitrogen case (left) and the Helium driving Nitrogen case (right). These traces demonstrate that, although the Nitrogen driving Nitrogen case has a higher density in the driver gas making it stable, the Helium driving Nitrogen case has a discernible decrease in density across the contact surface making it unstable as it progresses along the tube.

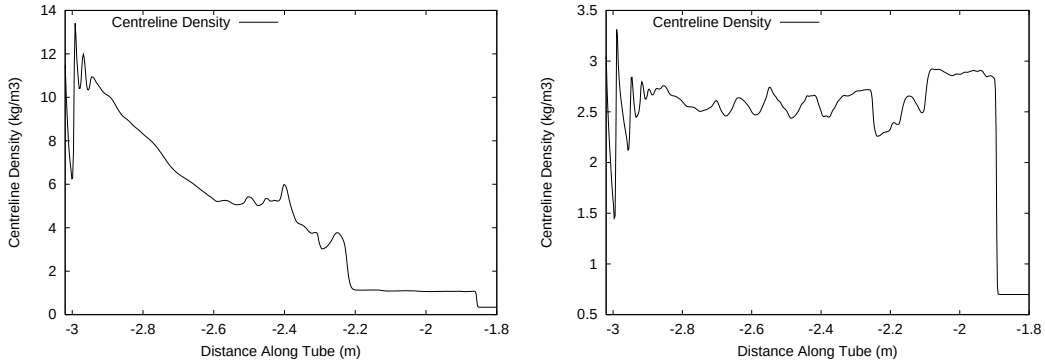

Figure 7.46: Centreline density profile from the primary diaphragm (at -3.02 m) to past the incident shock. The Nitrogen driving Nitrogen case is shown on the left and the Helium driving Nitrogen case is shown on the right.

This instability is evident in the shape of the contact surface as it progresses along the tube. Figure 7.47 shows the resulting contact surface shape for the Nitrogen driving Nitrogen case. The effect of the diaphragm rupture mechanics on the contact surface evolution is also shown in this figure (as is discussed in Section 7.3.1. It can be seen that this contact surface is stable and it reaches a steady profile along the tube. This was suggested by the combination of Figure 7.46 and Figure 7.5.

Figure 7.48 shows the resulting contact surface shape for the Helium driving Nitrogen case. It can be seen in this figure that this contact surface is unstable, as was suggested by combination of Figure 7.46 and Figure 7.18. The dense driven gas can be seen to be penetrating the light driver gas with the fingers of heavy gas that are characteristic of the Rayleigh-Taylor instability as the gas travels along the tube.

The outward mixing motion caused by the diaphragm rupture model shows a similar structure to the Rayleigh-Taylor instability. This makes the two effects difficult to distinguish from one another; however, the relative levels of mixing evident in Figures 7.48 and 7.47 demonstrate the two effects occurring in isolation.

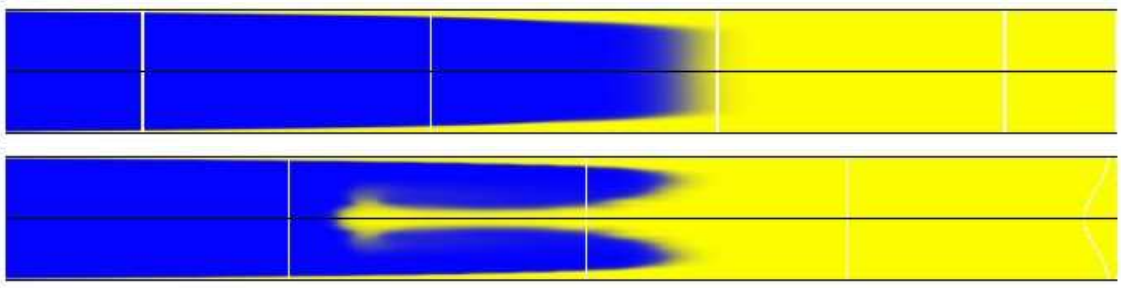

Figure 7.47: Contact surface shape for the Nitrogen driving Nitrogen case resulting from the ideal removal of the diaphragm (top) and the iris based diaphragm rupture model (bottom).

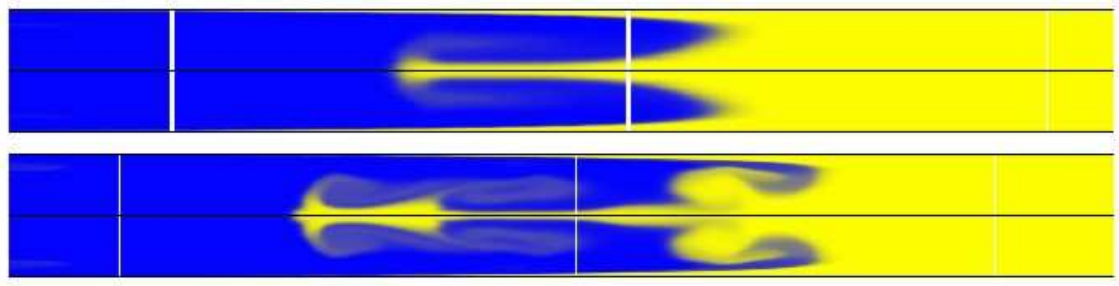

Figure 7.48: Contact surface shape for the Helium driving Nitrogen case resulting from the ideal removal of the diaphragm (top) and the iris based diaphragm rupture model (bottom).

Effect of Mesh Refinement

It is important to identify any mesh dependent behaviour of this development. Figure 7.49 shows the resulting contact surface from the coarse (top), medium (middle) and fine (bottom) meshes, for the Helium driving Nitrogen case. As the mesh is refined there is a noticeable increase in the level of mixing; however, the same general characteristics are evident.

Limitations of the Simulation

The contact surface deforms under the influence of the diaphragm rupture (through the generation of vorticity and the radial velocity gradient), and under the influence of drag from the boundary layers and its inherent stability or instability. Much of the physics leading to the characteristics of the contact surface are modelled; however, there are some important physical processes, such as free stream turbulence and molecular diffusion that are not. It is believed that these effects may be responsible for the shape of the contact surface being too sharply defined.

The experimental results indicate that there is more mixing at the contact surface

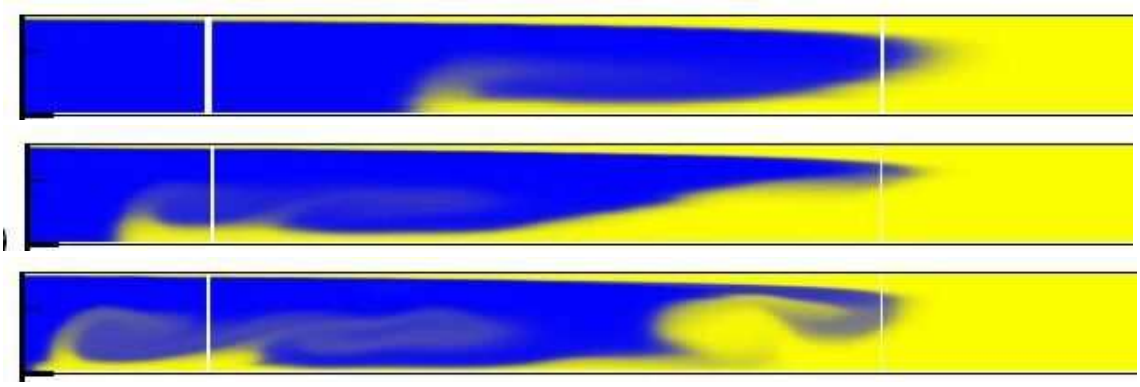

Figure 7.49: Effect of mesh refinement on the contact surface evolution and resulting shape. The coarse mesh is shown on the top, the medium mesh in the middle and the fine mesh on the bottom. The simulation of the Helium driving Nitrogen case is used.

than in the simulations. This difference between the simulation and experiment is evident in:

1. the contact surface arrival measured by the rake of heat flux probes (evident in the Nitrogen driving Nitrogen with the blanked end case)

2. the sharply defined arrival of tailoring waves at the pressure transducer (evident in both Nitrogen driving Nitrogen cases)

3. the heat flux gauge trace, which indicates mixing of the contact surface both forward and backward from the position that it is located in the simulation (evident in all cases)

It was shown in Figure 7.14 that the simulations, incorporating the effect of the diaphragm rupture mechanics, reproduced the mixing length at the contact surface. The levels of mixing observed in the simulated contact surface would result in the acceleration of mixing and the generation of turbulence. It has been observed experimentally that the contact surface is predominantly a region of turbulence, engulfing significant parts of both the driver and driven gases [255]. It is therefore reasonable to expect that the differences observed in the shape of the contact surface result from the turbulent, mixed profile that would be present in the real shock tunnel, but result from the two physical processes that are not modelled in the simulation. Although the effect of turbulence in the boundary layers is accounted for by an eddy viscosity model [11], no attempt has been made to account for turbulence in the core flow following the incident shock.

Sequence of Images of the Contact Surface

Figure 7.50 shows the evolution of the contact surface for the Nitrogen driving Nitrogen case. It can be seen in this sequence that the evolution of the contact surface is driven by the mixing initiated by diaphragm rupture. The outward motion imposed on the outer gas forces it forward along the tube wall. As this outer gas pushes forward, it folds back on itself and continues to move forward as a loop of gas. There is no indication of further instability in the contact surface.

Figure 7.51 shows the evolution of the contact surface for the Helium driving Nitrogen case. In addition to the mixing caused by the diaphragm rupture model, the effect of the Rayleigh-Taylor instability is evident in the evolution of this contact surface. The outward flow of material resulting from the diaphragm rupture is not as defined as in the Nitrogen driving Nitrogen case and the region of mixing is seen to be dominated by the penetration of the heavy driven gas into the light driver gas along the centreline.

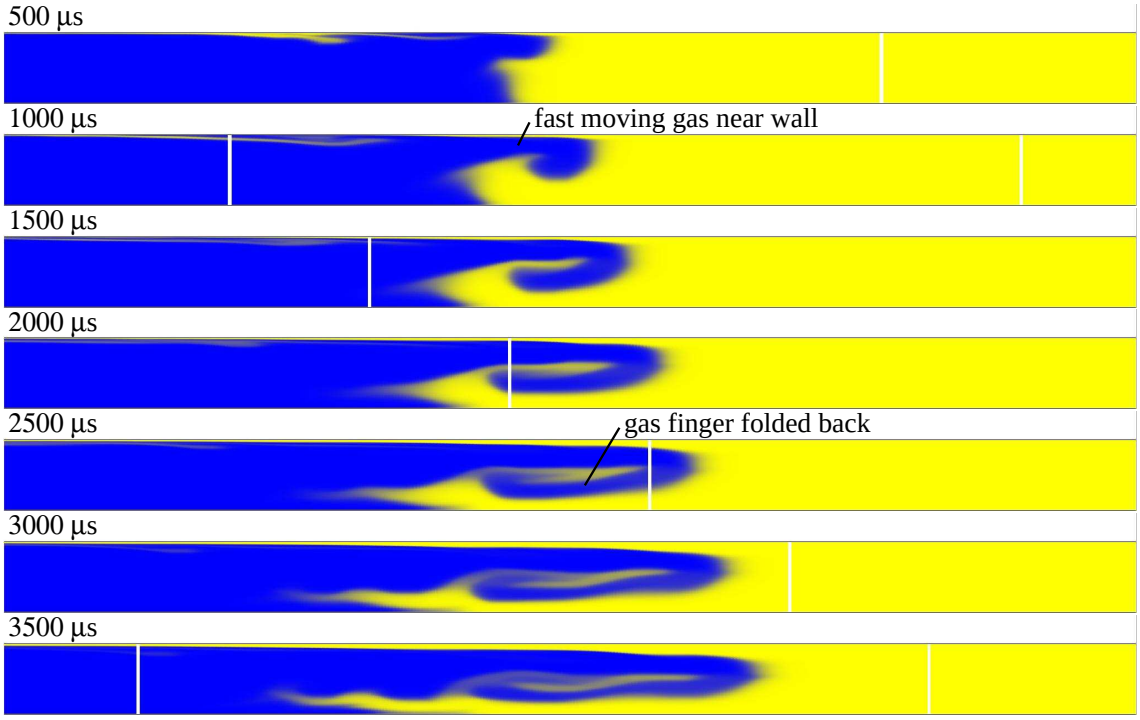

Figure 7.50: The evolution of the contact surface along the shock tube for the Nitrogen driving Nitrogen case.

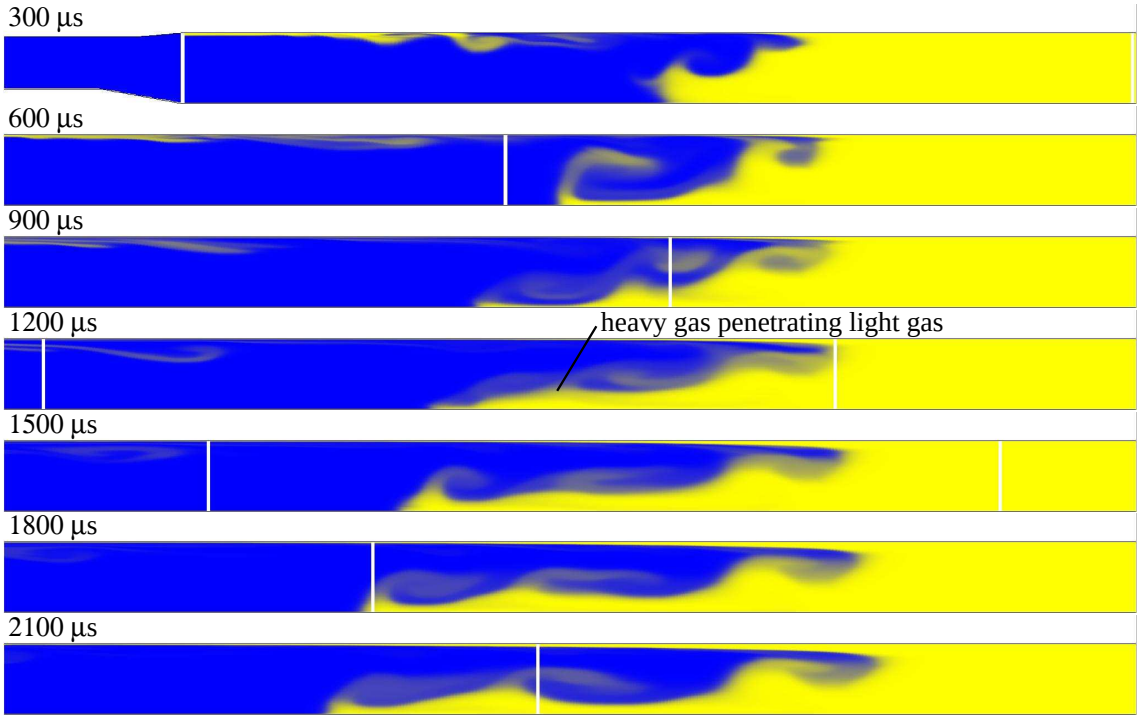

Figure 7.51: The evolution of the contact surface along the shock tube for the Helium driving Nitrogen case.

7.3.7 Simulations of Shock Contact Surface Interaction

The representation of the interaction between the reflected shock and the contact surface is crucial to the accurate simulation of a reflected shock tunnel. In addition to producing tailoring waves and introducing fluctuations to the nozzle supply region, this interaction process can also result in the projection of driver gas into the test flow.

The modelling of shock tunnels cited in the literature have only modelled the end of the shock tube. The assumptions resulting from only modelling part of a facility have a significant effect on these simulations. Figure 7.52 shows a schematic representation of the interaction process as it was modelled by Wilson, Sharma and Gillespie [247]. An important feature evident in this schematic is the representation of the contact surface as a planar, discontinuous interface. Similar assumptions were also made in other studies, such as Chue and Eitelberg [45]. The focus of these simulations were on the jetting of gas through the foot of the reflected shock and were relatively unconcerned with the characteristics of the contact surface.

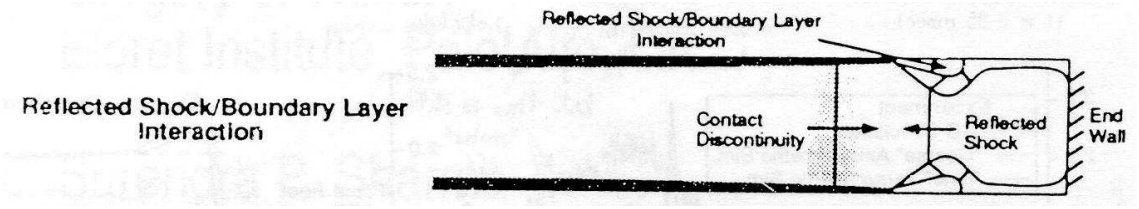

Figure 7.52: Schematic representation of the interaction process as modelled by Wilson et al. [247].

The present simulations show an interaction that occurs between the evolved contact surface and the reflected shock following its interaction with the boundary layer. The contact surface shape is dependent on its initial properties, from the rupture of the primary diaphragm, and its stability and resulting evolution along the shock tube, as was discussed in Section 7.3.6. The characteristics of the reflected shock are a result of the facilities operating conditions, the level of attenuation along the shock tube, the reflection process from the end of the shock tube and its interaction with the boundary layer. The process affecting the reflected shock have been discussed throughout Section 7.3.

The further interaction of the reflected shock with the contact surface can be characterised as a form of the Richtmyer-Meshkov instability [25]. The flow following this interaction is dominated by the effects of this instability. Its driving mechanism can be described in terms of the generation of vorticity at the interaction.

The level of tailoring of the contact surface defines much of how the interaction of the contact surface with the reflected shock occurs. The interaction resulting from the two levels of tailoring simulated in this thesis, one over-tailored and the other roughly tailored, will be discussed separately.

Over-tailored Interaction

Figures 7.53, 7.54 and 7.55 show a sequence of the interaction of the contact surface with the reflected shock for the Nitrogen driving Nitrogen case. Numerical Schlieren is shown in the upper half of the frames and mass-fractions are shown in the lower half of the frames (with driver gas in blue and driven gas in yellow). This interaction is with the over-tailored contact surface.

The first frames show the incident contact surface and the bifurcated reflected shock. The effect of the diaphragm rupture model has resulted in the shape of the contact surface. The interaction begins at the instant of the second frame, at 4700 μs. In these frames a series of vortices shed from the material at the foot of the reflected shock are evident. These vortices are shed before the contact surface interaction begins.

The driver gas moves predominantly through the oblique shocks near the shock tube walls. This is partly due to the shape of the contact surface, but this also occurs for simulations without the diaphragm rupture model and, therefore, with contact surfaces with more planar shapes. In the frame at 5000 μs, the driver gas can be seen moving along the walls; however, this gas quickly moves back in towards the centreline of the tube, as can be seen in the frame at 5200 μs. This gas continues to move along the centreline of the shock tube and towards the test flow.

The generation of a strong vortex at the head of the driver gas can be seen to accelerate a some of the gas ahead of the main gas body. It has been observed by Sudani and Hornung [228] that driver gas prematurely arrives in the test flow for over-tailored conditions in the T5 shock tunnel. This premature arrival is potentially caused by the same vorticity driven mechanism that is observed here. In the last frame, at 6900 μs, the main body of the driver gas is about to reach the nozzle. By this time, the driver gas accelerated by the strong vortex has been in the test flow for at least 0.5 ms.

This interaction process results in a complex structure to the reflected shock as it moves upstream through the driver gas. This shock structure has been described as the pseudo shock train [144], and can be seen to be forming in the frame at 4800 μs. The structure becomes very complex through the frames up to 6100 μs The late time evolution of the flow would be dominated by turbulence.

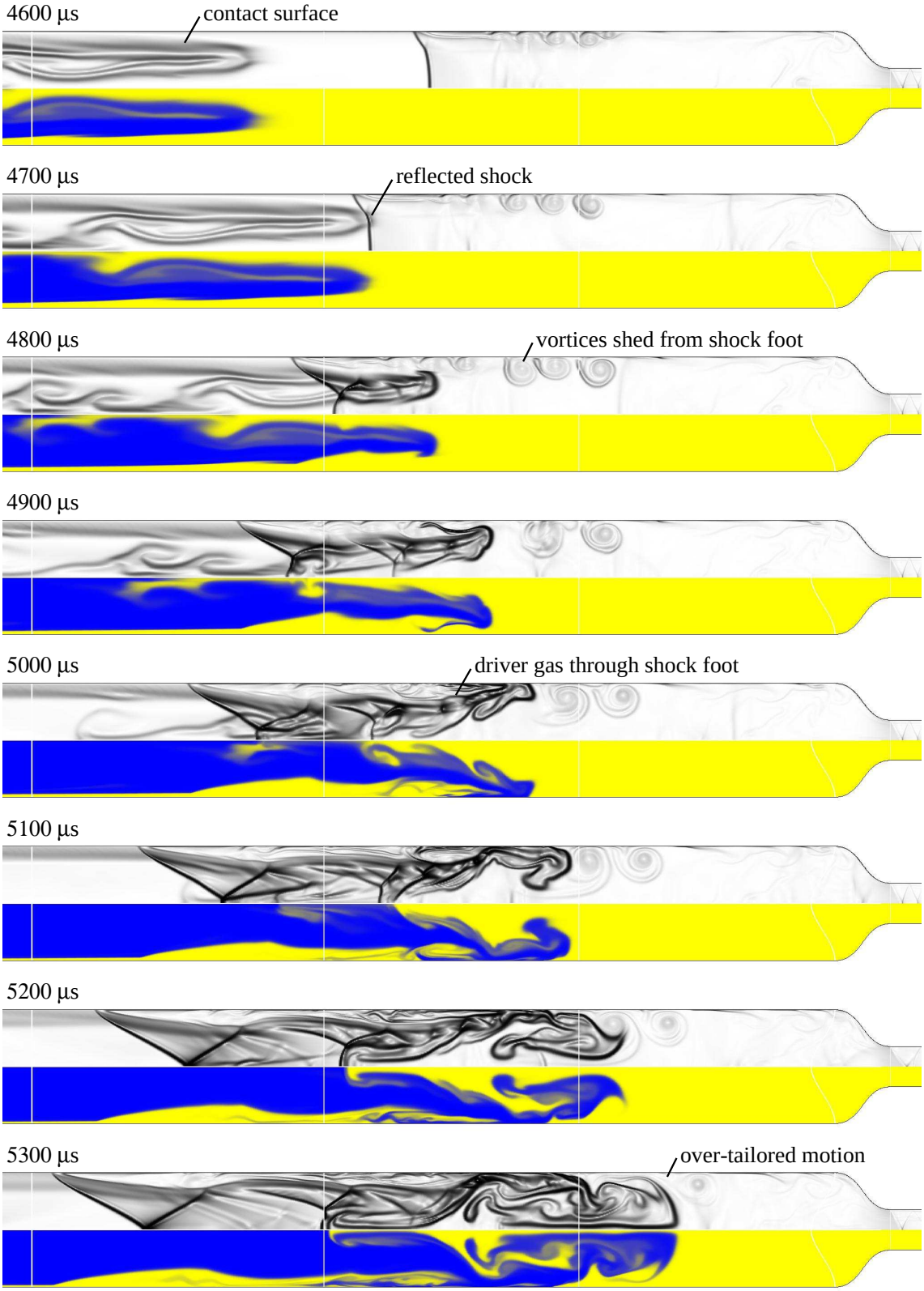

Figure 7.53: Part one of the sequence of numerical Schlieren images (top) and Driver gas mass fraction (bottom) showing the over-tailored interaction of the reflected shock with the contact surface. Nitrogen driving Nitrogen with the Mach 4 nozzle.

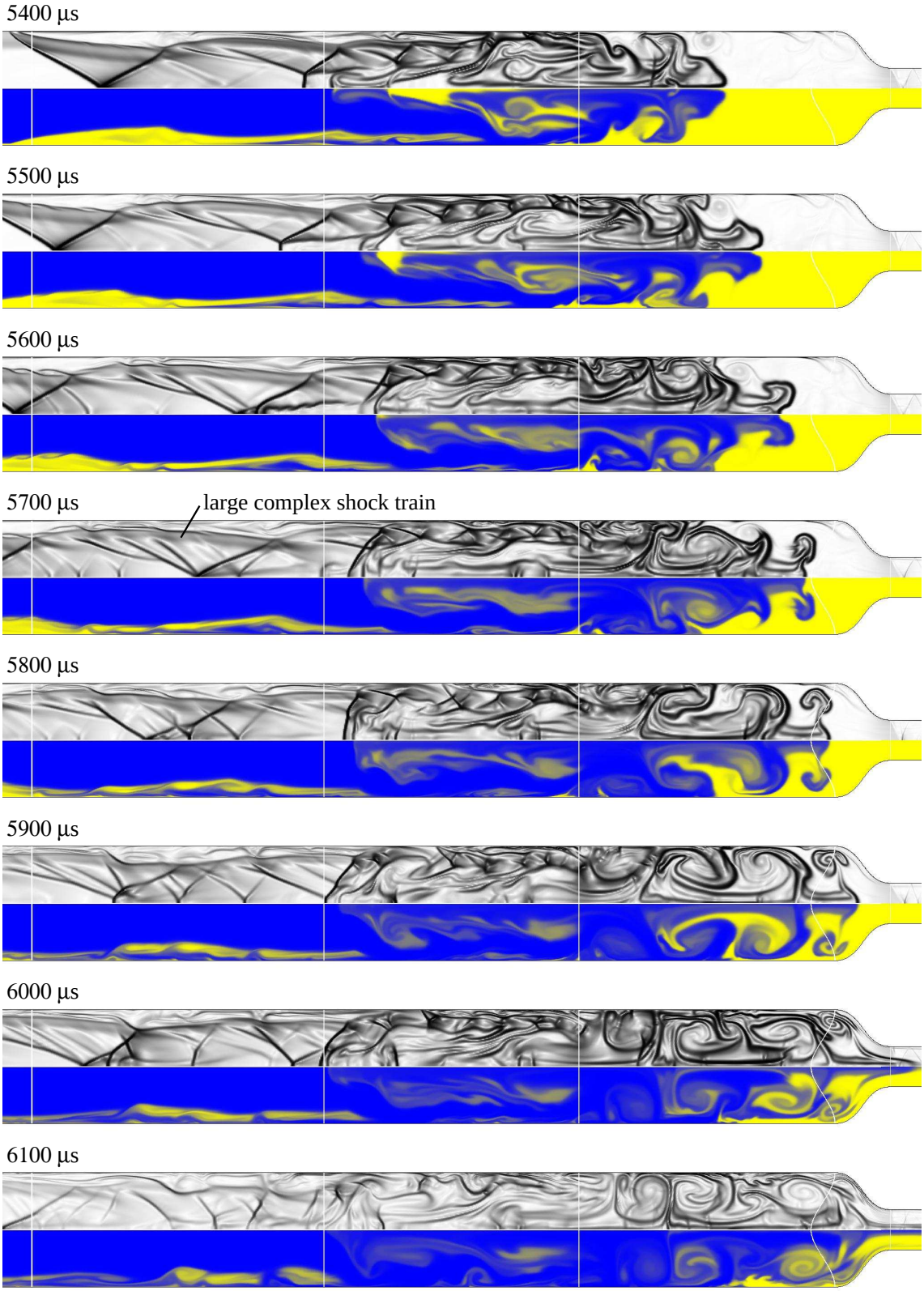

Figure 7.54: Part two of the sequence of numerical Schlieren images (top) and Driver gas mass fraction (bottom) showing the over-tailored interaction of the reflected shock with the contact surface. Nitrogen driving Nitrogen with the Mach 4 nozzle.

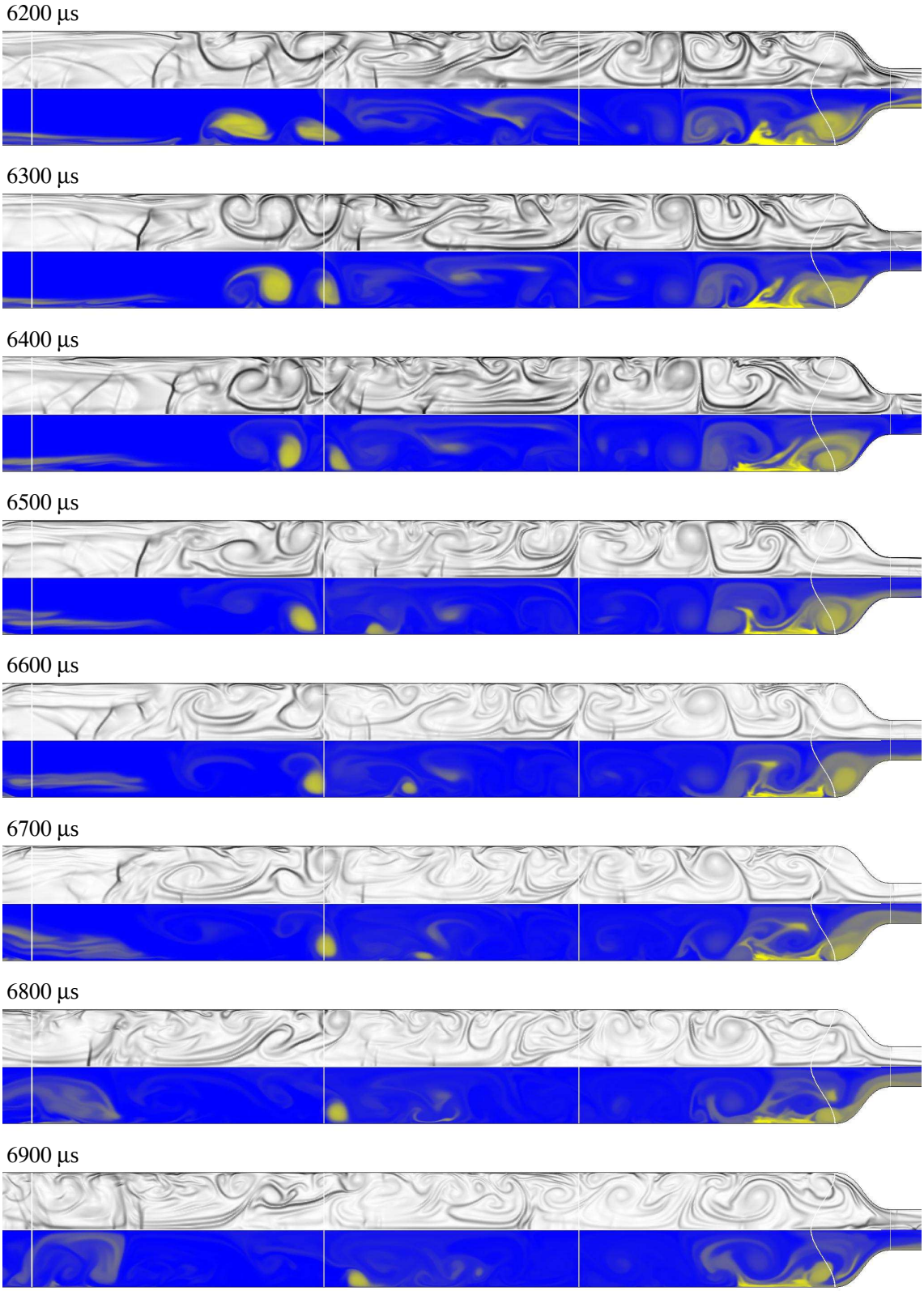

Figure 7.55: Part three of the sequence of numerical Schlieren images (top) and Driver gas mass fraction (bottom) showing the over-tailored interaction of the reflected shock with the contact surface. Nitrogen driving Nitrogen with the Mach 4 nozzle.

Roughly-Tailored Interaction

Figures 7.56, 7.57 and 7.58 show a sequence of the interaction of the reflected shock with the contact surface for the Helium driving Nitrogen case. Numerical Schlieren is shown in the upper half of the frames and mass-fractions are shown in the lower half of the frames (with driver gas in blue and driven gas in yellow). This interaction is with the roughly tailored contact surface.

The first frame, at 3100 μs, shows the incident contact surface and the reflected shock at the starting instant of their interaction. The contact surface can be seen to be significantly affected by its evolution through the shock tube. As with the over-tailored case, the driver gas can be seen to move predominantly through the oblique shocks at the foot of the reflected shock. A vortex is shed from the shock foot ahead of the contact surface.

A significant amount of vorticity is deposited in the contact surface as it moves through the reflected shock. When the reflected shock, laden with vorticity reaches the trailing edge, the vortex spins up in the wake of the shock foot. The vortex ring is formed by the vorticity generated through the shock interaction and is contributed to by its formation in the wake of the upstream motion of the material in the shock foot. The initial formation of this vortex is evident in the frames from 3240 μs through 3400 μs.

The motion caused by this vorticity is in a direction as to stop the jetting through the shock foot; this was also observed by Chue [43]. This is a tailored contact surface interaction and the driver gas does not continue to move downstream immediately following this interaction.

A strong vortex is seen to be forming the head of the tailored driver gas. It becomes more defined as it is drawn towards the centreline of the shock tube and accelerates axially along the shock tube towards the nozzle and the test flow. This vortex moves along the shock tube centreline with constant axial velocity through the frames from 3420 μs to the end of the sequence. The rest of the gas is shown to be stopped by the reflected shock, as would be expected with the tailored mode of operation. Significant vortical motion is also evident in this, otherwise stationary, gas. The vortex that was seen to be shed from the foot of the reflected shock can be seen to move along the shock tube ahead of the driver gas vortex.

This vortex and its effect on contamination of the test flow will be discussed in Section 7.3.9, which includes a sequence of the motion of the driver gas following this sequence.

As with the Nitrogen driving Nitrogen case, the late time evolution of the flow would be dominated by turbulence.

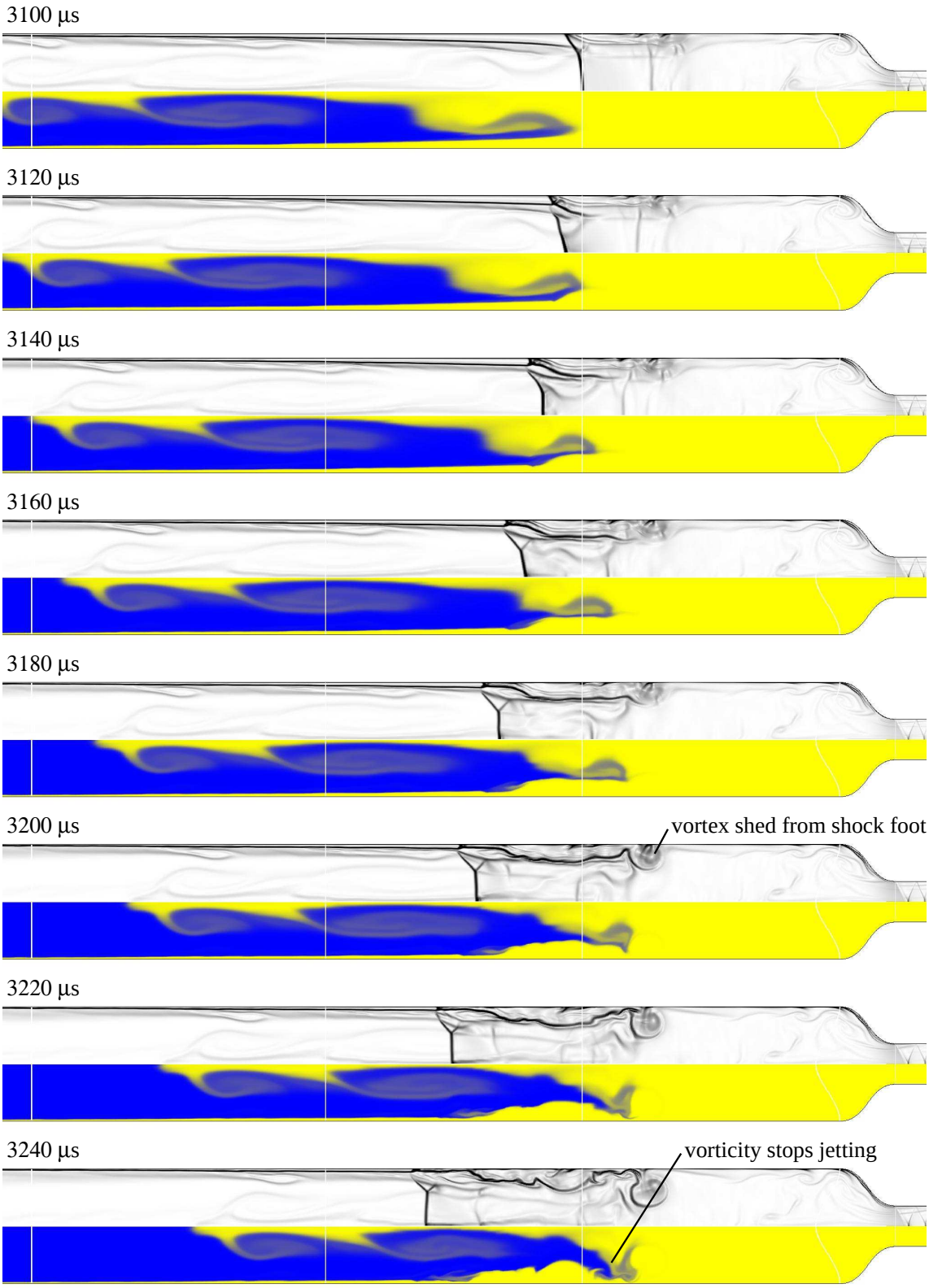

Figure 7.56: Part one of the sequence of numerical Schlieren images (top) and Driver gas mass fraction (bottom) showing the tailored interaction of the reflected shock with the contact surface. The Helium driver case simulation is used. The frames start at 3.10 ms and are at 20 μs intervals.

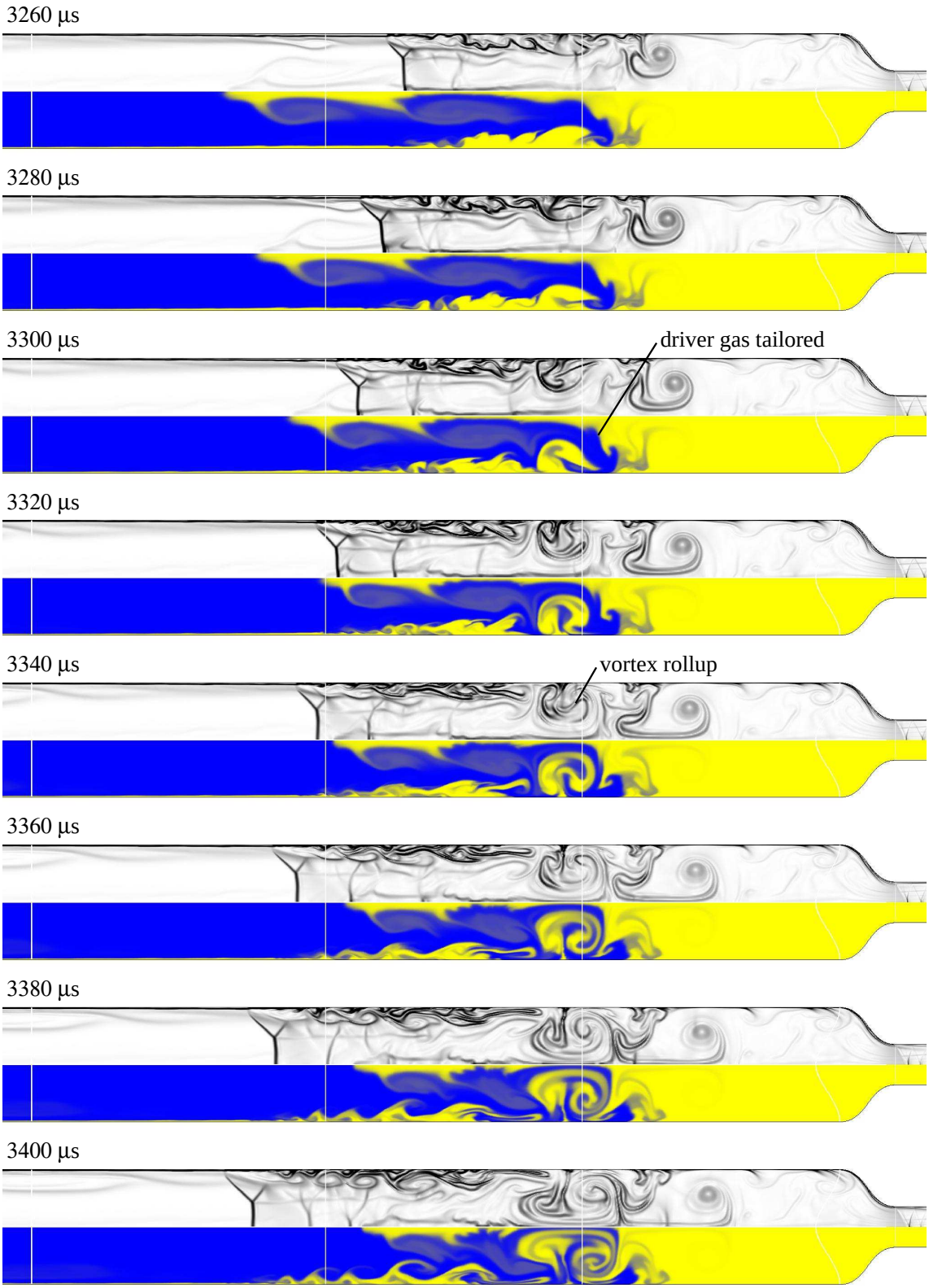

Figure 7.57: Part two of the sequence of numerical Schlieren images (top) and Driver gas mass fraction (bottom) showing the tailored interaction of the reflected shock with the contact surface. The Helium driver case simulation is used. The frames start at 3.26 ms and are at 20 μs intervals.

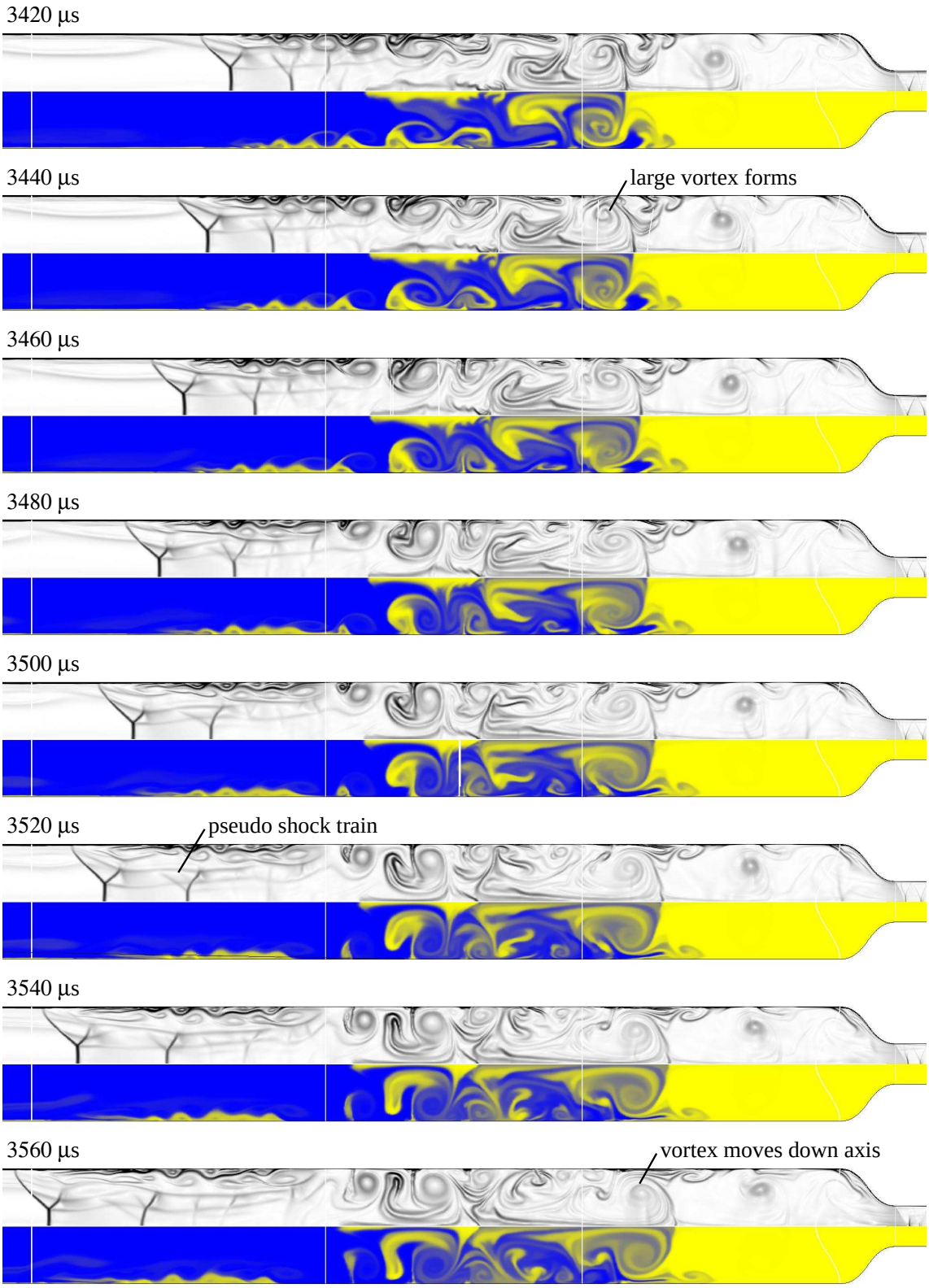

Figure 7.58: Part three of the sequence of numerical Schlieren images (top) and Driver gas mass fraction (bottom) showing the tailored interaction of the reflected shock with the contact surface. The Helium driver case simulation is used. The frames start at 3.42 ms and are at 20 μs intervals.

Effect of Mesh Refinement

Figure 7.59 shows the effect of mesh refinement on the interaction of the reflected shock with the contact surface. The coarse resolution mesh is shown on the top, the medium resolution mesh in the middle and the fine resolution mesh on the bottom. The three frames are taken at the same time after the start of the simulation and, therefore, the three frames appear further in time as the mesh is refined, due to the slightly higher shock speed along the shock tube. The same characteristics of the interaction are evident in the three frames, but the interaction appears to occur closer to the nozzle for the coarser meshes. A similar flow field is evident between the medium and fine meshes supporting the assertion that the simulation of the interaction on the fine mesh has reached mesh resolution independence. An important feature of these frames is that the strong vortex described in the previous section can be seen to form in all three cases; the vortex is in the early stages of formation in the coarse mesh frame.

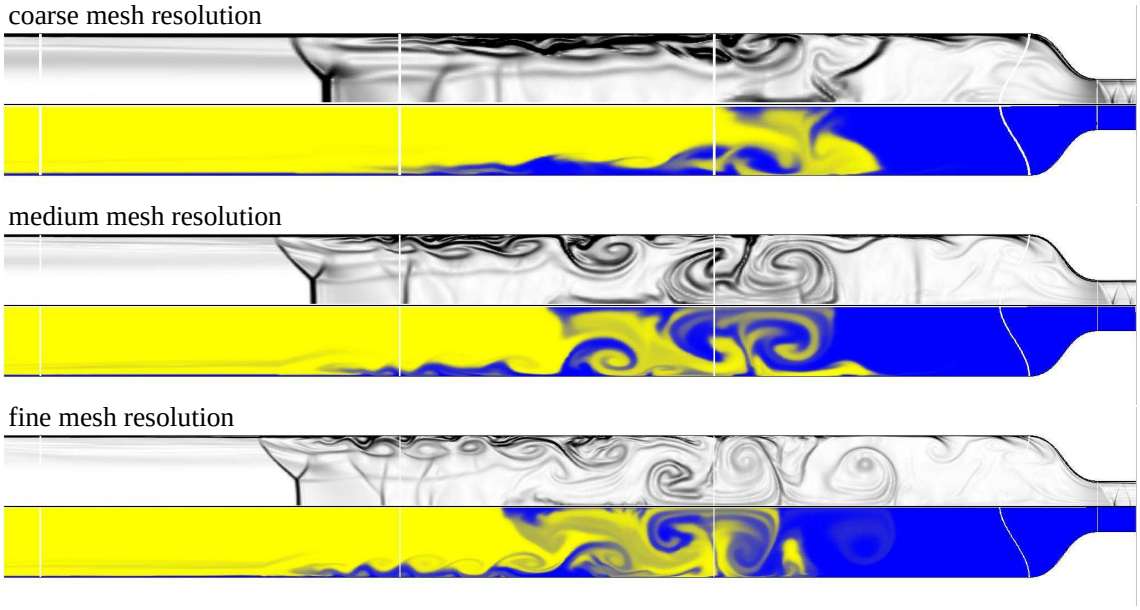

Figure 7.59: Effect of mesh refinement on the interaction of the reflected shock with the contact surface for the Helium driving Nitrogen case. The coarse mesh is shown on the top, the medium mesh in the middle and the fine mesh on the bottom.

Generation of Vorticity

The previous figures showed that the evolution of the flow following the interaction of the reflected shock with the contact surface is driven by the generation of vorticity through the interaction. This vorticity is generated by the mis-alignment of the density gradients with the shock pressure gradients (the baroclinic torque). This process was described in Chapter 6, and the baroclinic torque term, $\nabla\rho \times \nabla p/\rho^2$, was shown in Equation 6.1. The previous sequences of the interaction processes show numerical Schlieren in the upper halves of the frames. The numerical Schlieren is generated through the gradient of density, making it an indicator of where vorticity will be generated by interaction with the pressure gradient of the reflected shock.

Figure 7.60, again shows a sequence tailored contact surface interaction with the reflected shock. In this sequence, contours of the baroclinic torque are shown in the upper half of the frames (in black) and the contours of the accumulated vorticity are shown in the upper half of the frames (in red). The sequence of images of these derived variables allows the driving mechanisms of this interaction and of the Richtmyer-Meshkov instability to be illustrated.

As the contact surface is deformed by the vorticity that is generated, and as the reflected shock moves through this gas, additional density and pressure gradients are generated. These additional gradients interact with additional gradients and the process of vorticity generation is amplified. The result of this is that the vorticity continues to build in magnitude as gas continues to flow into the interaction region. As the interaction progresses, the regions of black (where vorticity is being generated) increase rapidly. In the frame at 3340 μs, the vorticity generation region extends from the shock structure back to head of the contact surface. The large regions of red at this time are also evident in the lower half of the frames, including two strong vortices, one at the head of the contact surface and one downstream of it. The vortex at the head of the contact surface eventually propagates downstream and into the test section, taking driver gas with it.

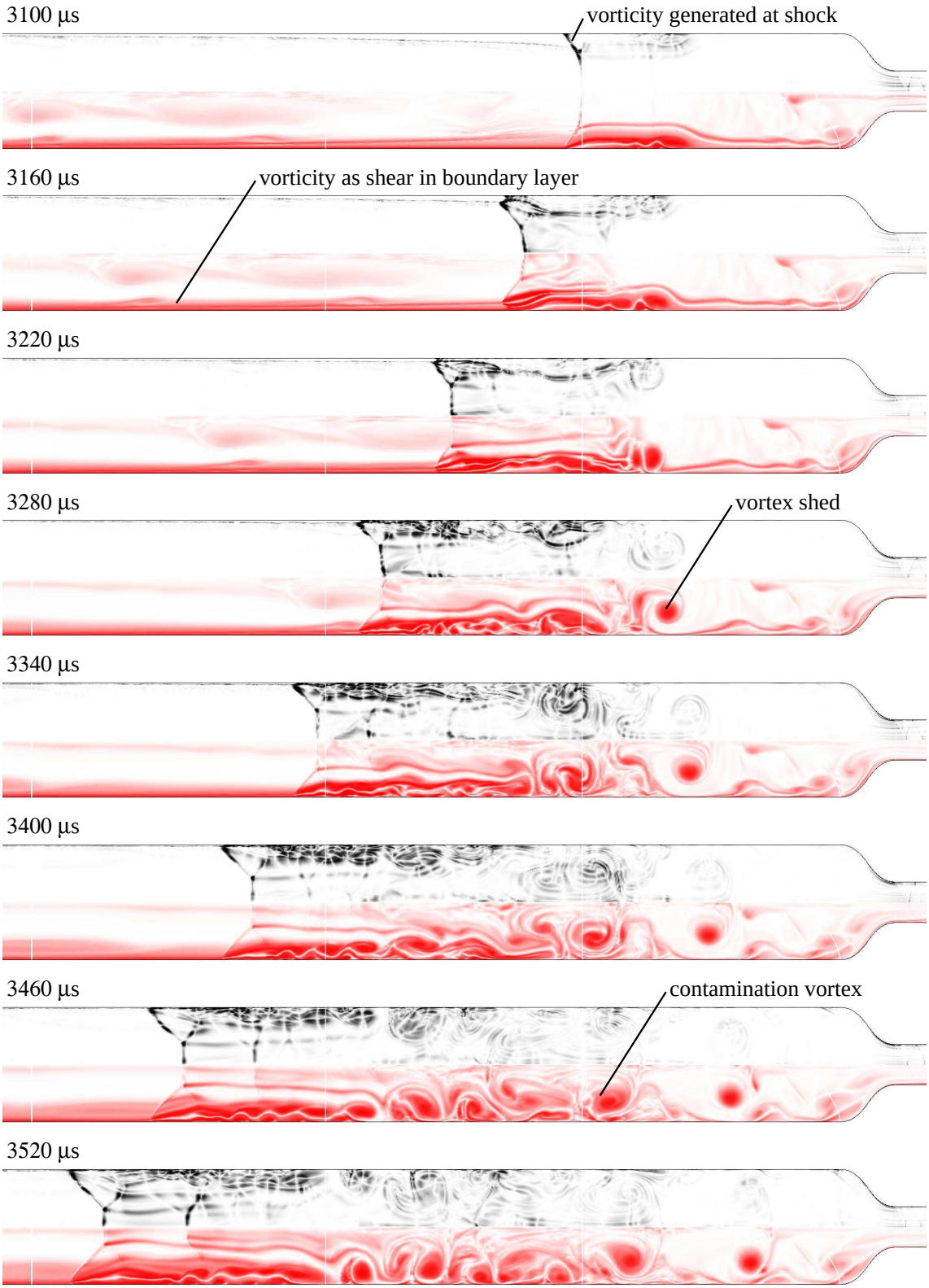

Figure 7.60: Sequence of the interaction of the reflected shock with the contact surface. The frames show the baroclinic generation of vorticity (in black on the top) and accumulated vorticity (in red on the bottom). The Helium driver case simulation is used. The frames start at 3.10 ms and are at 60 μs intervals.

7.3.8 Nozzle Test Flow

With the complete shock tunnel being simulated, the test flow can be reproduced. The nozzle test flow pitot pressure profiles were shown in Section 7.2. The pitot pressure traces indicate that the simulations accurately reproduce the test flow, including all of the transient features.

Turbulence in the boundary layers on the walls of the nozzle have a significant impact on the resulting test flows. As the flow is expanding through this region, the turbulent boundary layers increase in size dramatically. Without the turbulence model, the under-predicted boundary layer thickness means that, even with the pressure accurately predicted in the nozzle supply region, the nozzle test flow pitot pressure will be lower than that measured in the experiments.

Sequence of the Nozzle Start-up

Figure 7.61 shows the nozzle startup sequence. This sequence shows contours of the divergence of the velocity field. Numerical Schlieren contours (which were used in previous sequences) do not show the field throughout the expansion through the nozzle, because the density gradients in the nozzle supply region are much stronger than in the test flow.

The nozzle flow is initiated by the rupture of the secondary diaphragm, which is removed ideally in the simulations. This rupture is triggered by the sharp rise in pressure at the arrival of the shock, and gas expands into the test section following a series of startup waves. These startup waves consist of the primary shock, an upstream facing shock, a contact surface separating these two shocks, the upstream head of the unsteady expansion and the steady expansion back to the nozzle throat [111]. The startup processes occurring in shock tunnel nozzles were first described by Smith [214]. The processes are also described in more detail in Jacobs and Stalker [111] and Lee [129].

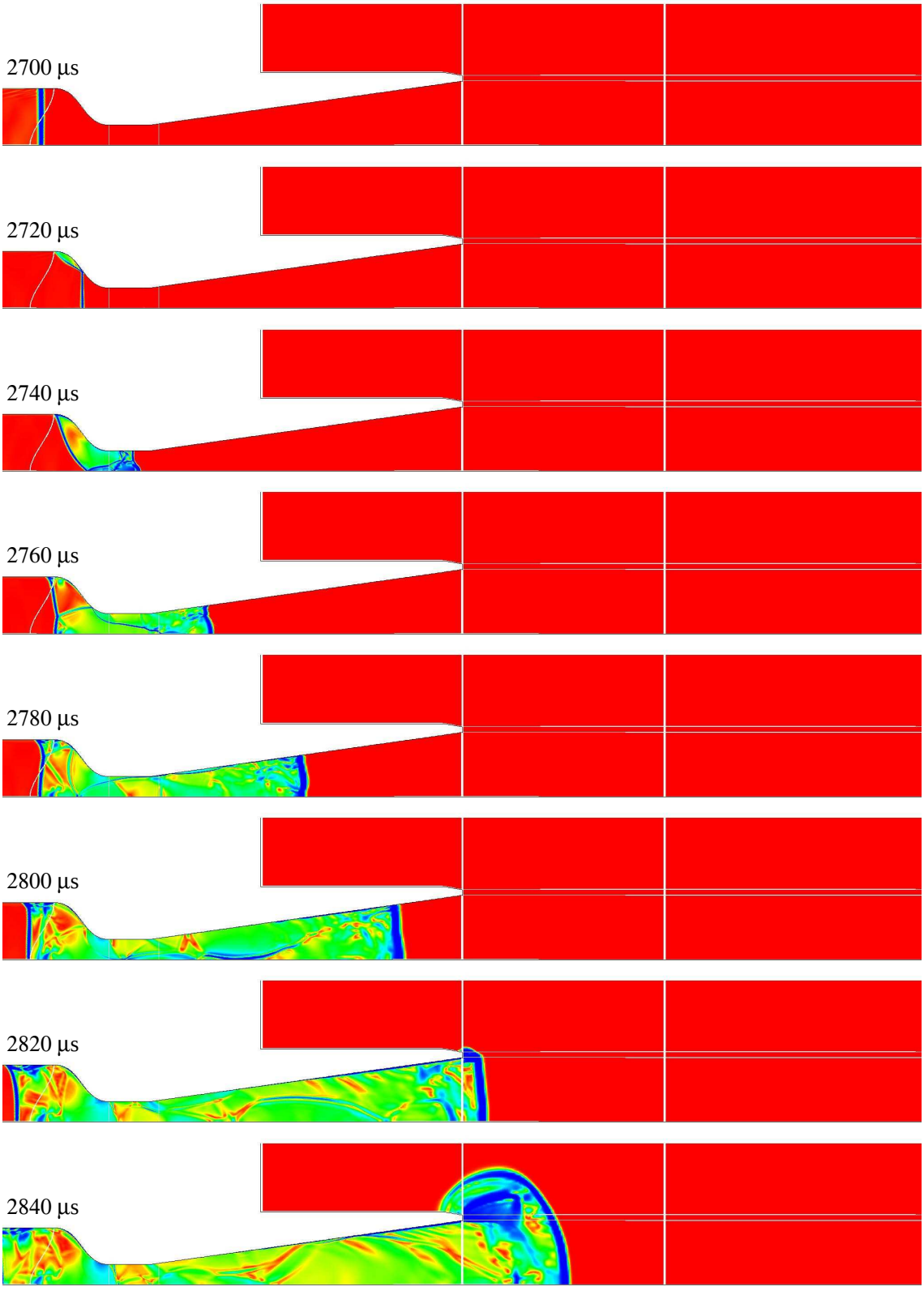

Figure 7.61: Part one of the sequence of nozzle startup. Contours of the divergence of the velocity are shown.

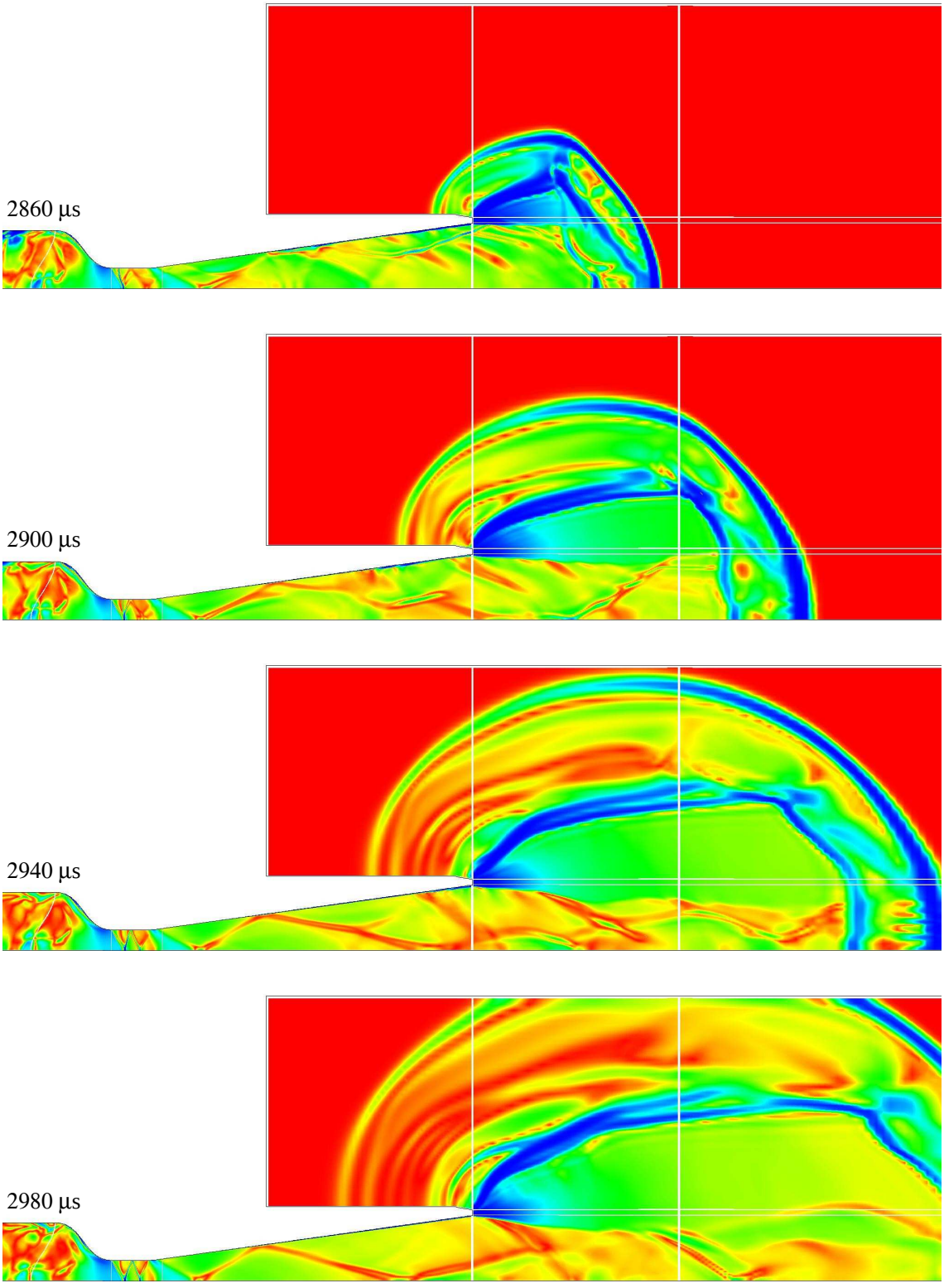

Figure 7.62: Part two of the sequence of nozzle startup. Contours of the divergence of the velocity are shown.

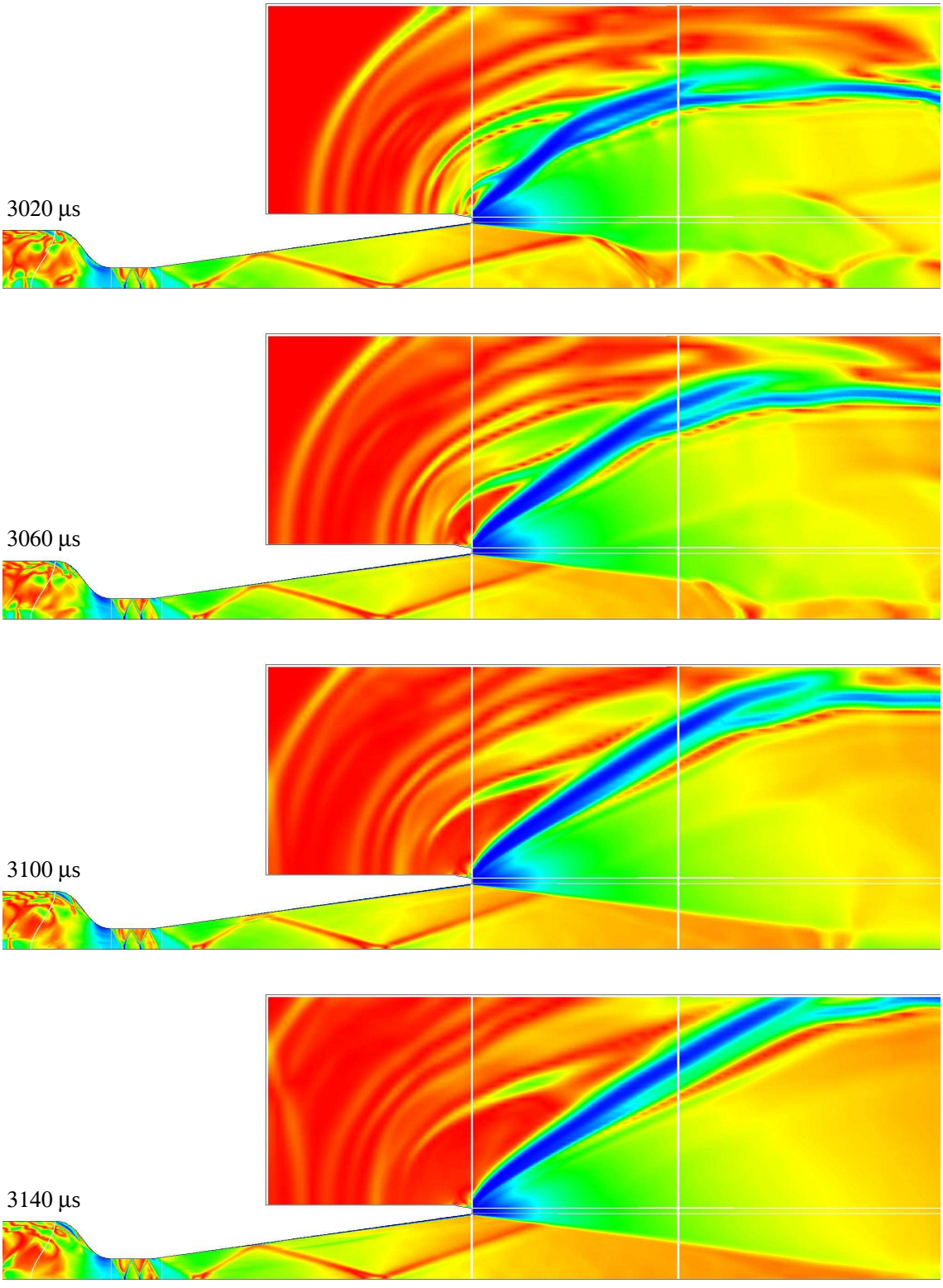

Figure 7.63: Part three of the sequence of nozzle startup. Contours of the divergence of the velocity are shown.

Test Flow Profiles

Figure 7.64 shows simulated profiles of test flow properties extracted 5 mm downstream from the nozzle exit plane during the test time. The profiles are from the Helium driving Nitrogen case simulation. The profile of Mach number shows the steady profile of Mach 4 flow produced across the core flow. Two peaks in Mach number are evident in the expansion at the nozzle edge, the inner jump being the conical shocks shown leaving the nozzle in Figure 7.63. The profile of axial velocity shows small fluctuations across the profile. In the profile of pressure the two steady shocks that pass through the nozzle exit plane are evident.

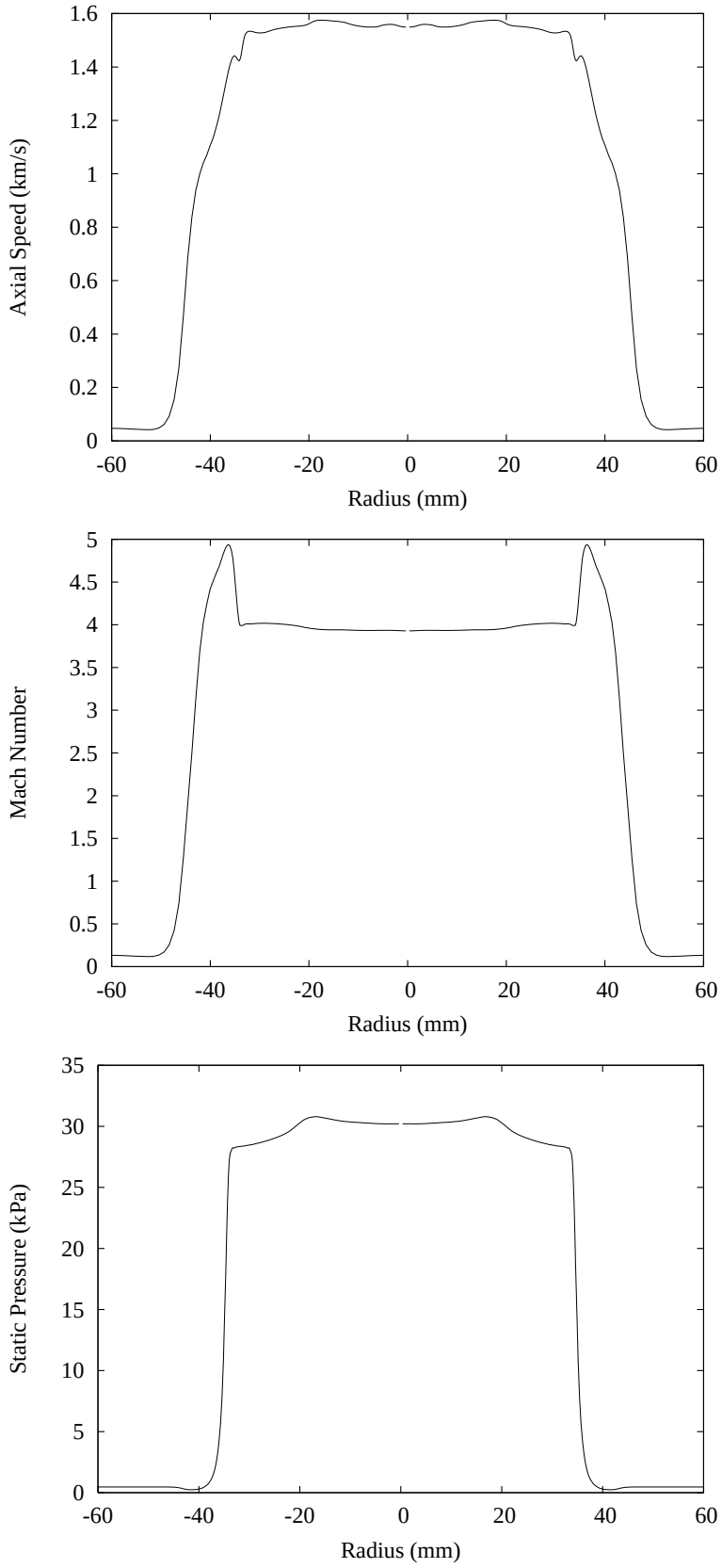

Figure 7.64: Profiles through the test flow 5 mm downstream of the nozzle exit plane. The profiles are of axial velocity (top), Mach number (middle) and static pressure (bottom). The profiles are taken from the flow development at 3.5 ms.

7.3.9 Driver Gas Contamination

Of the two experimental cases simulated, neither was ended by the arrival of driver gas in the test flow. Being a low-enthalpy facility, the Drummond Tunnel is less susceptible to driver gas contamination than larger, high-enthalpy facilities such as the T4 shock tunnel. The facility, and the simulations performed in this thesis, do provide a test-bed for investigating the mechanisms that lead to driver gas contamination. If the simulations can accurately predict the interaction of the reflected shock with the contact surface, then these simulations, provide a method of predicting driver gas contamination in reflected shock tunnels.

Figures 7.56, 7.57 and 7.58 showed a sequence of the interaction of the reflected shock with the contact surface. The mode of operation is roughly tailored; however, acceleration of driver gas towards the test flow is evident. Through the interaction process, a significant amount of vorticity is generated at the head of the contact surface. This resulted in the formation of a strong vortex, which is drawn towards the shock tube centreline and accelerates axially along the shock tube. Figures 7.65 and 7.66 continue this description by showing a sequence of driver gas mass fraction frames at later times. This sequence demonstrates the propagation of the vortex along the axis of the shock tube and into the test flow. This vortex, carrying driver gas material, represents a novel driver gas contamination mechanism that has not been noted in the literature. Even through the bifurcated foot of the reflected shock plays a vital role in the generation of the vortex, this contamination mechanism is distinct the jetting of driver gas through the shock foot described previously [228].

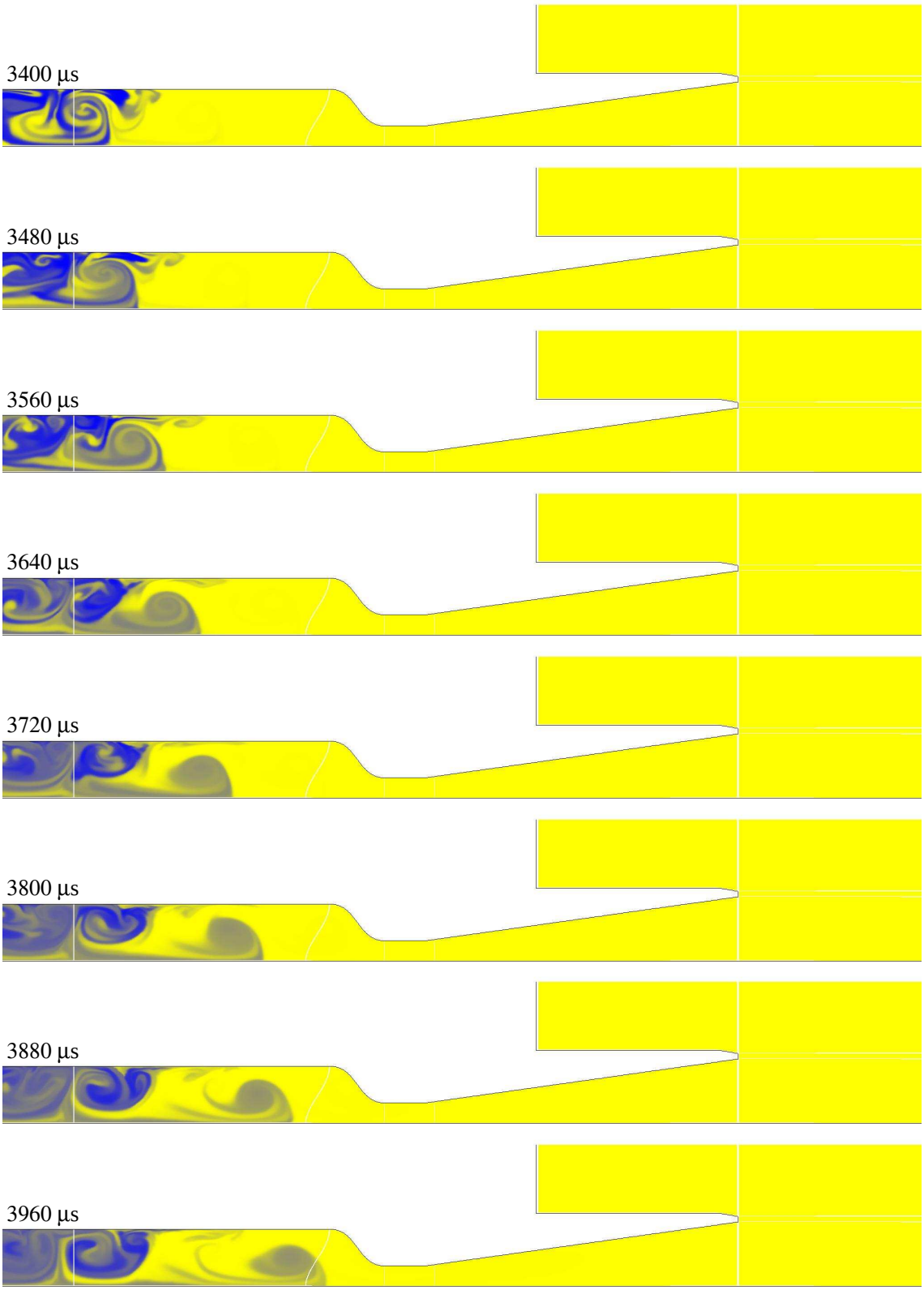

Figure 7.65: Part one of the sequence of driver gas contamination resulting from the vortex that was described in Section 7.3.7. Driver gas mass fraction contours are shown, with the driver gas in blue and the test gas in yellow). Helium driver case.

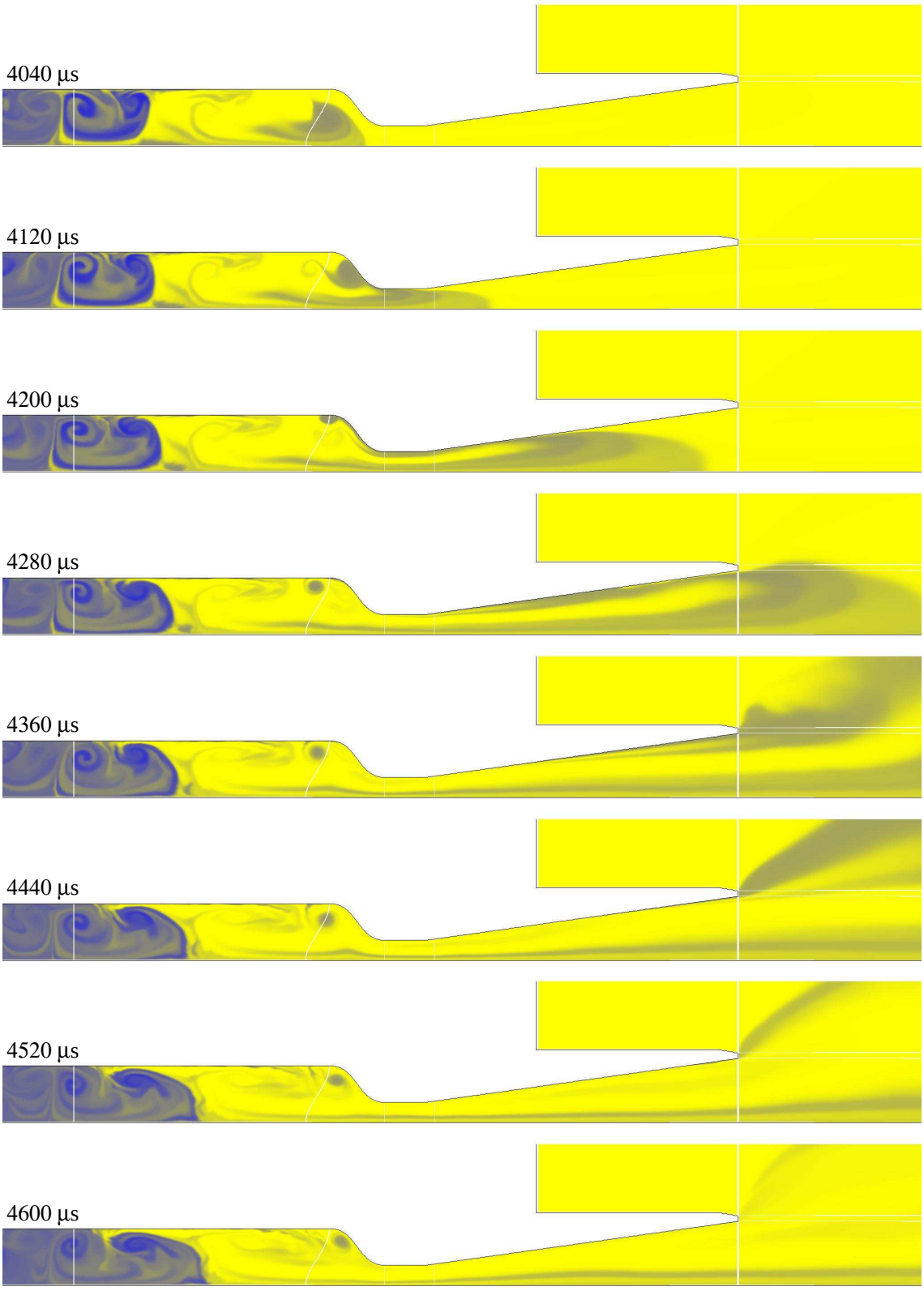

Figure 7.66: Part two of the sequence of driver gas contamination resulting from the vortex that was described in Section 7.3.7. Driver gas mass fraction contours are shown, with the driver gas in blue and the test gas in yellow). Helium driver case.

Figure 7.67 qualifies the arrival of driver gas in the test flow. The nozzle exit pitot pressure is shown on the left and the driver gas mass fraction over the equivalent time is shown on the right. The duration of the test flow is evident in the pitot pressure trace. Throughout the test flow duration, the test flow is uncontaminated (the test time is ended by the reflected expansion); however, some time following this the mass fraction of Helium in the test flow rises to between 20 and 30 %. This level remains relatively constant following the arrival of the vortex in the test flow.

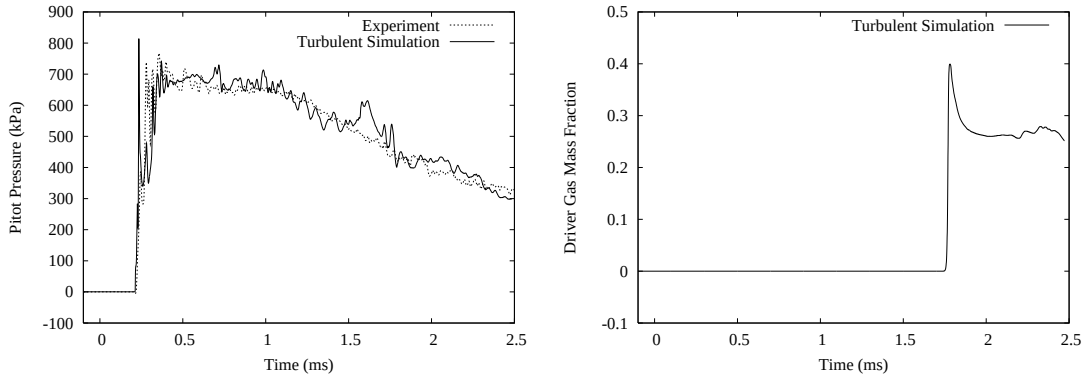

Figure 7.67: The arrival of driver gas in the simulated test flow for the Helium driving Nitrogen case. The nozzle exit pitot pressure is shown on the left and the driver gas mass fraction over the equivalent time is shown on the right.

The nozzle exit stagnation probe can be used as direct evidence of driver gas arriving in the test flow in the Nitrogen driving Nitrogen case. This plot, which is shown in Figure 7.16, shows a good comparison between the two, with the arrival of the driver gas predicted to within a 100 μs. The characteristics of the flow in this time are also predicted well. This case is over-tailored and so driver gas would be expected to arrive in the test flow; however, as noted by Sudani and Hornung [228] using experimental evidence obtained in the T5 shock tunnel, a form of driver gas contamination is also experienced in over-tailored cases, with driver gas being accelerated towards the test flow. This acceleration of some of the driver gas was evident in sequence of the over-tailored interaction shown in Figures 7.53, 7.54 and 7.55. The accuracy to which its arrival in the test flow is predicted by this simulation provides support to the assertion that the interaction of the reflected shock with the contact surface is being modelled accurately. No direct evidence of driver gas arriving in the test flow was measured experimentally in the Helium driver case.

The validity of the simulation of this interaction process and the resulting flow field is supported by the fact that the two important features in the interaction are modelled as accurately as possible. These are the characteristics of the contact surface, and the reflected shock following its interaction with the boundary layer.

There is some concern over the accuracy of the contact surface before the interaction. The contact surface in the Nitrogen driving Nitrogen case could not be reproduced accurately; however, the differences were influenced by turbulence in the real flow, which is not modelled. The contact surface was part of the way through a process of recirculation and mixing, which also influenced the results.

The contamination is driven by the vorticity generated at the interaction of the reflected shock with the contact surface. The bifurcated foot of the reflected shock, resulting from the interaction of the reflected shock with the boundary layer, does cause some jetting of gas as the reflected shock moves through the test gas; however, this jetting ceases as the reflected shock reaches the contact surface. This is because, as was suggested by Chue and Eitelberg [45], the vorticity generated in this interaction acts against the jetting of gas. The vorticity continues to form as the gas passes over the back of the material being carried upstream

The simulations discussed here have illustrated a new mechanism for driver gas contamination in shock tunnels, that compliments the near wall jetting associated with the bifurcation of the reflected shock.

7.3.10 Test Flow Noise Levels

The mechanisms that result in the generation of noise occur through the development of the flow. By modelling these processes, these simulations provide the potential to predict the noise levels in the test flow across a range of frequencies. Section 2.1.10 discussed the processes that were expected to contribute to the generation of noise, both in front of and behind the contact surface.

Significant levels of noise were generated in the driver gas, predominantly from the oblique waves generated during the finite opening time of the primary diaphragm. Paull and Stalker [175] showed that the penetration of noise from the driver gas into the test gas was limited by a sufficient increase in sound speed across the contact surface.

The effect of noise propagation across the contact surface is evident in these simulations. For the Nitrogen driving Nitrogen case, the change in sound speed across the contact surface is sufficient to prevent noise in the driver gas from entering the driven gas. In the traces ,from both the supply pressure transducer and the nozzle exit pitot pressure probe, the arrival of the driver gas is evident in a significant increase in fluctuations. This increase in noise is evident in both the experiment and the simulations. In the Helium driving Nitrogen case, change in sound speed across the contact surface does not prevent noise from crossing the contact surface. It is believed that noise from the driver gas enters the driven gas as the contact surface propagates along the shock tube. Fluctuations are evident in the test gas. These fluctuations remain at relatively the same level as the driver gas arrives at the two transducers.

The noise generated by the growth of the boundary layer can be visualised using the divergence of the velocity field in the region around the boundary layer. Contours of the divergence of the velocity field are shown in Figure 7.68. Several regular patterns of waves can be seen propagating in the region behind the shock, along with some regularly spaced spikes in flow which are numerical remnants from the passage of the shock between block boundaries. The boundary layers in the facility rapidly undergo transition to turbulence. Noise is generated from the turbulent fluctuations; however, these fluctuations are not modelled in these simulations.

The shock reflection and interaction processes have a significant effect on noise generation. This noise can be seen in the numerical Schlieren frames in the sequences of these processes throughout Section 7.3. As the reflected shock processes are occurring, fluctuations in density can be seen in the numical Schlieren frames. Some of these fluctuations can be seen to propagate into the test flow. As significant proportion of these fluctuations propagate along the shock tube walls.

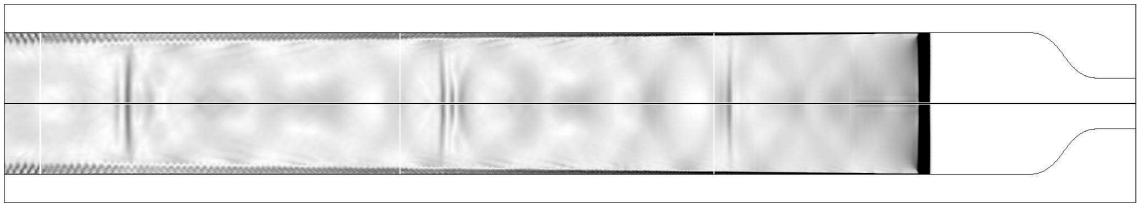

Figure 7.68: Divergence of the velocity field in the shock tube around the growing boundary layer. The transverse waves generated by the boundary layer growth are evident in the flow.

Simulations of a complete facility provide a method of estimating the effect of noise and simulating its effect on experimental results. The level of noise in shock tunnel test flows is difficult to estimate by analytical means. Adam and Hornung [1] showed that there is no clear relationship between transition Reynolds number, and therefore noise levels, and reservoir enthalpy. Difficult to impose noise levels in a simulation without generating noise throughout the flow.

Some potentially relevant noise generating processes have not been simulated. The iris based model of diaphragm rupture does not take into account the deformation and fragmentation of the diaphragm material. The turbulent eddies on the wall of the nozzle are known to have a significant effect on the noise levels measured in the test flow. These turbulent motions are not modelled in these simulation; only the increase in boundary layer thickness resulting from the turbulence is taken into account. The effect of free stream turbulence is not simulated. In addition, the propagation of high frequencies of noise is limited by the mesh.

Measurement of Fluctuations in the Test Flow

Figure 7.69 shows the simulated and experimental pitot pressure at the nozzle exit for the Helium driving Nitrogen case. The experimental trace is shown on the left and the simulated trace is shown on the right. The fluctuations in the Helium driver case are important because the pitot probe used in these experiments utilized a high bandwidth configuration (with a resonance of around 230 kHz), whereas in the N_2 driver experiments the pitot probe was protected from the flow with a pneumatic cavity (with a resonance of around 10 kHz). Restricting attention to frequencies between 10 and 50 kHz, the measured RMS fluctuation in the He driver case is 1.4 % over the period from about 0.5 to 1 ms on the time scale in the figure.

The simulation yields RMS pitot pressure fluctuations of 1.2 % over the same period and frequency band. This result suggests that, in this case, the dominant source of the measured pitot pressure fluctuations is within the shock tube rather than noise radiating from the nozzle boundary layers since the contribution of tur-

bulence in the nozzle boundary layers is not modelled in these simulations. It is known that in this case, from the relative levels of noise in the driver and the driven gas, and from the acoustic characteristics of the contact surface, that noise from the driver gas enters the driven gas. This is likely to make a significant contribution to these measured noise levels.

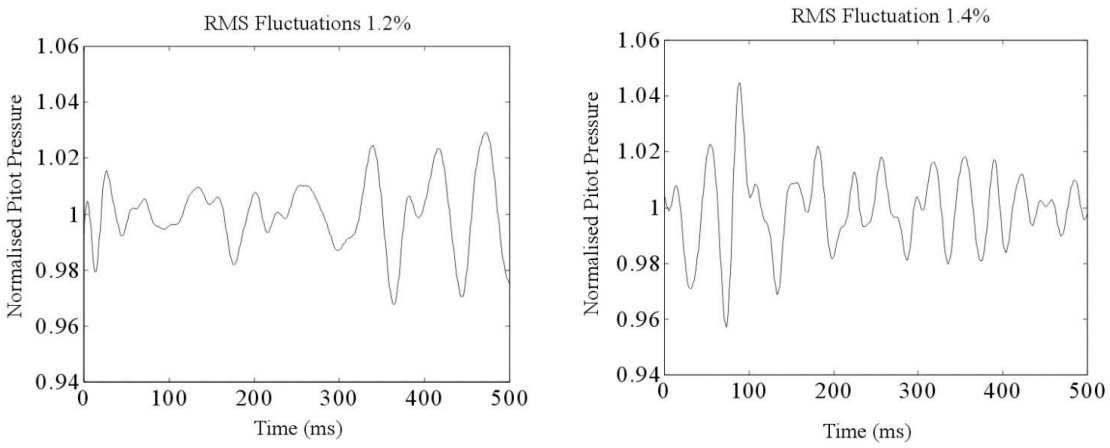

Figure 7.69: Fluctuations in pitot pressure measured in the test flow. The experimentally measured fluctuations are shown on the left and the fluctuations measured in the simulated test flow are shown on the right.

CHAPTER 8

Conclusions

The primary motivation for this thesis was the development of numerical simulations of a reflected shock tunnel for the purpose of providing a better understanding of shock tunnel flows. A multi-block Navier-Stokes code, MB_CNS, was used in the simulations. Measurements recorded during a set of physical experiments were used for calibration and validation of the simulations. The following sections discuss the outcomes of this thesis.

Modelling and Validation

The Drummond Tunnel is a relatively low-enthalpy shock tunnel operated at The University of Queensland. In this thesis, simulations of the Drummond Tunnel are developed and these simulations are used to investigate the flow through the facility. The simulations assumed axisymmetric flow and used computational meshes covering the complete shock tunnel, from the driver section to the dump tank. By modelling the complete facility, some of the assumptions required by other simulation approaches have been removed, allowing a more complete description of the flow development. The use of an axisymmetric mesh precluded fully three dimensional motion, but still allowed the simulation of many of the complex, non-ideal processes that occur during the operation of a reflected shock tunnel. The simulations included a model of the effect that the rupturing primary diaphragm has on the flow and assumed an ideally rupturing secondary diaphragm. An over-tailored and a roughly tailored mode of operation was investigated.

The simulations of shock bubble interaction discussed in Chapter 6 demonstrate the capability of the numerical techniques used in MB_CNS to accurately model the shock induced instability and deformation of a contact surface separating different gases. This provides support for the validity of the simulations used in the shock tunnel modelling in Chapter 7. The simulations of the Drummond Tunnel show that instabilities in the contact surface, separating the driver gas from the driven gas, play an important role in shock tunnel operation. These instabilities are dependent

on the characteristics of the flow as it evolves throughout the facility.

The validation of the simulations with experimental measurements is an important aspect of this study. Given that the simulations are provided with only the initial operating conditions used in the shock tunnel, the reproduction of the experimental measurements by the simulations provides strong support for the overall validity of the simulations. Experimental measurements are made inside the boundary layer, at an axial position close to the location of the interaction of the reflected shock with the contact surface. The comparisons with the experimental results are, therefore, sensitive to the accuracy of the simulations.

The Baldwin-Lomax eddy viscosity model [11] was shown to be effective in the simulation of shock tunnel flows. Significant improvements in the reproduction of the experimental traces, resulting from the implementation of the model, were demonstrated in Section 7.2. The Baldwin-Lomax model is an incomplete turbulence model and requires that certain parameters be calibrated for the flow conditions being simulated. This required prior knowledge of the flow and a significant amount of work in calibrating the coefficients. The values of two coefficients that are commonly modified, C_{cp} and C_{kleb}, were obtained from their dependence on Mach number, as described in Kim, Harloff and Sverdup [121]. The original coefficients used in the model were found to be incompatible with the shock tunnel simulations, and an additional modification to the model, through the Karman constant, was required in order to reproduce the experimental results in the simulations.

Along with the Baldwin-Lomax model, not knowing the initial temperature of the driver gas meant that the simulations were not fully predictive and required prior knowledge of the flow in the form of the experimental traces. Future work in this area would benefit from the implementation of a more complete turbulence model, such as the one equation Spalart-Allmaras model [217].

Smoothed Particle Hydrodynamics

The Smoothed Particle Hydrodynamics (SPH) technique [137, 78] was investigated for advantages that it may provide in the simulation of shock tunnel flows. Being Lagrangian in nature, the technique provides a more natural treatment of convecting fluid interfaces.

A CFD code based on the SPH technique was developed and applied to a number of test cases. Through the development of the SPH code and its implementation in the test cases, significant limitations were discovered that make the technique unsuitable for the modelling of shock tunnel flows. These limitations included its treatment of solid boundaries, difficulties in accurately specifying initial conditions

and the problem of particle penetration through the fluid interfaces. In addition, the technique is limited by its relatively low resolution, which is a result of its requirement for the use of an artificial viscosity. The limitations identified in the technique, and its implementation, meant that the development of the SPH code for shock tunnel simulation was discontinued.

Parallel Computing

In the past, simulations of reflected shock tunnels have been limited, by the available computational power, to simulating only the end of the shock tube [206, 240, 45]. The use of parallel computation in this thesis has allowed the extension of the simulation of shock tunnels to a complete facility on an axisymmetric mesh. MB_CNS utilises a multi-block solution procedure in order to exploit parallelism. The flow field can be solved separately in each of these blocks, requiring information to be shared only at the block boundaries at the end of each time step.

Two versions of MB_CNS have been developed, one using shared memory and the other distributed memory. The shared memory version of MB_CNS, using OpenMP, was developed from an earlier Power C version. The simulations described in this thesis, both in Chapter 6 and 7, used the OpenMP version of MB_CNS with four threads. The OpenMP version is simple, since all flow data can be read from the shared memory space, and is efficient; however, it requires special shared memory hardware to run and is limited to four processors on the APAC National Facility.

The Message Passing Interface (MPI) parallel version of MB_CNS has also been developed. This version solves each of the flow-field blocks in a separate memory space and so the block boundaries must be transferred between the blocks explicitly in the code. This version of MB_CNS was not used significantly in the simulations of described in this thesis; however, it is important for the extension of the simulations in this thesis to large scale, high-enthalpy facilities, where the modelling of finite-rate chemistry will add significantly to the computational work. This version allows MB_CNS to be run in parallel on Beowulf workstation clusters [223].

The OpenMP version was found to be efficient and predictable in performance across the range of simulation sizes considered. The MPI version was shown to provide unacceptable levels of efficiency in the small diagnostic simulations; however, this version is aimed at the type of large scale simulations to which this form of parallelism is more suited. The SPH technique is naturally parallel and its performance in parallel is investigated in detail in Appendix A.

Driver Gas Contamination

Through the investigation of the simulated flow fields, a previously unobserved mechanism for the premature contamination of the test flow with driver gas was discovered. This contamination mechanism is driven by the generation of vorticity in the contact surface through its interaction with the reflected shock. Vorticity is generated as the density gradient at the contact surface passes through the oblique shocks that form the bifurcated foot of the reflected shock. This means that the interaction of the reflected shock with the boundary layer, and the resulting shock bifurcation, is an important aspect of the observed contamination mechanism. The vorticity is generated through the baroclinic torque that exists where gradients of density are mis-aligned with gradients of pressure. The vorticity subsequently rolls up to form a strong vortex at the head of the contact surface. Drawn in towards the centerline, the vortex accelerates downstream along the axis of the shock tube. Driver gas carried along by the vortex, reaches the nozzle and contaminates the test flow. The vorticity driven contamination of the test flow was observed in the simulations for both the tailored and over-tailored operating conditions. In the over-tailored case, the premature arrival of driver gas in the test flow was verified with the experiment measurements.

Previous numerical studies of driver gas contamination have described the jetting of driver gas through the bifurcated foot of the reflected shock, and along the shock tube wall, as the driving mechanism from the contamination [206, 240]. The simulations performed in this thesis have shown that, although some relative movement of gas through the reflected shock foot is evident, it is not sufficient to be responsible for jetting driver gas into the test flow. In addition to this, it has been shown that the vorticity generated at the contact surface acts to prevent this gas jetting as soon as the reflected shock reaches the driver gas; this was also observed by Chue and Eitelberg [45]. The new observations made in this thesis are enabled by the simulation of the complete facility, which includes an evolving contact surface that is significantly distorted by the time it encounters the reflected shock. The resulting contamination mechanism transports the driver gas along the shock tube centreline. This may explain why methods previously aimed at preventing driver gas moving along the shock tube walls from reaching the test flow have been sometimes unsuccessful [228].

Test Flow Noise Levels

The present simulations have the potential to investigate the mechanisms that cause the noise experienced in the test flows of shock tunnel facilities. These noise levels are

often an order of magnitude larger than those experienced in the flight environment [202]. This means that investigation, both of the mechanisms that generate the noise, and the effect that the noise has on experimentation are important areas of shock tunnel research.

Noise is generated through various processes occurring in operation of the shock tunnel, including the rupture of the primary diaphragm, the growth of boundary layers, shock reflection, and the interaction of the reflected shock with the boundary layers and the contact surface. Many of the processes leading to the generation of noise are modelled in the simulations described in this thesis, and the numerical Schlieren images in the sequences shown in Chapter 7 demonstrate the generation of fluctuations in the shock tube and propagation of these fluctuations into the test flow.

The noise levels were measured in the test flow pitot pressure during the experiments conducted by Dr. D. R. Buttsworth in the Drummond Tunnel. The magnitude of fluctuations, at frequencies between 10 and 50 kHz, in the nozzle exit flow was accurately reproduced by the simulations. This demonstrates the potential of these simulations in reproducing the mechanisms that lead to these high noise levels.

Limitations of the Modelling

The simulations are thought to provide a realistic representation of the flow through the complete shock tunnel. Comparisons between the simulated flow and the experimental measurements indicate that the two main limitations in the simulations are associated with shock boundary layer interaction and the representation of the contact surface.

Shock Boundary Layer Interaction Turbulent motions are expected to play an important role in the interaction of the reflected shock with the turbulent boundary layer. The pressure profiles measured through this interaction in the experiments were different from those measured in the simulations. The boundary layers modelled in the simulations are steady approximations of the turbulent boundary layer and, in the simulated interaction, material is allowed to build up steadily at the foot of the shock. In the real interaction, the turbulent fluctuations in the boundary layer would result in an interaction with material crossing the reflected shock unsteadily. The material at the foot of the shock is expected to be highly turbulent.

Representation of the Contact Surface The representation of the contact surface is the most serious deficiency of the simulated shock tunnel. Despite

modelling the effect of the primary diaphragm, the flow of material through the boundary layers and the resulting stability properties of the contact surface, the level of mixing and diffusion at the real contact surface was not reproduced in the simulations. This was thought to be caused by diffusion and turbulent mixing not being included in the simulations. Deficiencies in the representation of the contact surface were evident in the sharply defined tailoring waves and the incorrect contact surface characteristics recorded at the locations of the shock tube heat flux rake.

In addition, turbulence in the free stream flow was not accounted for. The Reynolds-averaging of the flow and the axisymmetry imposed by the mesh prevent turbulent fluctuations from being modelled in detail. The late time flow field would be dominated by turbulence but, in the present axisymmetric simulations, the late time vortex field is constrained to be a series of coherent vortex rings.

Future Developments

The simulations performed in this thesis have demonstrated the feasibility using simulations of a complete shock tunnel to gain a better understanding of the flows through these facilities. They have been able to reproduce experimental measurements with reasonable accuracy. Future studies of the numerical simulation of shock tunnels can improve on these simulations in some specific areas.

More complete experimental data would be helpful. For example, in the simulations presented in this thesis the driver temperature was not known. Measurements that would otherwise be redundant would also be useful, as any measurements provide important constraints to the problem and can be used as validation.

As was discussed earlier, the implementation of the Spalart-Allmaras turbulence model would be of benefit in future simulations. The Baldwin-Lomax model is an incomplete turbulence model and, therefore, requires the calibration of coefficients using knowledge of the flow.

The most important improvement required in these simulations concerns the representation of the contact surface. Simulations using the Equilibrium Interface Method (EIM) [140], have the potential to provide a better representation of the level of diffusion at the contact surface. Alternatively, a diffusion model in the system of equations solved by MB_CNS could be implemented.

The efficiency of the parallel versions of MB_CNS is critical to the extension of these simulations to large scale facilities. The efficiency of the MPI version of

MB_CNS could be improved by performing the inter-block communication concurrently with some of the computation. The solution of the flow in the interior of blocks is not dependent on the block boundaries and could be performed while boundary data is still being updated from the surrounding blocks.

It has been assumed that the differences observed between the experimental and simulated measurements of the interaction of the reflected shock with the boundary layer are caused by the effect of turbulent fluctuations, which are not modelled in detail in the simulations. A detailed study of this interaction process, using a numerical technique that included these turbulent fluctuations, such as Large Eddy Simulation (LES), could provide a better understanding of this process.

Preventing Driver Gas Contamination

Attempts at preventing driver gas contamination in the past have focused on the jetting of the gas through the foot of the reflected shock and along the wall. Methods of preventing driver gas contamination based on this assumption have been largely unsuccessful [228]. The simulations described in this thesis provide a method of being able to predict the premature arrival of driver gas in the test flow through the new contamination mechanism observed. They therefore provide the means of being able to conduct trials of methods aimed at driver gas contamination.

A new style of annular diaphragm has been designed as is being tested in the T4 shock tunnel at the University of Queensland. This diaphragm opens through a series of holes drilled through a thick steel plate, resulting in a more even opening profile. This diaphragm has the potential to postpone driver gas contamination by preventing the jetting of driver gas along the walls of the shock tube, as is observed in the simulations in this thesis, or the jetting of driver gas along the centreline of the shock tube as was predicted by Skinner [212].

The use of 'cookie cutters' is common in experiments to remove unwanted flow from the test section. This type of device could be used to remove the flow around the walls with the boundary layers, meaning that only the core flow would pass through to reflect from the nozzle. The shock would reflect into a much smaller boundary layer, built up over the length of the cookie cutter only, thus reducing the reflected shock interaction processes that lead to the generation of noise and the contamination of the test flow with driver gas. The test gas material in the boundary layers is unsuitable for use in the test flow anyway.

Bibliography

[1] Adam, P. H. and Hornung, H. G., "Enthalpy effects on hypervelocity boundary-layer transition: Ground test and flight data," *Journal of spacecraft and rockets*, Vol. 34, 1997, pp. 614–619.

[2] Ahmad, I., "Cluster computing: A glance at recent events," *IEEE Concurrency*, Vol. 8, No. 1, 2000, pp. 67–69.

[3] Ali, T. S. A., "MPI: A message passing interface," *MPI Forum*, 1993, pp. 1–5.

[4] Amann, H. O., "Experimental study of the starting process in a reflection nozzle," *The Physics of Fluids, Supp. 1* , Vol. 12, 1969, pp. I150–I153.

[5] Amza, C., Cox, A., Dwarkadas, S., Keleher, P., Lu, H., Rajamony, R., Yu, W., and Zwaenepoel, W., "Treadmarks: Shared memory computing on networks of workstations," *IEEE Computer*, Vol. 29, No. 2, 1996, pp. 18–28.

[6] Arnold Engineering Development Center, Tennessee, *AEDC Aerodynamics*, http://www.arnold.af.mil/aedc/aerodynamics/, Accessed February 2003.

[7] Austin, J. M., Jacobs, P. A., Kong, M. C., Barker, P., Littleton, B. N., and Gammie, R., "The small shock tunnel facility at UQ," Department of Mechanical Engineering Report 2/1997, The University of Queensland, Brisbane, July 1997.

[8] Badcock, K. J., "A numerical simulation of boundary layer effects in a shock tube," *International Journal for Numerical Methods in Fluids*, Vol. 14, 1992, pp. 1151–1171.

[9] Bagabir, A. and Drikakis, D., "Mach number effects on shock-bubble interaction," *Shock Waves*, Vol. 11, 2001, pp. 209–218.

[10] Bal, H. E., Steiner, J. G., and Tanenbaum, A. S., "Programming languages for distributed computing systems," *ACM Computing Surveys*, Vol. 21, No. 3, 1989, pp. 261–322.

[11] Baldwin, B. S. and Lomax, H., "Thin layer approximation and algebraic model for separated turbulent flows," *AIAA 16th Aerospace Sciences Meeting*, Huntsville, Alabama, 16-18 January, 1978, pp. Paper 78–257.

[12] Bazhenova, T., Eremin, A., Kochnev, V., and Naboko, I., "Test time behind reflected wave in a shock tube with nozzle," *Recent Developments in Shock Tube Research, Proceedings of the Ninth International Shock Tube Symposium*, edited by D. Bershader and W. Griffith, Stanford University Press, Stanford University, California, 1973, pp. 474–485.

[13] Beazley, D. M. and Lomdahl, P. S., "Controlling the data glut in large-scale molecular dynamics," *Computers in Physics*, Vol. 11, No. 3, 1997, pp. 230–238.

[14] Beckman, C. J. and McManus, D. D., "Horizons in scientific and distributed computing," *Computing in Science and Engineering*, Vol. January-February, 1999, pp. 23–30.

[15] Beckwith, I. E. and Miller, C. G., "Aerothermodynamics and transition in high-speed wind tunnels at NASA langley," *Annu. Rev. Fluid Mech.*, Vol. 22, 1990, pp. 419–439.

[16] Belytschko, T., Krongauz, Y., Dolbow, J., and Gerlach, C., "On the completeness of meshfree particle methods," *International Journal for Numerical Methods in Engineering*, Vol. 43, 1998, pp. 785–819.

[17] Belytschko, T., Krongauz, Y., Organ, D., Fleming, M., and Krysl, P., "Meshless methods: An overview and recent developments," *Computational Methods Applied Mechanics and Engineering*, Vol. 139, 1996, pp. 3–47.

[18] Benz, W., "Smooth particle hydrodynamics: Theory and application to the origin of the moon," *Use of Supercomputers in Stellar Dynamics*, Springer Verlag, 1986, pp. 117–124.

[19] Benz, W., "Smooth particle hydrodynamics: A review," *The Numerical Modelling of Nonlinear Stellar Pulsations*, 1990, pp. 269–288.

[20] Besnard, D. and Haas, J. L., "Two-dimensional simulation of contact surface instabilities in shock tubes," 1989, pp. 296–301.

[21] Blume, W. and Eigenmann, R., "Performance analysis of parallelizing compilers on the perfect benchmark programs," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 3, No. 6, 1992, pp. 643–656.

[22] Boris, J. P. and Book, D. L., "Flux-corrected transport," *Journal of Computational Physics*, Vol. 11, 1973, pp. 38–69.

[23] Bratley, P. and Fox, B. L., "Algorithm 659: Implementing sobol's quasirandom sequence generator," *ACM Transactions on Mathematical Software*, Vol. 14, No. 1, 1988, pp. 88–100.

[24] Bravo, E. and Garcia-Senz, D., "Smooth particle hydrodynamics simulations of deflagrations in supernovae," *The Astrophysical Journal*, Vol. 450, 1995, pp. L17–L21.

[25] Brouillette, M., "The richtmyer-meshkov instability," *Annual Review of Fluid Mechanics*, Vol. 34, 2002, pp. 445–468.

[26] Brouillette, M. and Sturtevant, B., "Richtmyer-meshkov instability at a continuous interface," *17th International Symposium on Shock Waves and Shock Tubes*, edited by Y. W. Kim, American Institute of Physics, Lehigh University, Bethlehem, Pa, 1989, pp. 284–289.

[27] Bull, D. C. and Edwards, D. H., "An investigation of the reflected shock interaction process in a shock tube," *AIAA Journal*, Vol. 6, No. 8, 1968, pp. 1549–1555.

[28] Burrage, K., *Parallel and Sequential Methods for Ordinary Differential Equations*, Oxford University Press, Oxford, 1995.

[29] Burtschell, Y., Brun, R., and Zeitoun, D., "Two dimensional numerical simulation of the marseille university free piston shock tunnel-tcm2," *The 18th International Symposium on Shock Waves*, Sendai, Japan, July 20-25, 1991.

[30] Buttsworth, D. R., Goozée, R. J., and Jacobs, P. A., "Numerical simulation of fluctuations in a shock tunnel flow." *International Conerfence on Computational Fluid Dynamics 2*, Springer-Verlag, July 2002.

[31] Buttsworth, D. R. and Jacobs, P. A., "Total temperature measurements in a shock tunnel facility," *13th Australasian Fluid Mechanics Conference*, edited by M. C. Thompson and K. Hourigan, Melbourne, December 1998, pp. 51–54.

[32] Buttsworth, D. R. and Jacobs, P. A., "Measurement of fluctuations in a Mach 4 shock tunnel nozzle flow," *Heat and Mass Transfer Australasia 2000: Proceedings of the Seventh Australasian Heat and Mass Transfer Conference*, edited by G. B. Brassington and J. C. Patterson, Chalkface Press, Western Australia, July 2000, pp. 53–59.

[33] Buyya, R., *High Performance Cluster Computing; Volume 1 and 2*, Prentice Hall, New Jersey, 1999.

[34] Calspan-UB Research Center, *University of Buffalo Research Center*, http://www.cubrc.org/, Accessed February 2003.

[35] Cambier, J. L., Tokarcik, S., and Prabhu, D. K., "Numerical simulations of unsteady flow in a hypersonic shock tunnel facility," *AIAA 17th Aerospace ground testing conference*, Nashville, TN, July 1992, pp. 127–140.

[36] Campbell, G. A., Kimber, G. M., and Napier, D. H., "Bursting of diaphragms as related to the operation of shock tubes," *Journal of Scientific Instruments*, Vol. 42, 1965, pp. 381–381.

[37] Cardoso, M., Burtschell, Y., Zeitoun, D., and Abgrall, R., "Numerical simulation of different methods for avoiding driver gas contamination in shock tunnels," *Proceedings of the 21st International Symposium on Shock Waves, Volume 1*, Panther Publishing, Great Keppel, Australia, 1997, pp. 537–542.

[38] Carrieo, N. and Gelernter, D., "How to write parallel programs: A guide to the perplexed," *ACM Computing Surveys*, Vol. 21, No. 3, 1989, pp. 323–357.

[39] CEA (Chemical Equilibrium with Applications) Program, *The NASA Glenn Chemical Equilibrium Program*, http://www.grc.nasa.gov/WWW/CEAWeb/, Accessed February 2003.

[40] Cebeci, T. and Smith, A. M. O., *Analysis of Turbulent Boundary Layers*, Academic Press, New York, NY, 1974.

[41] Centre for Hypersonics, The University of Queensland, *T4 Shock Tunnel*, http://www.mech.uq.edu.au/hyper/level2/hyper_T4.html, Accessed February 2003.

[42] Chen, J. K., Beraun, J. E., and Carney, T. C., "A corrective smoothed particle method for boundary value problems in heat conduction," *International Journal for Numerical Methods in Engineering*, Vol. 46, 1999, pp. 231–252.

[43] Chue, R. S. M., "Numerical analysis of reflected shock/boundary layer interaction in a high enthalpy shock tunnel," *Proceedings of the 20th International Symposium on Shock Waves*, World Scientific Publishing, California Institute of Technology, Pasadena, California, 1995.

[44] Chue, R. S. M. and Dumitrescu, M. P., "Boundary layer suction device preliminary investigation," *Proceedings of the 21st International Symposium on Shock Waves*, Panther Publishing, Fyshwick, Australia, 1997, pp. 549–554.

[45] Chue, R. S. M. and Eitelberg, G., "Studies of the transient flows in high enthalpy shock tunnels," *Experiments in Fluids*, Vol. 25, 1998, pp. 474–486.

[46] Chue, R. S. M. and Itoh, K., "Influence of reflected-shock/boundary-layer interaction on driver gas contamination in high-enthalpy shock tunnels," *Proceedings of the 20th International Symposium on Shock Waves, Volume 1*, World Scientific, Singapore, 1996, pp. 777–782.

[47] Cleary, P. and Ha, J., "SPH modelling of isothermal high pressure die casting," *13th Australian Fluid Mechanics Conference*, 1998, pp. 663–666.

[48] Cleary, P. W. and Monaghan, J. J., "Boundary interactions and transition to turbulence for standard cfd problems using SPH," *Proc. 6th International Computational Techniques and Applications Conference*, Canberra, 1993, pp. 157–165.

[49] Cockrell Jr., C. E., Auslender, A. H., and White, J. A., "Aeroheating predictions for the X-43 cowl-closed configuration at mach 7 and 10," *AIAA 40th AIAA Aerospace Sciences Conference and Exhibit*, Reno, Nevada, 2002, pp. Paper 2002–0218.

[50] Coles, D. E. and Hirst, E. A., "Computation of turbulent boundary layers," *1968 AFOSR-IFP-Stanford Conference, Volume II*, Thermosciences Division, Stanford University, 1969.

[51] Cox, A., Dwarkadas, S., Keleher, P., Lu, H., Rajamony, R., and Zwaenepoel, W., "Software vs. hardware shared memory implementation: A case study," *Proceedings of the Twenty-first Symposium on Computer Architecture*, Vol. April, 1994, pp. 106–117.

[52] Craddock, C. S., "A quasi-one-dimensional space-marching flow solver with finite rate chemical effects," Department of Mechanical Engineering Report 7/1996, The University of Queensland, Brisbane, October 1996.

[53] Craddock, C. S., *Computational Optimisation of Scramjets and Shock Tunnel Nozzles*, Ph.D. thesis, The University of Queensland, Brisbane, Queensland, Australia, 1999.

[54] Craddock, C. S., Jacobs, P. A., and Gammie, R., "Operational instructions for the small shock tunnel at UQ," Department of Mechanical Engineering Report 8/1998, The University of Queensland, Brisbane, July 1998.

[55] Davies, W. R. and Bernstein, L., "Heat transfer and transition to turbulence in the shock-induced boundary layer on a semi-infinite flat plate," *Journal of Fluid Mechanics*, Vol. 36, 1969, pp. 87–112.

[56] Decyk, V. K., "How to write (nearly) portable fortran programs for parallel computers," *Computers in Physics*, Vol. 7, No. 4, 1993, pp. 418–424.

[57] Dewey, J. M. and Anson, W., "Calibration of shock tubes for studying aerodynamic loading," *The Physics of Fluids, Supplement 1, Shock Tube Symposium*, 1969, pp. 140–143.

[58] Dilley, A. D., "Turbulent heating prediction techniques and comparison with hypersonic experimental data," NASA Contractor Report CR-2001-210837, 2001.

[59] Dilts, G. A., "Moving least squares particle hydrodynamics - i. consistency and stability," *International Journal of Numerical Methods in Engineering*, Vol. 44, 1999, pp. 1115–1155.

[60] Dilts, G. A., "Moving least squares particle hydrodynamics - ii. conservation and boundaries," *International Journal of Numerical Methods in Engineering*, Vol. 48, 2000, pp. 1503–1524.

[61] DLR Germany, Institute for Aerodynamics and Flow Technology, *HEG - High Enthalpy Shock Tunnel Goettingen*, luna.sm.go.dlr.de/index.php?page=heg, Accessed February 2003.

[62] Don, W. S. and Quillen, C. B., "Numerical simulation of shock-cylinder interactions, 1. resolution," *Journal of Computational Physics*, Vol. 122, 1995, pp. 244–265.

[63] Dongarra, J., "Overview of HPC," Tech. rep., Mathematical Sciences Section, Oak Ridge National Laboratory, 1996.

[64] Doolan, C. J. and Jacobs, P. A., "Modeling mass entrainment in a quasi-one-dimensional shock tube code," *AIAA Journal*, Vol. 34, No. 6, 1996, pp. 1291–1293.

[65] Duff, R. E., "Shock-tube performance at low initial pressure," *The Physics of Fluids*, Vol. 2, No. 2, 1959, pp. 207–216.

[66] Dumitrescu, L., Brun, R., and Sides, J., "Computation of gas parameters and compensation of attenuation effects in real shock-tube flows," *Recent Developments in Shock Tube Research, Proceedings of the Ninth International Shock Tube Symposium*, edited by D. Bershader and W. Griffith, Stanford University Press, Stanford University, California, 1973, pp. 439–448.

[67] Dumitrescu, L. Z., Popescu, C., and Brun, R., "Experimental studies of the shock reflection and interaction in a shock tube," *Proceedings of the Seventh International Shock Tube Symposium*, University of Toronto, Toronto, Canada, 1969, pp. 751–770.

[68] Dumitrescu, M. P., "Trapping the boundary layer: A method to diminish flow contamination in shock tunnels," *Proceedings of the 20th International Symposium on Shock Waves, Volume 2*, World Scientific, Singapore, 1996, pp. 1581–1586.

[69] Earth Simulator Center, *Earth Simulator*, http://www.es.jamstec.go.jp/esc/eng/, Accessed December 2002.

[70] Flynn, M. J., "Very high speed computing systems," *Proceedings of IEEE*, Vol. 54, 1966, pp. 1901–1909.

[71] Fortune, S. and Wyllie, J., "Parallelism in random access machines," *Proceedings of the ACM Symposium on the Theory of Computing*, 1978, pp. 114–118.

[72] Fuehrer, R. G., "Measurements of incident-shock test time and reflected shock pressure at fully turbulent boundary-layer test conditions," *Shock Tubes, Proceedings of the Seventh International Shock Tube Symposium*, edited by I. Glass, University of Toronto Press, University of Toronto, Toronto, Canada, 23-25 June 1969, pp. 31–59.

[73] Fulk, D. A., *A numerical analysis of smoothed particle hydrodynamics*, Ph.D. thesis, Air Force Institute of Technology, Air University, College Park, MD, 1994.

[74] Fulk, D. A. and Quinn, D. W., "An analysis of 1-D smoothed particle hydrodynamics kernels," *Journal of Computational Physics*, Vol. 126, 1996, pp. 165–180.

[75] Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Manchek, R., and Sunderam, V., *PVM: Parallel Virtual Machine*, MIT Press, Cambridge, Massachusetts, 1994.

[76] Germain, P. D. and Hornung, H. G., "Transition on a slender cone in hypervelocity flow," *Exp. Fluids*, Vol. 22, 1997, pp. 183–190.

[77] Gibbons, A. and Spirakis, P., *Lectures on Parallel Computation*, Cambridge University Press, Cambridge, Great Britain, 1993.

[78] Gingold, R. and Monaghan, J. J., "Smoothed particle hydrodynamics: Theory and application to non-spherical stars," *Mon. Not. R. Astr. Soc.*, Vol. 181, 1977, pp. 375–389.

[79] Glass, I. I. and Patterson, G. N., "A theoretical and experimental study of shock-tube flows," *Journal of Aeronautical Sciences*, Vol. 22, 1955, pp. 73–100.

[80] Gnoffo, P. A., Weilmuenster, K. J., and Alter, S. J., "Multiblock analysis for shuttle orbiter re-entry heating from mach 24 to mach 12," *Journal of Spacecraft and Rockets*, Vol. 31, No. 3, May-June 1994, pp. 367–377.

[81] Gnoffo, P. A., Weilmuenster, K. J., Hamilton II, H. H., Olynick, D. R., and Venkatapathy, E., "Computational aerothermodynamic design issues for hypersonic vehicles," *Journal of Spacecraft and Rockets*, Vol. 36, No. 1, January-February 1999, pp. 21–43.

[82] Gotoh, H., Shibahara, T., and Sakai, T., "Sub-particle-scale turbulence model for the mps method - lagrangian flow model for hydraulic engineering," *Computational Fluid Dynamics Journal*, Vol. 9, No. 4, 2001, pp. 339–347.

[83] Graduate Aeronautical Labotaroties, California Institute of Technology, *The T5 Hypervelocity Shock Tunnel Facility*, http://www.galcit.caltech.edu/T5/, Accessed February 2003.

[84] Granville, P. S., "Baldwin-lomax factors for turbulent boundary layers in pressure gradients," *AIAA Journal*, Vol. 25, No. 12, 1987, pp. 1624–1627.

[85] Griffith, B. J., Maus, J. R., Majors, B. M., and Best, J. T., "Addressing the hypersonic simulation problem," *Journal of Spacecraft*, Vol. 24, No. 4, July-August 1987, pp. 334–341.

[86] Gropp, W. and Lusk, E., "Installation guide to MPICH, a portable implementation of MPI. version 1.2.0," *ANL/MCS-TM-ANL-96/5 Rev B*, 1996, pp. 1–63.

[87] Gropp, W., Lusk, E., and Skjellum, A., *Using MPI: Portable Parallel Programming with the Message Passing Interface*, The MIT Press, Cambridge, Massachusetts, 1994.

[88] Ha, J., Ahuja, V., and Cleary, P., "Comparison of SPH simulation of high speed die filling with experiment," *13th Australian Fluid Mechanics Conference*, 1998, pp. 901–904.

[89] Haas, J. F. and Sturtevant, B., "Interaction of weak shock waves with cylindrical and spherical gas inhomogeneities," *Journal of Fluid Mechanics*, Vol. 181, 1987, pp. 41–76.

[90] Hannemann, K., Jacobs, P. A., Austin, J. M., Thomas, A., and McIntyre, T. J., "Transient and steady-state flow in a small shock tube," *21st International Symposium on Shock Waves*, Great Keppel Island, Australia, July 20-25, 1997, p. Paper 2630.

[91] Harlow, F. H., "The particle-in-cell computing method in fluid dynamics," *Methods in Computational Physics*, Vol. 3, 1964, pp. 319–342.

[92] He, J. and Walker, J. D. A., "A note on the baldwin-lomax turbulence model," *Transactions of the ASM*, Vol. 117, 1995, pp. 528–531.

[93] He, Y. and Morgan, R., "Transition of compressible high enthalpy boundary layer flow over a flat plate," *Aero. J.*, Vol. 98, No. 25, 1994, pp. 25–34.

[94] Henderson, L. F., Colella, P., and Pucket, E. G., "On the refraction of shock waves at a slow-fast gas interface," *Journal of Fluid Mechanics*, Vol. 224, 1991, pp. 1–27.

[95] Henneken, E. and Icke, V., "SPH faces emery's jump," *Computers in Physics Communications*, Vol. 74, 1993, pp. 239–246.

[96] Hernquist, L., "Some cautionary remarks about smoothed particle hydrodynamics," *Astrophysics Journal*, Vol. 404, 1993, pp. 717–722.

[97] Hernquist, L. and Katz, N., "TREESPH: A unification of SPH with the hierarchical tree method," *The Astrophysical Journal Supplement Series*, Vol. 70, 1989, pp. 419–446.

[98] Hertzberg, A., "Shock tube research, past, present and future: A critical survey," *Proceedings of the Seventh International Shock Tube Symposium*, University of Toronto, Toronto, Canada, 1969, p. editorial.

[99] Hickman, R. S., Farrar, L. C., and Kyser, J. B., "Behaviour of burst diaphragms in shock tubes," *The Physics of Fluids*, Vol. 18, No. 10, 1975, pp. 1249–1252.

[100] High Performance Fortran Forum, *High Performance Fortran Home Page*, http://www.crpc.rice.edu/HPFF/home.html, Accessed December 2002.

[101] Hill, M. D. and Larus, J. R., "Cache considerations for multiprocessor programmers," *Communications of the ACM*, Vol. 33, No. 8, 1990, pp. 97–102.

[102] Hiscock, S., Kilpin, D., and Drummond, L., "A versatile shock tube and its analytical instrumentation," Tech. Rep. 1819(W), Department of Defence, Salisbury, South Australia, 1977.

[103] Hockney, R. W. and Eastwood, J. W., *Computer Simulation Using Particles*, Adam Hilger, Bristol, Great Britain, 1988.

[104] Hooker, W. J., "Testing time and contact-zone phenomena in shock-tube flows," *The Physics of Fluids*, Vol. 4, No. 12, 1961, pp. 1451–1463.

[105] Hornung, H., "Experimental hypervelocity flow simulation, needs, achievements, and limitations," *First Pacific International Conference on Aerospace Science and Technology*, National Cheng-Kung University, Tainan, Taiwan, 1993, pp. 1–10.

[106] Houwing, A. F. P., Palma, P. C., Danchy, P. M., Fox, J. S., O'Byne, S., Mere, P., Gaston, M. J., Mallinson, S. G., and Cooper, M., "Fluorescence-imaging-based flow diagnostics applications to free piston shock tunnel flows," *22nd International Symposium on Shock Waves*, Imperial College, London, 1999.

[107] Hwang, K. and Zhiwei, X., *Scalable Parallel Computing : Technology, Architecture and Programming*, WCB/McGraw-Hill, Boston, 1998.

[108] Hyshot Scramjet test programme, *Centre for Hypersonics, The University of Queensland*, http://www.mech.uq.edu.au/hyper/hyshot/, Accessed February 2003.

[109] Jacobs, J. W., "Shock induced mixing of a light-gas cylinder," *Journal of Fluid Mechanics*, Vol. 234, 1992, pp. 629–649.

[110] Jacobs, P. A., "Single-block Navier-Stokes integrator," Interim Report 18, ICASE, 1991.

[111] Jacobs, P. A., "Transient, hypervelocity flow in an axisymmetric nozzle," Tech. Rep. ICASE Report 91-01, Institute for Computer Applications in Science and Engineering, NASA Langley Research, Hampton, VA, 1991.

[112] Jacobs, P. A., "An approximate Riemann solver for hypervelocity flows," *AIAA Journal*, Vol. 30, No. 10, 1992, pp. 2558–2561.

[113] Jacobs, P. A., "Numerical simulation of transient hypervelocity flow in an expansion tube," *Computers Fluids*, Vol. 23, No. 1, 1994, pp. 77–101.

[114] Jacobs, P. A., "MB_CNS: A computer program for the simulation of transient compressible flows," Department of Mechanical Engineering Report 10/1996, The University of Queensland, Brisbane, October 1996.

[115] Jenkins, D. M., Stalker, R. J., and Morrison, W. R. B., "Performance considerations in the operation of free-piston driven hypersonic test facilities," *Proceedings of the 18th International Symposium on Shock Waves*, Berlin, 1992, pp. 597–602.

[116] John D. Anderson, J., *Computational Fluid Dynamics: The Basics with Applications*, McGraw-Hill, New York, 1995.

[117] John J. Lacey, J., "Experimental shock tube test time - turbulent regime," *Shock Tubes, Proceedings of the Seventh International Shock Tube Symposium*, edited by I. Glass, University of Toronto Press, University of Toronto, Toronto, Canada, 23-25 June 1969, pp. 126–142.

[118] Kaneko, M., Men'shov, I., and Nakamura, Y., "Computation of nozzle starting process with thermal and chemical nonequilibrium in high-enthalpy shock tunnel," *AIAA 40th Aerospace Sciences Meeting and Exhibit*, AIAA, Reno, Nevada, January, 2002, pp. Paper AIAA–2002–0142.

[119] Kaufmann III, W. J. and Smarr, L. L., *Supercomputing and the Transformation of Science*, Scientific American Library, New York, 1993.

[120] Kernighan, Brian, W. and Ritchie, D. M., *The C Programming Language*, Prentice Hall, New Jersey, 1978.

[121] Kim, S. C., Harloff, G. J., and Sverdup, H., "Hypersonic turbulent wall boundary layer computations," *AIAA/ASME/SAE/ASEE 24th Joint Propulsion Conference*, AIAA, Washingtion, D.C., 1988, pp. AIAA–88–2829.

[122] Kinsey, D. W. and Eastep, F. E., "Turbulence modeling in separated flow behind strong shocks," *Journal of Aircraft*, Vol. 26, No. 2, 1989, pp. 185–186.

[123] Kuck, D. J., "What is good parallel performance? and how do we get it?" *IEEE Computational Science and Engineering*, Vol. Spring, 1996, pp. 81–85.

[124] Kuck and Associates, Inc. Software, *KAP/Pro for OpenMP*, http://www.kai.com/parallel/kappro/, Accessed December 2002.

[125] Lattanzio, J. C., Monaghan, J. J., Pongracic, H., and Schwarz, M. P., "Controlling penetration," *SIAM Journal of Scientific and Statistical Computing*, Vol. 7, No. 2, 1986, pp. 591–598.

[126] Lecomber, D. S., Siniolakis, C. J., and Sujithan, K. R., "PRAM programming: In theory and in practice," *Concurrency: Practice and Experience*, Vol. 12, 2000, pp. 211–226.

[127] Lee, C., *Transition and Heat Transfer in Compressible Boundary Layer Flow Over a Flat Plate*, Ph.D. thesis, The University of Queensland, Brisbane, Queensland, Australia, 1991.

[128] Lee, H. G., Murakami, T., and Nishida, M., "Computational and experimental studies of unsteady viscous nozzle flows," *JSME International Journal Series B: Fluids and Thermal Engineering*, Vol. 38, No. 3, 1995, pp. 346–352.

[129] Lee, J. Y., *A numerical study of the starting process in a hypersonic shock tunnel*, Ph.D. thesis, The University of Maryland, College Park, MD, 1993.

[130] Lee, J. Y. and Lewis, M. J., "Numerical study of the flow establishment time in hypersonic shock tunnels," *Journal of Spacecraft and Rockets*, Vol. 30, No. 2, 1993, pp. 152–163.

[131] Lee, M.-G. and Nishida, M., "Numerical simulation of starting process in a hypersonic nozzle," *Memoirs of the Faculty of Engineering, Kyushu University*, Vol. 52, No. 3, September 1992, pp. 311–317.

[132] Lee, S. H., Jeung, I. S., and Yoon, Y., "Computational investigation of shock-enhanced mixing and combustion," *AIAA Journal*, Vol. 35, No. 12, 1997, pp. 1813–1820.

[133] Leiss, Ernst, L., *Parallel and Vector Computing*, McGraw-Hill, New York, 1995.

[134] Li, S. and Liu, W. K., "Moving least squares reproducing kernel method. part ii: Fourier analysis," *Computer Methods in Applied Mechanical Engineering*, Vol. 139, 1996, pp. 159–193.

[135] Liou, M. S. and Steffen, C. J., "A new flux splitting scheme." NASA Technical Memorandum 104404, 1991.

[136] Loewenstein, M. and Mathews, W. G., "Adiabatic particle hydrodynamics in three dimensions," *Journal of Computational Physics*, Vol. 62, 1986, pp. 414–428.

[137] Lucy, L., "A numerical approach to the testing of the fission hypothesis," *The Astronomical Journal*, Vol. 82, No. 12, 1977, pp. 1013–1024.

[138] Lukasiewicz, J. L., *Experimental methods of hypersonics*, Marcel Dekker, New York, 1973.

[139] Macrossan, M. N., "The equilibrium flux method for the calculation of flows with non-equilibrium chemical reactions," *Journal of Computational Physics*, Vol. 80, No. 1, 1989, pp. 204–231.

[140] Macrossan, M. N. and Oliver, R. I., "A kinetic theory solution method for the navier-stokes equations," *International Journal for Numerical Methods in Fluids*, Vol. 17, 1993, pp. 177–193.

[141] Mark, H., "The interaction of a reflected shock wave with the boundary layer in a shock tube," *Journal of the Aeronautical Sciences*, Vol. April, 1957, pp. 304–306.

[142] Markstein, G. H., "Flow disturbances induced near a slightly wavy contact surface or flame front traversed by a shock wave," *Journal of the Aeronautical Sciences*, Vol. 24, 1957, pp. 238.

[143] Marvin, J. G., "Turbulence modeling for hypersonic flows," Tech. rep., NASA Technical Memorandum 101079, Ames Research Center, Moffett Field, California, 1989.

[144] Matsuo, K., Kage, K., and Kawagoe, S., "The interaction of a reflected shock wave with the contact region in a shock tube," *Bulletin of the JSME*, Vol. 18, No. 121, 1975, pp. 681–688.

[145] Mc.Ghee, A. M. and Jacobs, P. A., "Parallel computation of hypervelocity flow," *Computational Techniques and Applications: CTAC95*, World Scientific, 1995, pp. 1–7.

[146] Meshkov, "Instability in the interface of two gases accelerated by a shock wave," *Soviet Fluid Dynamics*, Vol. 4, 1970, pp. 101–108.

[147] Mirels, H., "Test time in low-pressure shock tubes," *Physics of Fluids*, Vol. 6, 1963, pp. 1201–1214.

[148] Mirels, H., "Shock tube test time limitation due to turbulent-wall boundary layer," *AIAA Journal*, Vol. 2, No. 1, 1964, pp. 84–93.

[149] Mirels, H., "Correlation formulas for laminar shock tube boundary layer," *The Physics of Fluids*, Vol. 9, No. 7, 1966, pp. 1265–1272.

[150] Monagan, J. J. and Gingold, R., "Shock simulation by the particle method SPH," *Journal of Computational Physics*, Vol. 52, 1983, pp. 374–389.

[151] Monaghan, J. J., "Why particle methods work," *Siam Journal of Scientific and Statistical Computing*, Vol. 3, No. 4, 1982, pp. 423–433.

[152] Monaghan, J. J., "Applications of the particle method SPH to hypersonic flow," *Computational Techniques and Applications: CTAC-85*, 1986, pp. 357–365.

[153] Monaghan, J. J., "An introduction to SPH," *Computers in Physics Communications*, Vol. 48, 1988, pp. 89–96.

[154] Monaghan, J. J., "On the problem of penetration in particle methods," *Journal of Computational Physics*, Vol. 82, 1989, pp. 1–15.

[155] Monaghan, J. J., "Simulating free surface flows with SPH," *Journal of Computational Physics*, Vol. 110, 1992, pp. 399–406.

[156] Monaghan, J. J., "Smoothed particle hydrodynamics," *Annual Review of Astronomy and Astrophysics*, Vol. 30, 1992, pp. 543–574.

[157] Monaghan, J. J., "Particle methods: A review," *Computational Techniques and Applications: CTAC 1993*, 1993, pp. 1–16.

[158] Monaghan, J. J., "SPH and riemann solvers," *Journal of Computational Physics*, Vol. 136, 1997, pp. 298–307.

[159] Monaghan, J. J. and Pongracic, H., "Artificial viscosity for particle methods," *Applied Numerical Mathematics*, Vol. 1, 1985, pp. 187–194.

[160] Morris, J. P. and Monaghan, J. J., "A switch to reduce SPH viscosity," *Journal of Computational Physics*, Vol. 136, 1997, pp. 41–50.

[161] MPI Forum, *MPI Forum Home Page*, http://www.mpi-forum.org/, Accessed December 2002.

[162] MPI Software Technology, Inc., *MPI/Pro Webpage*, http://www.mpi-softtech.com/, Accessed December 2002.

[163] National Aerospace Laboratory of Japan, *Kakuda Space Propulson Laboratory - Test Facilities*, http://www.nal.go.jp/krc/eng/tf/hiest.htm, Accessed February 2003.

[164] Nishida, M. and Kishige, H., "Numerical simulation of focusing process of reflected shock waves," *Shock tubes and waves : proceedings of the Sixteenth International Symposium on Shock Tubes and Waves*, edited by H. Gronig, VCH, Aachen, West Germany, July 26-31, 1987, pp. 551–557.

[165] Nishiguchi, A. and Yabe, T., "Second order fluid particle scheme," *Journal of Computational Physics*, Vol. 52, 1983, pp. 390–413.

[166] Noguchi, S. and Ota, M., *Correct Models for Parallel Computing*, IOS Press, 1997.

[167] Okui, H., Hiraki, H., Park, M.-K., Oshima, S., and Yamane, R., "Control of pseudo-shock by slot injection," *JSME International Journal, Series B*, Vol. 45, No. 1, 2002, pp. 150–157.

[168] OpenMP Architecture Review Board, *OpenMP Webpage*, http://www.openmp.org/, Accessed December 2002.

[169] Outa, E., Tajima, K., and Hayakawa, K., "Shock tube flow influence by diaphragm opening (two-dimensional flow near the diaphragm)," *10th international symposium on shock waves and shock tubes*, 1975, pp. 312–319.

[170] Owen, J. M., Villumsen, J. V., Shapiro, P. R., and Martel, H., "Adaptive smoothed particle hydrodynamics: Methodology. ii." *The Astrophysical Journal Supplement Series*, Vol. 116, 1998, pp. 155–209.

[171] Oxford University, *Oxford BSP Toolkit BSPlib*, http://www.bsp-worldwide.org/implmnts/oxtools.htm, Accessed December 2002.

[172] Pate, S. R. and Schueler, C. J., "Radiated aerodynamic noise effects on boundary layer transition in supersonic and hypersonic wind tunnels," *AIAA Journal*, Vol. 7(3), 1969, pp. 450–457.

[173] Paull, A., "A simple shock-tunnel driver-gas detector," 1996, pp. 1557–1562.

[174] Paull, A. and King, M. D., "A driver gas detection device for shock tunnels," *Shock Waves*, Vol. 4, No. 5, 1995, pp. 289–291.

[175] Paull, A. and Stalker, R. J., "Test flow disturbances in an expansion tube," *Journal of Fluid Mechanics*, Vol. 245, 1992, pp. 493–521.

[176] Petrie-Repar, P. J., *Numerical simulation of diaphragm rupture*, Ph.D. thesis, The University of Queensland, Brisbane, Queensland, Australia, 2000.

[177] Petrie-Repar, P. J. and Jacobs, P. A., "A computational study of shock speeds in high-performance shock tubes," *Shock Waves*, Vol. 8, 1998, pp. 79–91.

[178] Physics Department, the Australian National University, *Aerophysics and Laser Diagnostics Research (ALDiR) Laboratory*, http://www.anu.edu.au/Physics/aldir/, Accessed February 2003.

[179] Picone, J. M. and Boris, J. P., "Vorticity generation by shock propagation through bubbles in a gas," *Journal of Fluid Mechanics*, Vol. 189, 1988, pp. 23–51.

[180] Pullin., D. I., "Direct simulation methods for compressible inviscid ideal-gas flow," *Journal of Computational Physics*, Vol. 34, No. 2, 1980, pp. 231–244.

[181] Pulsonetti, M. V., "The purpose of scramjets," Department of Mechanical Engineering Report 5/1995, The University of Queensland, Brisbane, October 1995.

[182] Quirk, J. J., "A contribution to the great Riemann solver debate," ICASE Report 92-64, Institute for Computer Applications in Science and Engineering, NASA Langley Research Center, 1992.

[183] Quirk, J. J., "A contribution to the great Riemann solver debate," *International Journal for Numerical Methods in Fluids*, Vol. 18, No. 6, 1994, pp. 555–574.

[184] Quirk, J. J. and Karni, S., "On the dynamics of a shock-bubble interaction," NASA Contractor Report 194978, ICASE Report No. 94-75, Institute for Computer Applications in Science and Engineering, NASA Langley Research Center, Hampton, Virginia, 1994.

[185] Randles, P. and Libersky, L., "Smoothed particle hydrodynamics: Some recent improvements and applications," *Computer Methods in Applied Mechanics and Engineering*, Vol. 139, 1996, pp. 375–408.

[186] Randles, P. W. and Libersky, L. D., "Normalized SPH with stress points," *International Journal for Numerical Methods in Engineering*, Vol. 48, 2000, pp. 1445–1462.

[187] Rasio, F. A. and Lombardi Jr., J. C., "Smoothed particle hydrodynamics calculations of stellar interactions," *Journal of Computational and Applied Mathematics*, Vol. 109, 1999, pp. 213–230.

[188] Reichl, P., Morris, P., Hourigan, K., Thompson, M., and Stoneman, S., "Smooth particle hydrodynamics simulation of surface coating," *Applied Mathematical Modelling*, Vol. 22, 1998, pp. 1037–1046.

[189] Reid, R., Prausnitz, J., and Poling, B., *The Properties of Gases and Liquids: 4th edition*, McGraw Hill, New York, 1987.

[190] Richardson, H., "High performance fortran: History, overview and current status," Tech. rep., Edinburgh Parallel Computing Centre, The University of Edinburgh, Scotland, 1995.

[191] Richardson, L. F., *Weather Prediction by Numerical Processes*, 1922.

[192] Richtmyer, R. D., "Taylor instability in shock acceleration of compressible fluids," *Communications in Pure and Applied Mathematics*, Vol. 8, 1960, pp. 297–319.

[193] Robinson, N., "Simulation of gas jetting through a diaphragm in a shock tube using smoothed particle hydrodynamics," *Bachelor of Engineering Thesis*, 1994.

[194] Rodriguez, C., Roda, J. L., Sande, F., Morales, D. G., and F., A., "A new parallel model for the analysis of asynchronous algorithms," *Parallel Computing*, Vol. 26, 2000, pp. 753–767.

[195] Roshko, A., "On flow duration in low-pressure shock tubes," *The Physics of Fluids*, Vol. 3, No. 6, 1960, pp. 835–842.

[196] Rothkopf, E. M. and Low, W., "Shock formation distance in a pressure driven shock tube," *The Physics of Fluids*, Vol. 19, No. 12, 1967, pp. 1885–1888.

[197] Rothkopf, E. M. and Low, W., "Diaphragm opening process in shock tubes," *The Physics of Fluids*, Vol. 17, No. 6, 1974, pp. 1169–1173.

[198] Rudinger, G., "Effect of boundary-layer growth in a shock tube on shock reflection from a closed end," *The physics of fluids*, Vol. 4, No. 12, 1961, pp. 1463–1473.

[199] Salmon, J., Stein, C., and Sterling, T., "Scaling of beowulf-class distributed systems," Tech. rep., Center for Advanced Computing Research, California Institute of Technology, 1998.

[200] Sanderson, R. J., "Interpretation of pressure measurements behind the reflected shock in a rectangular shock tube," *AIAA Journal*, Vol. 7, No. 7, 1969, pp. 1370–1372.

[201] Satofuka, N., "A numerical study of shock formation in cylindrical and two dimensional shock tubes," Tech. Rep. 451, ISAS, University of Tokyo, Japan, June 1970.

[202] Schneider, S. P., "Effects of high-speed tunnel noise on laminar-turbulent transition," *Journal of spacecraft and rockets*, Vol. 38, 2001, pp. 323–.

[203] Schultz, D. L. and Jones, T. V., "Heat-transfer measurements in short-duration hypersonic facilities," Agardograph no. 165, Advisory Group for Aerospace Research and Development, Department of Engineering Science, University of Oxford, Gt. Britain, 1973.

[204] SGI, "Cray hardware solutions," *Product Brochure*, 1999.

[205] Shapiro, P. R., Martel, H., Villumsen, J. V., and Owen, J. M., "Adaptive smoothed particle hydrodynamics, with appplication to cosmology: Methodology," *The Astrophysical Journal Supplement Series*, Vol. 103, 1996, pp. 269–330.

[206] Sharma, S. P. and Wilson, G. J., "Test times in hypersonic shock tubes," *AIAA 33rd Aerospace Sciences Meeting and Exhibit*, AIAA, Reno, Nevada, 1995, pp. Paper AIAA–95–0713.

[207] Sharma, S. P. and Wilson, G. J., "Computations of axisymmetric flows in hypersonic shock tubes," *Journal of Thermophysics and Heat Transfer*, Vol. 10, No. 1, 1996, pp. 169–176.

[208] Simon, H. D. and Dagum, L., "Experience in using SIMD and MIMD parallelism for computational fluid dynamics," Tech. rep., Numerical Aerodynamic Simulation Systems Division, NASA Ames Research Center, Moffet Field, CA, 1991.

[209] Simpson, C. J. S., Chandler, T. R. D., and Bridgman, K. B., "Effect of shock trajectory of the opening time of diaphragms in a shock tube," *The Physics of Fluids*, Vol. 10, No. 9, 1967, pp. 1894–1896.

[210] Skillicorn, D. B. and Talia, D., "Models and languages for parallel computation," *ACM Computing Surveys*, Vol. 30, No. 2, 1998, pp. 124–169.

[211] Skinner, K. and Stalker, R., "Mass spectrometer measurements of test gas composition in a shock tunnel," *AIAA Journal*, Vol. 34, No. 1, 1995, pp. 203–205.

[212] Skinner, K. A., *Mass spectrometry in shock tunnel experiments of hypersonic combustion*, Ph.D. thesis, The University of Queensland, Brisbane, Queensland, Australia, 1994.

[213] Smeets, G., "Turbulent boundary layers in high enthalpy flows. numerical code valuation by shock tube experiments," *17th International Symposium on Shock Waves and Shock Tubes*, edited by Y. W. Kim, American Institute of Physics, Lehigh University, Bethlehem, Pa, 1989, pp. 321–326.

[214] Smith, C. E., "The starting process in a hypersonic nozzle," *Journal of Fluid Mechanics*, Vol. 24, 1966, pp. 625–640.

[215] Snir, M., Otto, S. W., Huss-Lederman, S., Walker, D. W., and Dongarra, J., *MPI: The Complete Reference*, The MIT Press, Cambridge, Massachusetts, 1996.

[216] Snow, C. R., *Concurrent Programming*, Cambridge University Press, Cambridge, Great Britain, 1992.

[217] Spalart, P. R. and Allmaras, S. R., "A one-equation turbulence model for aerodynamic flows," AIAA paper, 1992.

[218] Speith, R. and Riffert, H., "The viscous gas ring as an astrophysical test problem for a viscous SPH-code," *Journal of Computational and Applied Mathematics*, Vol. 109, 1999, pp. 231–242.

[219] Stalker, R. J., "An investigation of free piston compression of shock tube driver gas," Report mt-44, Division of Mechanical Engineering, National Research Council, Canada, 1961.

[220] Stalker, R. J., "The free-piston shock tube," *The Aeronautical Quarterly*, November 1966, pp. 351–371.

[221] Stalker, R. J. and Crane, K. C. A., "Driver gas contamination in a high-enthalpy reflected shock tunnel," *AIAA Journal*, Vol. 16, 1978, pp. 277–278.

[222] Steinmetz, M. and Muller, E., "On the capabilities and limits of smoothed particle hydrodynamics," *Astronomy and Astrophysics*, Vol. 268, No. 1, 1992, pp. 391–410.

[223] Sterling, T. L., Salmon, J., Becker, D. J., and Savarese, D. F., *How to Build a Beowulf: A Guide to the Implementation and Application of PC Clusters*, The MIT Press, Cambridge, Massachusetts, 1999.

[224] Strohmaier, E., Dongarra, J. J., Meuer, H. W., and Simon, H. D., "The marketplace of high-performance computing," *Parallel Computing*, Vol. 25, 1999, pp. 1517–1544.

[225] Sturtevant, B., "Rayleigh-taylor instability in compressible fluids," *Shock tubes and waves : proceedings of the Sixteenth International Symposium on Shock Tubes and Waves*, edited by H. Gronig, VCH, Aachen, West Germany, July 26-31, 1987, pp. 89–100.

[226] Sudani, N. and Hornung, H. G., "Detection and reduction of driver gas contamination in a high-enthalpy shock tunnel," *Proceedings of the 21st International Symposium on Shock Waves*, Panther Publishing, Fyshwick, Australia, 1997, pp. 543–548.

[227] Sudani, N. and Hornung, H. G., "Detection of driver gas contamination in the t5 hypervelocity shock tunnel," AIAA paper, 1997.

[228] Sudani, N., Valiferdowsi, B., and Hornung, H. G., "Test time increase by delaying driver gas contamination for reflected shock tunnels," *AIAA Journal*, Vol. 38, No. 9, 2000, pp. 1497–1503.

[229] Sunderam, V. and Geist, G., "Heterogeneous parallel and distributed computing," *Parallel Computing*, Vol. 25, 1999, pp. 1699–1721.

[230] Sutherland, W., "The viscosity of gases and molecular force," *Phil. Mag.*, Vol. 5, 1983, pp. 507–531.

[231] Swegle, J. and Attaway, S., "On the feasibility of using smoothed particle hydrodynamics for underwater explosion calculations," *Computational Mechanics*, Vol. 17, 1995, pp. 151–168.

[232] Taylor, G. I., "The instability of liquid surfaces when accelerated in a direction perpendicular to their planes," *Proceedings of the Royal Society A*, Vol. 201, 1950, pp. 192–196.

[233] Taylor, J. R. and Hornung, H. G., "Real gas and wall roughness effects on the bifurcation of the shock reflected from the end wall of a tube," *Shock Tubes and Waves, Proceedings of the 13th Internation symposium on shock tubes and waves*, State University of New York Press, 1981, pp. 262–270.

[234] Tokarcik-Polsky, S. and Cambier, J.-L., "Numerical study of transient flow phenomena in shock tunnels," *AIAA Journal*, Vol. 32, No. 5, May 1994, pp. 971–978.

[235] University of Mannheim and University of Tennessee, *Top 500 Supercomputer Sites*, http://www.top500.org/, Accessed December 2002.

[236] University of Notre Dame, *LAM/MPI Parallel Computing*, http://www.lam-mpi.org/, Accessed December 2002.

[237] Valiant, L. G., "A bridging model for parallel computation," *Comm. of ACM*, Vol. 33, No. 8, 1990, pp. 103–111.

[238] Wada, Y. and Liou, M. S., "A flux splitting scheme with high-resolution and robustness for discontinuities," *AIAA 32nd Aerospace Sciences Meeting*, World Scientific, Reno, NV, 1994, pp. AIAA Paper 94–0083.

[239] Watkins, S. J., Bhattal, A. S., Francis, N., Turner, J. A., and Whitworth, A., "A new prescription for viscosity in smoothed particle hydrodynamics," *Astronomy and Astrophysics, Supplement Series*, Vol. 119, 1996, pp. 117–187.

[240] Weber, Y. S., Oran, E. S., Boris, J. P., and Anderson Jr., J. D., "The numerical simulation of shock bifurcation near the end wall of a shock tube," *Physics of Fluids*, Vol. 7, 1995, pp. 2475–2488.

[241] Weilmuenster, K. J., Gnoffo, P. A., and Greene, F. A., "Multiblock analysis for shuttle orbiter re-entry heating from mach 24 to mach 12," *Journal of Spacecraft and Rockets*, Vol. 31, No. 3, May-June 1994, pp. 355–366.

[242] Wheatley, V., *Modelling low-density flow in hypersonic impulse facilities*, Master's thesis, The University of Queensland, Brisbane, Queensland, Australia, 1999.

[243] White, D. R., "Influence of diaphragm opening time on shock-tube flows," *Journal of Fluid Mechanics*, Vol. 4, 1958, pp. 585–599.

[244] Wilcox, D. C., *Turbulence Modelling for CFD*, DCW Inductries, La Canada, California, 1998.

[245] Wilke, C. R., "A viscosity equation for gas mixtures," *Journal of Chemical Physics*, Vol. 18, 1950, pp. 517–519.

[246] Wilson, G. J., "Numerical studies of high-enthalpy impulse facilities," *Proceedings of the 20th International Symposium on Shock Waves (Volume 1)*, World Scientific Publishing Co. Pty. Ltd., California Institute of Technology, Pasadena, California, 1995.

[247] Wilson, G. J., Sharma, S. P., and Gillespie, W. D., "Time-dependent simulations of reflected shock/boundary layer interaction," *AIAA 31rd Aerospace Sciences Meeting and Exhibit*, AIAA, Reno, Nevada, 1993, pp. Paper AIAA–93–0480.

[248] Wittliff, C. E., Wilson, M. R., and Hertzberg, A., "The tailored-interface hypersonic shock tunnel," *Journal of the Aerospace Science*, Vol. 26, No. 4, 1959, pp. 219–228.

[249] Woodward, P. and Colella, P., "The numerical simulation of two-dimensional fluid flow with strong shocks," *Journal of Computational Physics*, Vol. 54, 1984, pp. 115–173.

[250] Worlton, J., "Seymour Cray's contributions to supercomputing and parallel computing," *Parallel Computing*, Vol. 25, 1999, pp. 1569–1582.

[251] Yang, J., Kubota, T., and Zukoski, E. E., "A model for the characterization of a vortex pair formed by shock passage over a light-gas inhomogeneity," *Journal of Fluid Mechanics*, Vol. 258, 1994, pp. 217–244.

[252] York, B. and Knight, D., "Calculation of two-dimensional turbulent boundary layers using the baldwin-lomax model," *AIAA Journal*, Vol. 23, No. 12, 1985, pp. 1849–1850.

[253] Zabusky, N. J., "Vortex paradigm for accelerated inhomogeneous flows: visiometrics for the rayleigh-taylor and richtmyer-meshkov environments," *Annual Reviews of Fluid Mechanics*, Vol. 31, 1999, pp. 495–536.

[254] Zeitoun, D., Brun, R., and Valetta, M.-J., "Shock-tube flow computation including the diaphragm and boundary-layer effects," *Shock Tubes and Waves, Proceedings of the 12th Invernational Symposium on Shock Tubes and Waves*, The Magnes Press, The Hebrew University, Jerusalem, July 16-19, 1979, pp. 180–186.

[255] Zuev, A., Vasilieva, R., and Mirshanov, D., "Turbulent mixing of driven and driver gases in shock tube," *17th International Symposium on Shock Waves and Shock Tubes*, edited by Y. W. Kim, American Institute of Physics, Lehigh University, Bethlehem, Pa, 1989, pp. 153–173.

[256] Zukas, J. A., *High Velocity Impact Dynamics*, John Wiley and Sons, New York, 1990.

APPENDIX A

Performance of the Parallel SPH Code

This appendix will investigate the performance of the parallel implementations of the Smoothed Particle Hydrodynamics (SPH) code. As well as analysing the effect varying parameters on the solution time directly, the effect of parallelism on the performance of the computing hardware, such as cache memory, will also be analysed. Some of the simulations being run as part of the present study are all smaller than would necessarily be parallelised in practice and so efficiencies are generally low.

The sequential performance of a code is dependent on how the code is written, the efficiency of the underlying mathematical algorithms used, the speed of the processor, and how well the computers memory is utilised. In contrast, parallel performance is dependent on many more factors, including: the bandwidth and latency of the network connecting the processors, the delay required for processors to be synchronised, the granularity of the parallelism and the relative amount of the code that must run sequentially. These additional factors make analysing parallel performance complex, requiring the consideration of many different aspects.

Running CFD codes in parallel requires the optimisation of many parameters, one of these being the number of processors to use for the simulation size. The number of processors is important as the parallel overhead, and therefore the parallel efficiency is heavily dependent on the number used. The overhead is increased because of the increased time required to synchronise larger numbers of processors working in parallel; the decreased message size, and therefore, the more significant effect of network latency; and the increased requirements on communication. Using more processors does, on the other hand, provide a larger potential speed-up and access to proportionally larger data caches and, on distributed memory machines, total memory space.

In the SPH technique, when no algorithmic acceleration, such as sorting the particles into cells as described in Section 3.2.1, is used, every particle, N, must account for every other particle in each calculation. This means that the computation time, t, is proportional to N^2. When this is the case, on a graph with logarithmic axes, the line representing the computation time, t, versus the simulation size, N, will be a

straight line with a gradient of two. In contrast, the scaling using a hierarchical tree, for example, is: $N\log(N)$ and so, $\log(t) = \log(N) + \log(\log(N))$. The $\log(\log(N))$ term can be neglected and the gradient of the line on the graph will approach one, for large N. The cell based sorting routine, reduces the amount of computational work from N^2, but not with the same efficiency as the hierarchical tree. This means that, for a particular number of processors, the slope of the line on the graph will be expected to be between one and two, depending on how optimally sized the cells are for the number of particles in the simulation.

In order to investigate the performance of the parallel versions of the SPH code, simulations varying in size from 6,250 particles to 200,000 particles were run. This represents a large range of simulation sizes suitable for two dimensional models. The simulations were run sequentially and using one, two, four and eight processors in parallel using OpenMP, MPI and BSP. The actual libraries used in the tests are discussed in Section 5.1. The parallel versions of the code will be compared to the sequential version of the code. The sequential version contains no parallel constructs and is not specifically structured for parallelism.

The test simulations were run over a total of 150 time steps. Several sections of computation and communication, along with three synchronisations are run each time step. The error resulting from using timing function calls in the program, and those associated with the `time` and `timex` commands are expected to be small in comparison to the time scales of the measurements. Where necessary, especially for short tests, these results were averaged over several tests. The number of time steps lessens the effect of start up effects.

The two computers used in the tests in this appendix, a previous computer of the QPSF, an SGI Origin 2000, and the Beowulf workstation cluster. The SGI Origin 2000 was replaced by an SGI Origin 3400 and the Origin 3400 was used in the performance tests of MB_CNS in Chapter 5. The performance of the MPI based code on the Origin 2000 and the Beowulf cluster will be compared. Given that the interconnection network on the Origin 2000 is significantly faster than that on the workstation cluster it would be pointless to do a direct comparison of their parallel efficiencies; however, a meaningful comparison can be provided by comparing the actual performance to the ideal performance of a particular system.

In the following plots, the slope of the line for the smallest simulations is significantly lower than for the largest simulations. This is due to the influence of the cell sizes being constant across all runs, regardless of the number of particles used. With this algorithm, the size of the cells must be optimised for the number of particles in the simulation, or rather, the density of particles. By not varying the cell size, the sequential performance of the code is predictable, and we will use this

to gauge the performance of the parallel libraries, without variation from sequential code optimisation.

A.0.11 Shared Memory (OpenMP)

Table A.1 shows the run times for the OpenMP parallel SPH code on the Origin 2000 as well as the sequential performance on an R10000 processor. This table shows how the run time is reduced by running the code in parallel as well as the scaling of the code's run time with the number of particles. There is a certain amount of variation in the timing data due to the use of shared processors on the system and network contention; however, this should not affect the trends as discussed earlier.

Table A.1: Run times of the OpenMP parallel SPH code on the Origin 2000 (in seconds).

N	Sequential	Number of processors			
		1	2	4	8
6250	6.58	6.89	4.62	3.04	2.44
12500	23.26	23.83	14.01	9.73	6.44
25000	78.37	79.10	45.20	27.30	18.22
50000	346.68	337.59	198.91	107.02	60.34
100000	1352.76	1301.76	717.43	384.93	232.81
200000	4805.65	4963.13	2672.26	1453.14	851.47

These results are plotted in Figure A.1, showing the overall trends present in the scaling of the solution time with simulation size. The figure shows that the solution time using the OpenMP code with a single processor is close to that of the sequential code. Any overheads that do not directly result from communication or multi-processor synchronisation would show up in this comparison.

The multiple processor run times scale with the sequential times, with speedups of 1.86, 3.41 and 5.83 for 2, 4 and 8 processors respectively, for the largest simulation size. These figures show that, although the reduction in solution time is quite good, there are still significant overheads present. In the code used in the study, communication is, ideally, transparent to the computation so this is unlikely to be the primary cause.

The relative effect that the overheads incurred by running the code in parallel have on the run time can be investigated by examining the parallel efficiency, that is the actual speedup, divided by the number of processors. Figure A.2 shows the parallel efficiency for the 1, 2, 4 and 8 processor simulations.

The efficiency can be seen to increase as the simulation size is increased. As well as this, the efficiency decrease, for the same sized simulation, as the number of processors is increased. For the largest simulation size, the efficiency is 97%, 93%, 85% and 73%, for 1, 2, 4, and 8 processors respectively. This reflects the decrease in

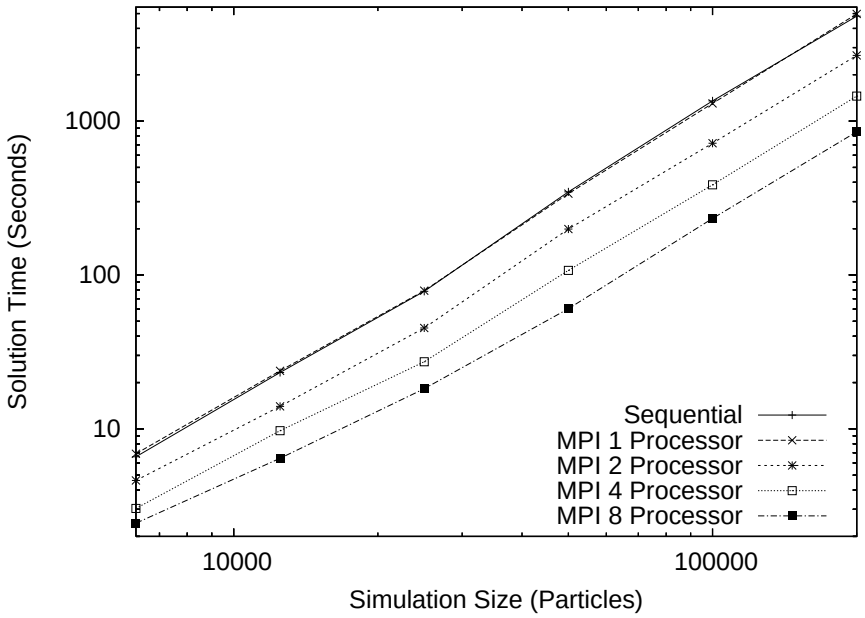

Figure A.1: Run times of the OpenMP parallel SPH code on the SGI Origin 2000.

efficiency with increasing processors, caused by the combined effect of the decreased run time and the increased amount of parallel overheads.

On the Origin 2000, the `speedshop` libraries allows the developer to identify individual run times for parts of a program. Parts of the SPH code were grouped as either `parallel` computation, `sequential` computation or parallel `overhead`. Figure A.3 shows the contribution of the three components to the run time on the left, and their relative proportions on the right. The left most graph shows how the parallel run time is reduced in proportion to the number of processors, how the sequential run time stays constant and the parallel overhead increases. It appears that 98.2% of the sequential code's run time is capable of running in parallel.

On the Origin 2000, at compiler optimisation levels of `O3` or above (`Ofast=ip27`), function inlining and other optimisations make it difficult to identify the parts of the program. For this reason the code was compiled with optimisation level `O2` which, it is hoped, provides a good indication of the performance of the optimised code.

The OpenMP based code performed well for the range of simulation sizes, with efficiency increasing with the simulation size. This means that it is suited to coarser grained parallelism and it is anticipated that the efficiency would further increase for larger simulations. The parallel overheads made up a very small part of the total run time.

Although the OpenMP code is still quite inefficient, it is not our intention to spend a large amount of time optimisation. The OpenMP code is useful as a baseline

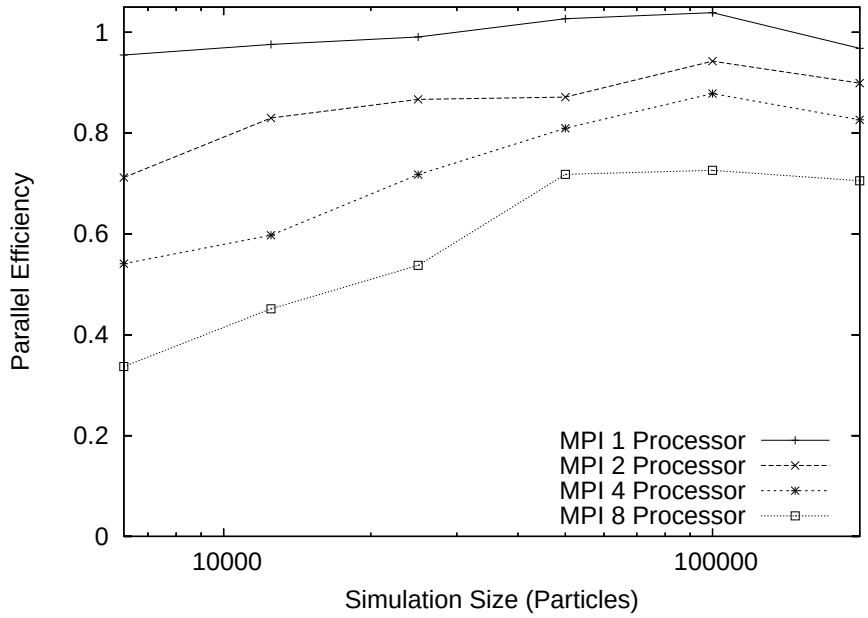

Figure A.2: Parallel efficiency of the OpenMP parallel SPH code on the SGI Origin 2000.

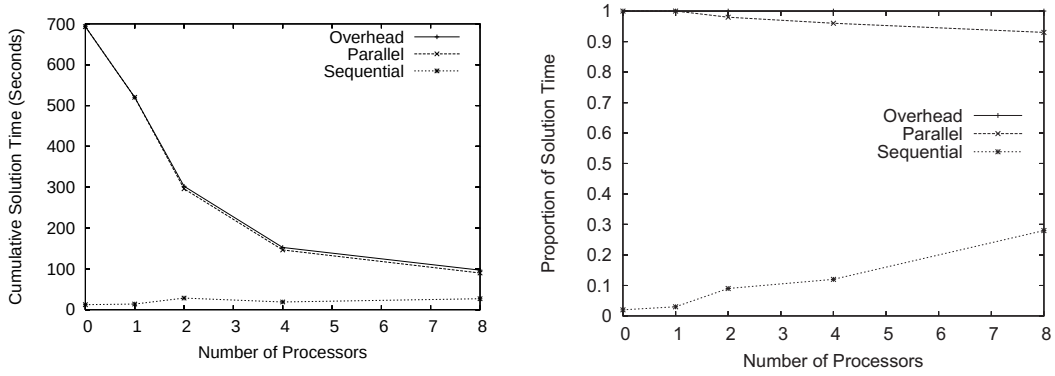

Figure A.3: Parallel, sequential and overhead components of the solution time for the OpenMP parallel SPH code on the Origin 2000.

for what could be achieved with little coding effort using the shared memory model and a specialised computer.

A.0.12 Message Passing Interface (MPI)

Table A.2 shows the run times for the MPI parallel SPH code on the Origin 2000 as well as the sequential performance on an R10000 processor. Significant overheads are evident in the data, even when using a single processor.

Table A.2: Run times of the MPI parallel SPH code on the Origin 2000 (in seconds).

N	Sequential	Number of processors			
		1	2	4	8
6250	6.58	8.99	6.21	6.70	6.54
12500	23.26	25.76	18.13	15.21	13.43
25000	78.37	85.30	54.29	39.61	27.39
50000	346.68	353.00	211.91	130.09	93.13
100000	1352.76	1386.16	816.40	469.47	321.62
200000	4805.65	5002.37	2683.61	1472.25	861.56

These results are plotted in Figure A.4, showing the overall trends present in the scaling of the solution time with simulation size. The multiple processor run times scale with the sequential times, with speedups of 1.79, 3.26 and 5.57 for 2, 4 and 8 processors respectively, for the largest simulation size.

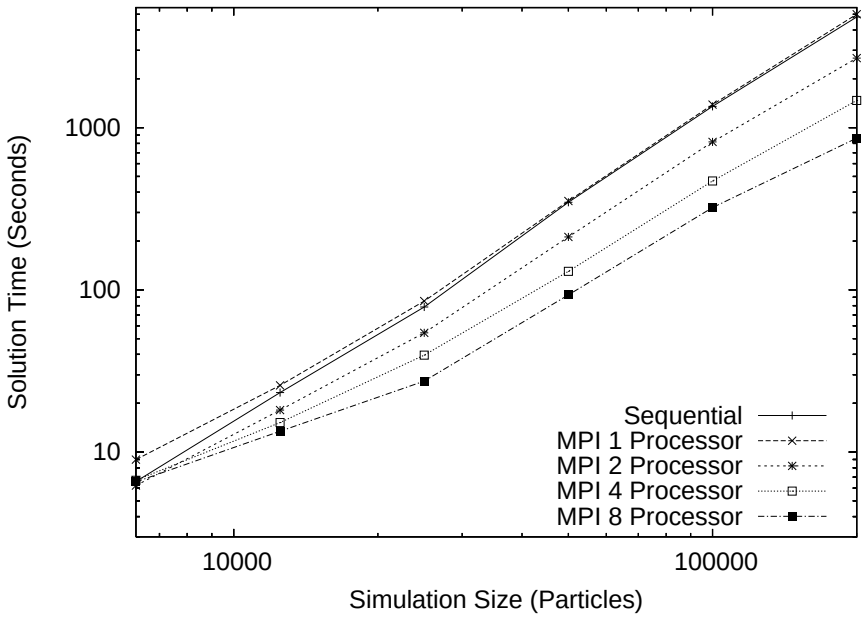

Figure A.4: Run times of the MPI parallel SPH code on the SGI Origin 2000.

Since communication must be performed separately to computation in the MPI version of the code, significant overheads may be expected. Even when running on a single processor, parallel overheads still account for between 4% to 10% of

the run time. This is expected since MPI collective communications require that a process transfers data to itself, incurring communication overheads even for a single processor run. In addition to this, overheads for synchronisation, as a part of the blocking communication, and process management would also be present. The parallel efficiency can be used to give a better picture of the effect of these overheads and is shown in Figure A.5.

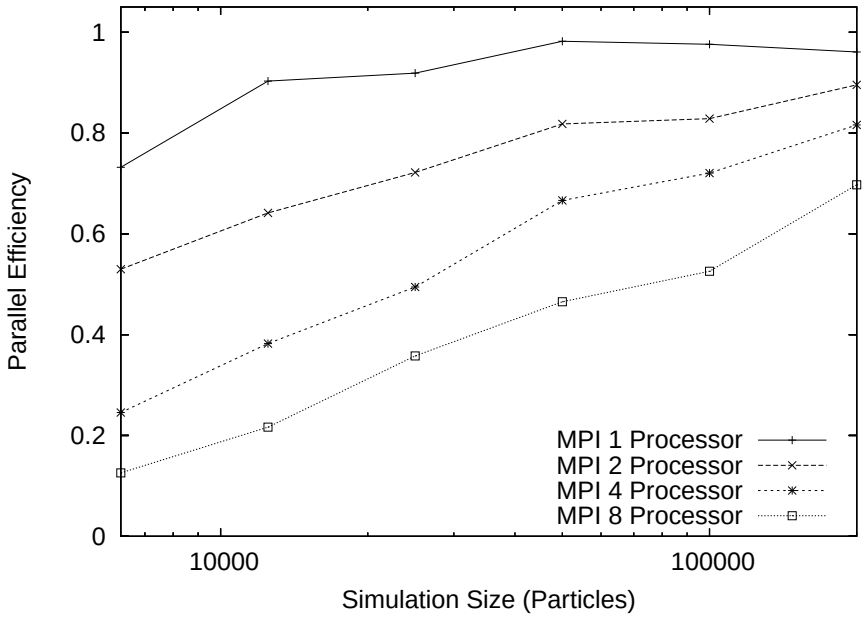

Figure A.5: Parallel efficiency of the MPI parallel SPH code on the SGI Origin 2000.

Message passing is an inefficient process with the small grained parallelism. Sending small messages through the interconnection network increases the effect of the system latency, as latency is independent of message size and, therefore, has a greater effect on the total send time for the message. The efficiency improves greatly as the number of particles is increased as would be expected with the use of message passing. The efficiencies reach 96%, 90%, 82% and 70% using 1, 2, 4 and 8 for the largest simulation size. This also shows the decrease in efficiency with increased number of processors. Although the simulations being run are quite small, the efficiencies are low and an investigation of the components making up the solution time may show where the inefficiencies are being caused.

On the Origin 2000, the `speedshop` libraries can be used to identify the contribution of parts of a program to the run times. The *parallel* computation, *sequential* computation and parallel *overhead* sections were grouped. The parallel overhead was identified by calls to the MPT libraries. Again, the code was compiled to optimisation level `O2` for these results to prevent inlining of functions. Figure A.6 shows

the contribution of the three components to the run time on the left, and their relative proportions on the right.

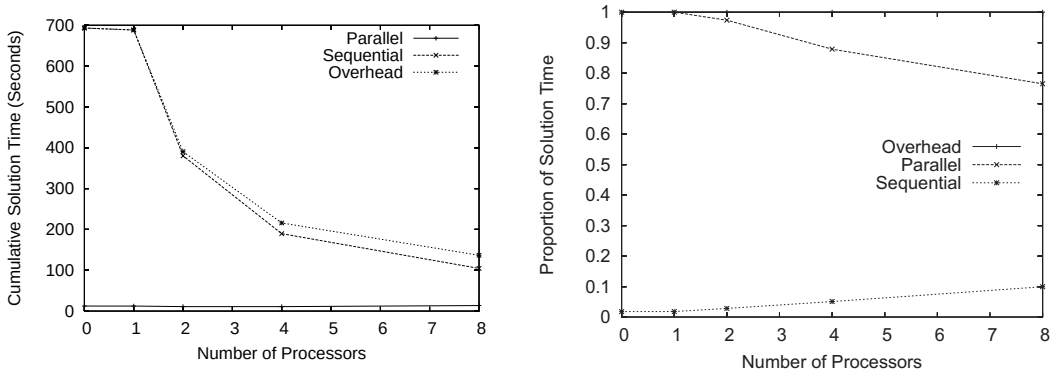

Figure A.6: Parallel, sequential and overhead components of the solution time for the MPI parallel SPH code on the Origin 2000.

The results obtained from `speedshop` were confirmed by bracketing sections of the code by calls to the MPI command `MPI_Wtime()` which can be used to measure elapsed time between lines in a code. These graphs show how the run time of parallel sections of the code is reduced in proportion to the number of processors, how the sequential run time stays constant and the parallel overhead increases with the number of processors. Since communication cannot be performed concurrently with computation with MPI, overheads make up a significant proportion of the run time, particularly for large numbers of processors; this can be seen in Figure A.6.

As well as the extra work incurred through parallelisation and the effect of non-parallelisable regions, another aspect of parallel execution is the effect of the parallelism on the action of the individual processors.

The time spent in sequential regions of the code and the time spent in parallel regions of the code, when normalised by the number of processes, should remain constant as the number of processes is increased. Figure A.7 shows that these times actually increase. The time taken in the sequential regions decreases below that of the sequential code for two and four processors before increasing back above the sequential level for eight processors. The normalised parallel times are are relatively constant, but are all above that for the sequential code. The reasons for this comes from a number of contributing factors and understanding their causes can enable improvements to the performance of the parallel code.

The Origin 2000 command, `perfex`, can be used to study the code behaviour in detail. A number of parameters associated with the calculation process can be counted and displayed. Parameters of interest to us in particular are those associated

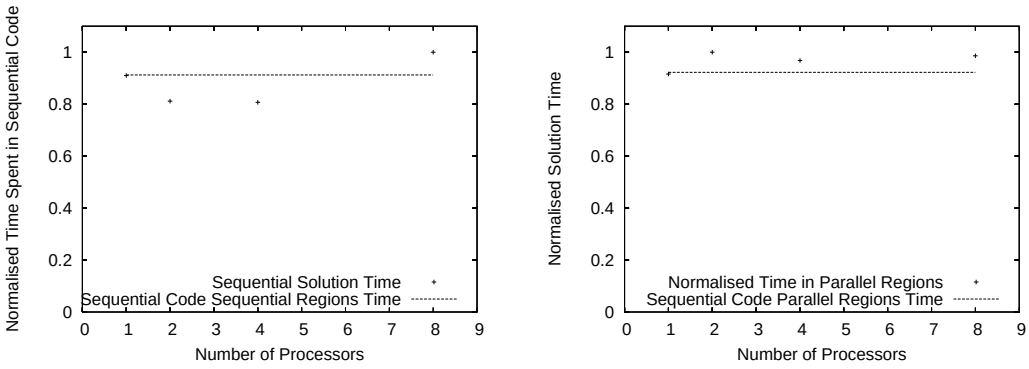

Figure A.7: Sequential and normalised parallel solution time, for a fixed simulation size.

with the parallel execution. For a multiple process program, the results returned by perfex are a per-process average as the `perfex` measures the aggregates of the counts returned for all of the threads.

Figure A.8 shows the effectiveness of cache utilisation when running in parallel. The left side of the figure shows the `data cache hit rate` which is the fraction of data accesses that are satisfied from a cache line already resident in the either the primary or secondary data caches. These two caches are shown on the plot as plus and times signs respectively. The plot shows that the primary data cache is unaffected by running the code with multiple processes. The primary data cache hit rate is over 97% for all jobs showing that it is being utilised very efficiently. The performance of the secondary data cache for a sequential job is 95.7% and 93.5% for a single processor MPI job. Running the code over multiple processors degrades the performance of the cache and the hit rate reduces to 90.0%, 88.9% and 86.1% as the number of processes is increased to 2, 4 and 8 processors.

The right side of Figure A.8 shows the `data cache line reuse` and much the same trend is evident. Data cache line reuse measures the number of times that, on average, a data cache line will be used once it is moved into the cache. When running in parallel, the number of primary data cache line reuses is above the sequential level of 35 uses. For the secondary cache, the number of line reuses drops significantly. For a sequential job, the secondary data cache line reuse is 22, dropping sharply to 14, 10, 8 and 6 when running in MPI on 1, 2, 4 and 8 processors respectively. Presently, it is not known why the secondary data cache is affected by MPI parallelism and the primary data cache is not.

Due to the message passing parallelism implemented in MPI, each process should be working on entirely different sets of data and therefore there should not be any affect on cache performance; however, the Origin 2000 is a shared memory computer and, although we are attempting to run in a distributed memory mode, cache

coherency issues may still be affecting performance.

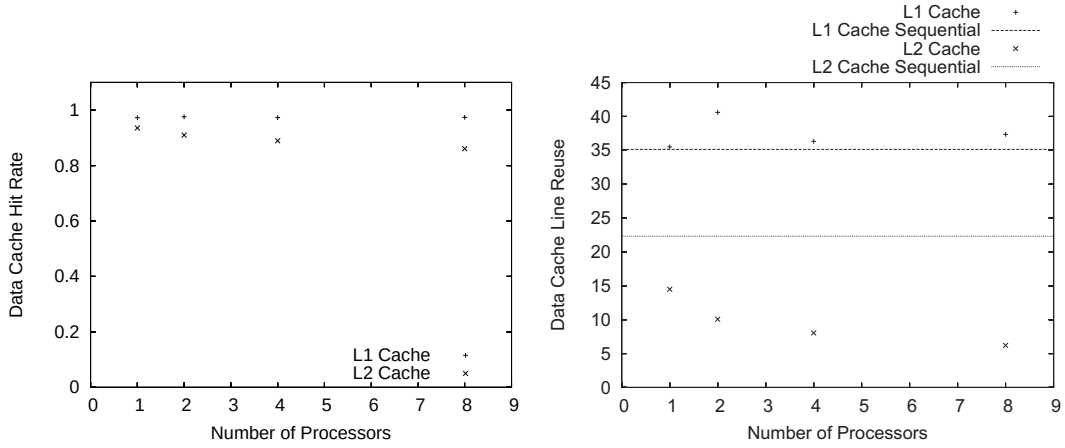

Figure A.8: Effect of parallelisation on cache utilisation.

As the number of processors is increased, so will the amount of data that needs to be moved through the interconnection network. The amount of data moved is proportional to the number of processors, as with domain decomposition of n particles over p processes, p segments of n/p particles, must be transferred p times. As a result of this, the bandwidth used per process should remain constant. The left side of Figure A.9 shows the sum of the typical costs of all memory accesses during the execution of the program as a fraction of the total run time. Memory access time includes that for graduated loads and stores, primary and secondary data cache misses and TLB misses. The fraction of the total run time spent accessing memory is around 22% for single process jobs and increases as the number of processors is increased to almost 30% for eight processors. This represents a significant proportion of the run time and explains a significant proportion of the decrease in parallel efficiency for a given simulation size as the number of processors is increased.

The right side of Figure A.9 shows the bandwidth used, per process, for both data moved to main memory and for data moved between the primary and secondary caches. The bandwidth used, per process, increased with the number of processors used, for both cases. This means that more data was being moved both between the caches and between the caches and the main memory. Being measured as the amount per process, this is above the amount that the data transfer between processes will increase by normally, as was discussed previously. This increase is possibly due to the increased amount of data that is transferred in order to maintain cache coherency as each of the caches are updated, even though this is not required for the MPI programs.

Figure A.10 shows two measures of how much each process of a multiple process

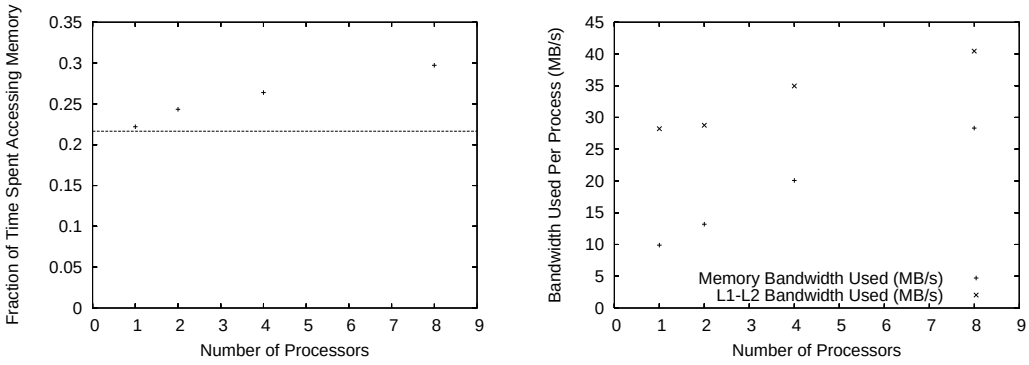

Figure A.9: Effect of parallelisation on memory access.

job affects one another. The left side of Figure A.10 shows the number of external interventions, which is a measure of the amount to which the processes are interfering with one anothers caches. The values for parallel jobs are given as points. The number of these will be relatively high if processors are constantly trying to write to cache lines stored on other processes.

In the case of the SPH code that we are analysing, this would be caused by a process updating the data for a particle which also exists as a copy in a cache line that another process is accessing. This number is negligibly small for the sequential and single process MPI jobs, and increases rapidly as the number of processors is increased, reaching over 16 million for 8 processors. This number is relatively small compared to the number of graduated stores and this is likely to have little impact on overall performance; however, as the number of processors is increased, issues relating to interference between processes will become more important.

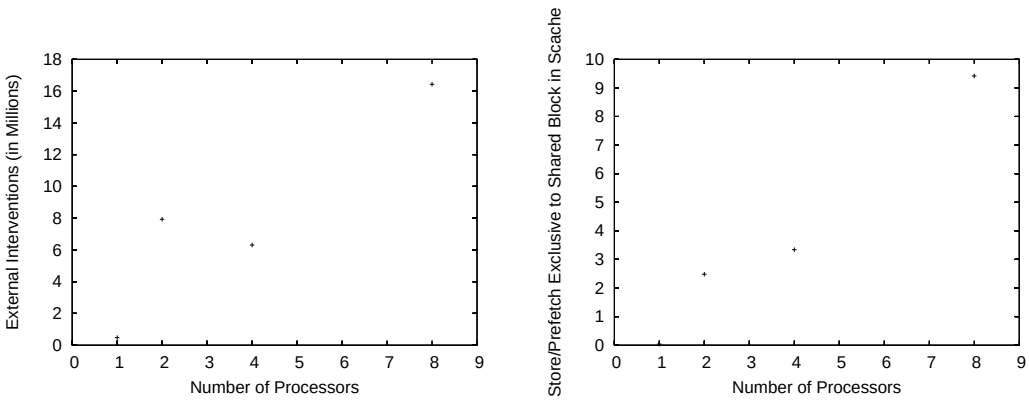

Figure A.10: Effect of processors on one another.

Pentium III Based Beowulf Cluster

As with the Origin 2000, the performance of the MPI parallel version of the SPH code is examined on a Beowulf cluster. In general, only one processor of the two was utilised at any time. In order to access unused processors, these second processors were used on machines that did not have more than one job running on them. This would work if the operating system would assign any new MPI processes on the second processor only, but this was not the case as the number of processors was increased. Trials were repeated on different nodes of the system and produced consistent results. Table A.3 shows the run times for the SPH code on the Beowulf cluster.

Table A.3: Run times of the MPI parallel SPH code on the Beowulf cluster (in seconds)

N	Sequential	Number of processors			
		1	2	4	8
6250	9.12	9.63	8.02	91.50	
12500	32.47	32.94	30.07	184.62	
25000	107.76	109.00	76.81	238.30	
50000	451.69	452.26	275.59	400.08	878.12
100000	1745.01	1745.38	997.25	1094.47	

Simulations with 200,000 particles were not run, neither were solutions using 8 processors for other that 50,000 particles. These results are also presented in Figure A.11. Overheads for the single processor runs were negligible. The 2 processor run times scaled relatively closely with the sequential code, but improves in performance as the simulation size is increased. The performance of the 4 processor run was very poor for the smaller simulations, but increased dramatically as the simulation size was increased

Figure A.12 shows the parallel efficiency for the MPI parallel runs on the cluster. The efficiency for the 2 processor simulation is acceptable and increased steadily from 57% to 87% across the range of simulation sizes tested. The efficiency for the 4 processor simulation is very low, increasing steadily from only 2% to 40%, across the range of simulation sizes. A single point was recorded for the 8 processor solution time since the parallel program did not appear to be running properly. The operating system function `top` showed that, for runs with four or eight processors, the utilisation of the second processor, which we aim at using, is below 5% while the job is running. This suggests that the problem lies with the operating system (Redhat Linux 7.0) not properly allocating the processes across the nodes on the system.

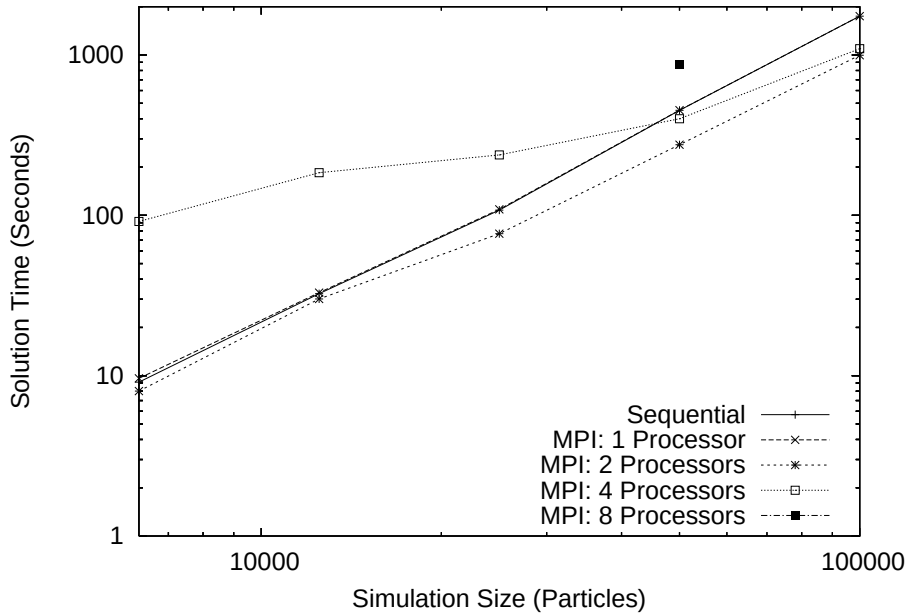

Figure A.11: Run times of the MPI parallel SPH code on the Beowulf cluster.

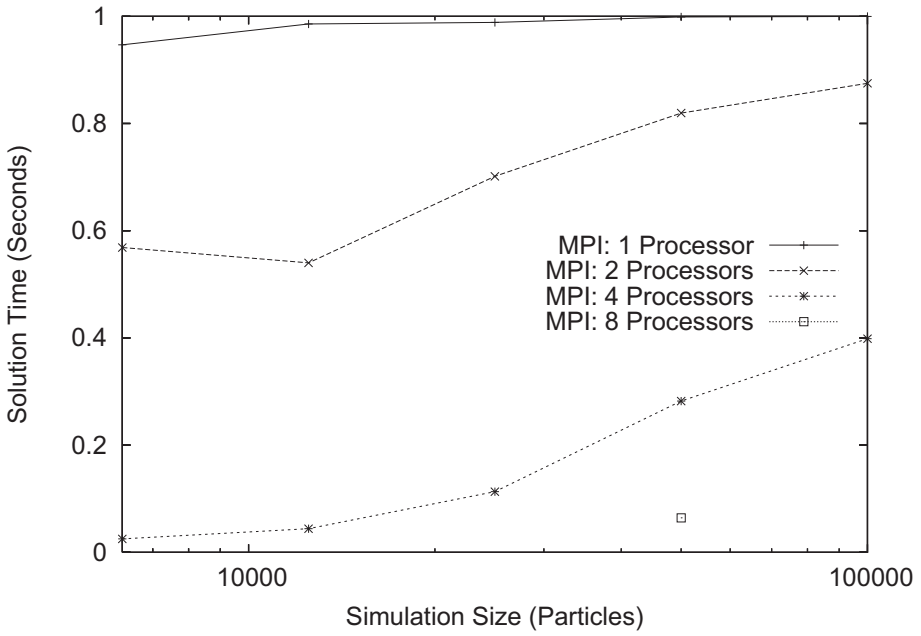

Figure A.12: Parallel efficiency of the MPI parallel SPH code on the SGI Origin 2000.

A.0.13 Bulk Synchronous Parallel (BSP)

Table A.4 shows the run times for the BSP parallel SPH code on the Origin 2000 as well as the sequential performance on an R10000 processor. It can be seen in this table how the run time is reduced by running the code in parallel, and how the run time scales with simulation size for the sequential and the parallel runs.

Table A.4: Run times of the BSP parallel SPH code on the Origin 2000 (in seconds).

N	Sequential	Number of processors			
		1	2	4	8
6250	6.58	7.41	5.45	6.03	7.02
12500	23.26	23.65	16.60	14.16	15.37
25000	78.37	84.78	51.19	39.46	36.47
50000	346.68	339.30	205.88	132.13	102.60
100000	1352.76	1335.47	758.78	501.29	301.17
200000	4805.65	4914.91	2809.43	1611.27	1030.56

These results are plotted in Figure A.13, showing the overall trends in the scaling of the solution time. The solution time using the BSP code with a single processor is close to that of the sequential code. This is expected since, as well as other overheads not being present, the BSP communication routine used does not require that a process communicate with itself.

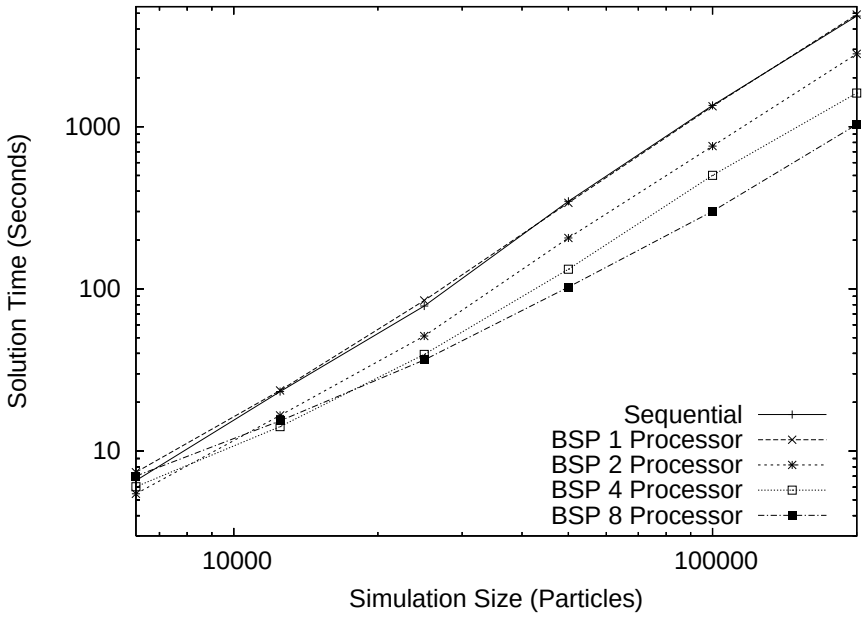

Figure A.13: Run times of the BSP parallel SPH code on the SGI Origin 2000.

Although the performance for the multi-processor runs on the smaller simulation

was quite bad, the relative run times increased quickly as the simulation size was increased. The speedups for the largest simulation sizes were 1.71, 2.98 and 4.66 for 2, 4 and 8 processors respectively.

The efficiency of the BSP parallel code is shown in Figure A.14. This figure shows that the efficiency for the single processor run fluctuate, but is relatively close to one. The efficiency drops markedly as the number of processors is increased; however, the efficiency increases steadily with simulation size. For the largest simulation size, the efficiency reaches 98%, 86%, 75% and 58% for the 1, 2, 4 and 8 processor runs.

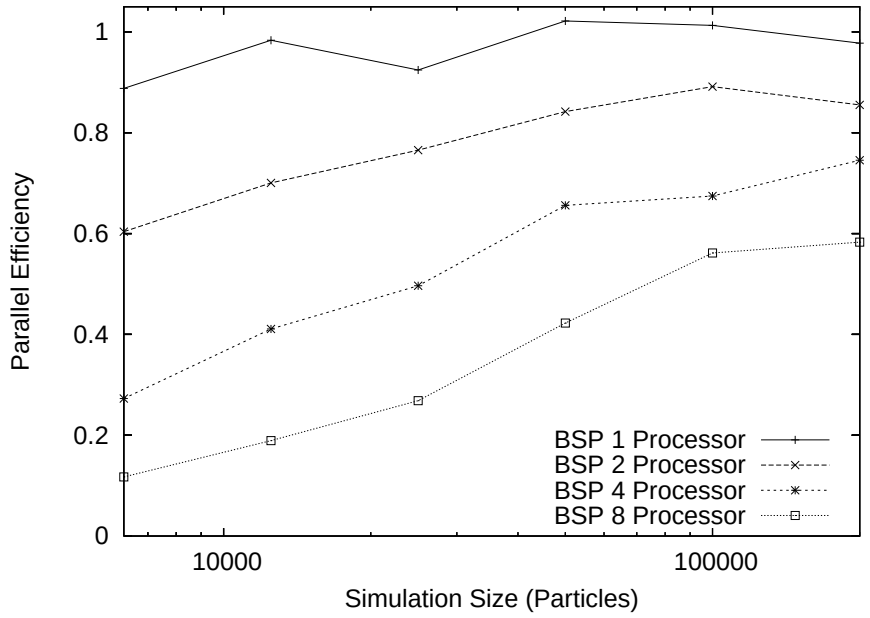

Figure A.14: Parallel efficiency of the BSP parallel SPH code on the SGI Origin 2000.

The lack of an efficient communication routine, with all variables being transferred to all processes at the end of each superstep, has probably meant that communication has made a significant contribution to the parallel overheads.

A.0.14 Comparing Performance

The parallel performance and efficiency of the SPH code using OpenMP, MPI and BSP on the SGI Origin 2000 can be compared directly. Table A.5 shows the run times for the three methods, using 4 processors, for the range of simulation sizes.

Table A.5: Comparison of parallel performance on the Origin 2000 with 4 processors for a range of simulation sizes (in seconds).

N	OpenMP	MPI	BSP
6250	3.04	6.70	6.03
12500	9.73	15.21	14.16
25000	27.30	39.61	39.46
50000	107.02	130.09	132.13
100000	384.93	469.47	501.29
200000	1453.44	1472.25	1611.27

These results are also plotted in Figure A.15. MPI and BSP, both being based on message passing, perform roughly the same as each other over the range of simulation sizes. They both perform badly for small simulations, since message passing is less efficient for small message sizes, such as those with small simulations. OpenMP, being based on the shared memory model, is not as susceptible to latency as message passing. Its performance is good for the whole range of simulation sizes used in the tests. The Origin 2000 was specially designed to take advantage of OpenMP code and good performance would be expected.

The performance of BSP is slightly better than for MPI, which is primarily due to the choice of collective communication routines, rather than the performance of the libraries themselves; however, the Message Passing Toolkit (MPT) on the Origin 2000 is specifically tuned for this architecture, so this would provide advantages over BSP, which was not as tuned.

Figure A.16 compares parallel efficiency between the methods, across the range of simulation sizes, using four processors. The much greater efficiency of OpenMP for the smaller simulations is evident.

As shown in Figure A.17, varying the number of processors for the two largest simulation sizes, of 100,000 and 200,000 particles shows the effect of processor dependent overheads; important amongst these are the increased overhead for synchronisation and communication for increased numbers of processors. These figures show the resulting decrease in parallel efficiency with increased number of processors, for a fixed simulation size. The efficiency of OpenMP, with the flattest curve, is the least affected by increasing the number of processors.

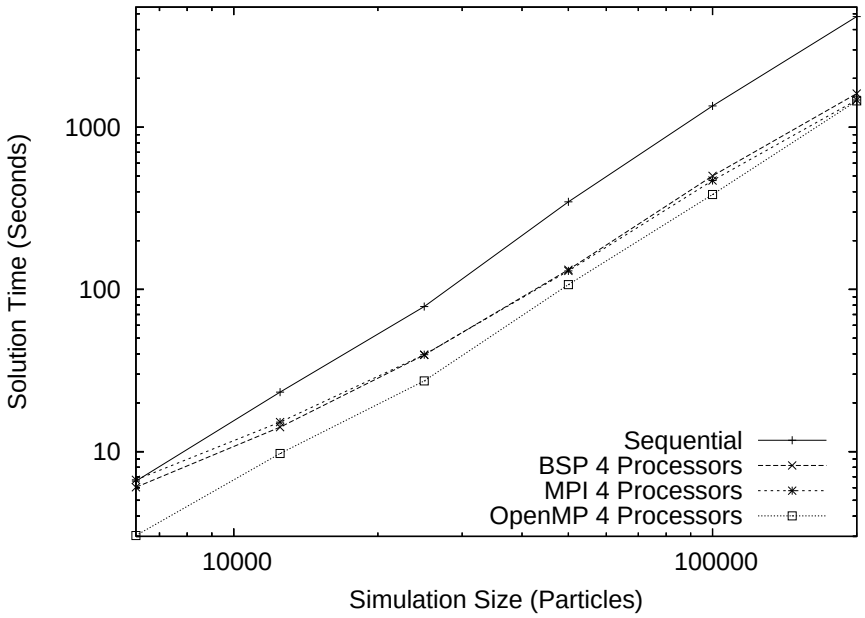

Figure A.15: Comparison of parallel performance on the Origin 2000 with 4 processors for a range of simulation sizes (in seconds).

Figure A.18 compares the run times (in seconds), and the parallel efficiency, using MPI on the Origin 2000 and the Beowulf cluster. As was discussed earlier, the performance of MPI on the Beowulf cluster could not be examined properly. Increasing the number of processors above two resulted in utilisation of the processors around 10% and so the performance results did not reflect the potential parallel performance; however, up to two processors the same general trend is evident as that on the Origin 2000, given the relative speeds of the systems.

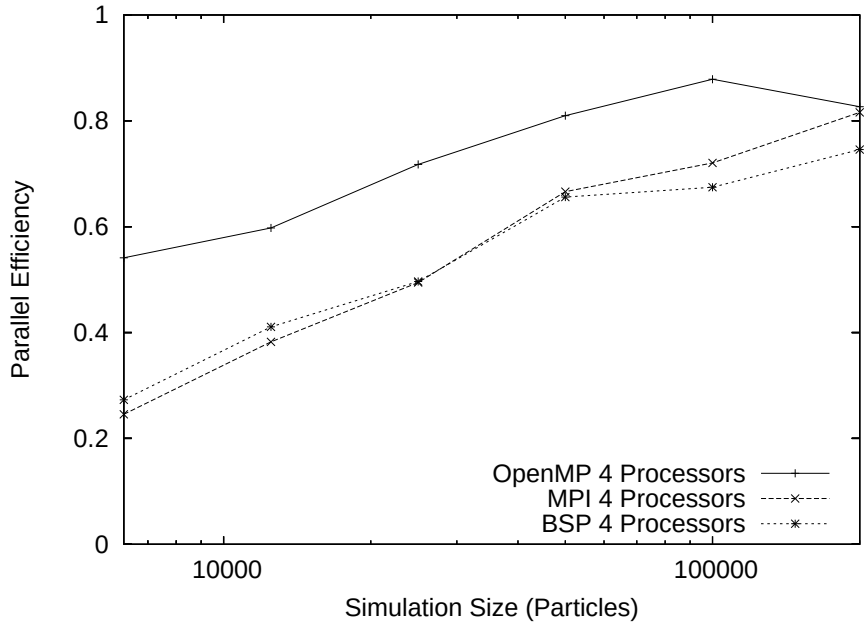

Figure A.16: Comparison of parallel efficiency on the Origin 2000 with 4 Processors for a range of simulation sizes.

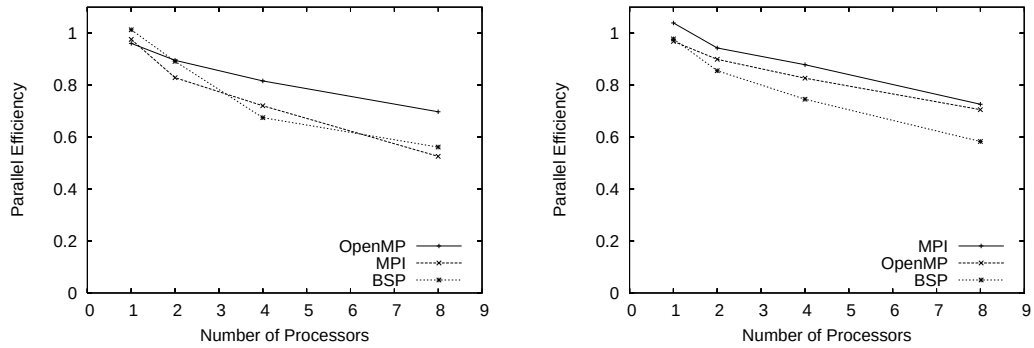

Figure A.17: Comparison of parallel efficiency on the Origin 2000 for simulation sizes of 100,000 particles (left) and 200,000 particles (right).

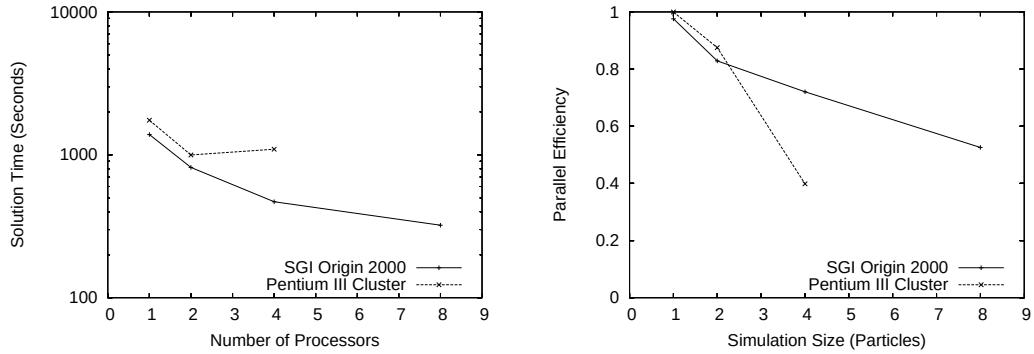

Figure A.18: Comparison of run times (left) and parallel efficiency (right) of MPI on the Origin 2000 and the Beowulf cluster.

APPENDIX **B**

MB_CNS Scriptit Files

```
# drummond_tunnel_m4nozzle_80.sit
# Complete Drummond Tunnel Simulation
# Mach 4 nozzle, new (1998) driver
# N2 driving N2 and He driving N2 cases
# Version 7.4 (06/02/03)

BEGIN_GEOMETRY

  # Driver Section Nodes
  NODE d0  -3.02100 0.00000
  NODE d1  -3.02100 0.03110
  NODE d2  -3.05701 0.00635
  NODE d3  -3.40600 0.00635
  NODE d4  -3.40600 0.02950
  NODE d5  -3.79100 0.00635
  NODE d6  -3.79100 0.02950
  NODE d7  -3.03900 0.02950
  NODE d10 -3.01160 0.03110
  NODE d14 -3.81850 0.02480
  NODE d15 -3.79600 0.03700
  NODE d16 -4.02100 0.02480
  NODE d17 -4.02100 0.03700

  # Shock Tube Nodes
  NODE s2  -2.60130 0.00000
  NODE s3  -2.60130 0.03110
  NODE s4  -2.25160 0.00000
  NODE s5  -2.25160 0.03110
  NODE s6  -1.90190 0.00000
  NODE s7  -1.90190 0.03110
  NODE s8  -1.55210 0.00000
  NODE s9  -1.55210 0.03110
  NODE s10 -1.20240 0.00000
  NODE s11 -1.20240 0.03110
  NODE s12 -0.85270 0.00000
  NODE s13 -0.85270 0.03110

  NODE s14 -0.60830 0.00000
  NODE s15 -0.60830 0.03110

  NODE s16 -0.38390 0.00000
  NODE s17 -0.38390 0.03110
```

```
NODE s18 -0.22500 0.00000
NODE s19 -0.22500 0.03110

NODE s20 -0.08620 0.00000
NODE s21 -0.08620 0.03110

# Nodes for the Mach 4.0 Nozzle
NODE e1   0.04000 0.00000
NODE e2   0.04000 0.01000
NODE e3   0.05305 0.02000
NODE e4   0.05305 0.03110
NODE f    0.06805 0.03110
NODE g    0.06805 0.01110
NODE h    0.08305 0.01110
NODE h0   0.08305 0.00000
NODE i    0.10005 0.01110
NODE j    0.10505 0.01110
NODE k    0.11005 0.01180
NODE k0   0.11005 0.00000

# Nozzle end and dumptank
NODE m    0.27535 0.03500
NODE n0   0.27535 0.00000
NODE n1   0.27535 0.03790
NODE n2   0.26535 0.03970
NODE n3   0.16535 0.03970
NODE n4   0.16535 0.15200
NODE n5   0.27535 0.15200
NODE t0   0.38535 0.00000
NODE t1   0.38535 0.03500
NODE t2   0.38535 0.03790
NODE t3   0.38535 0.15200
NODE t4   0.66535 0.00000
NODE t5   0.66535 0.03500
NODE t6   0.66535 0.03790
NODE t7   0.66535 0.15200

LINE d17d15 d17 d15
LINE d15d6  d15 d6
LINE d16d14 d16 d14
LINE d14d5  d14 d5
LINE d16d17 d16 d17
LINE d5d6   d5  d6
LINE d0d1 d0 d1
LINE d3d4 d3 d4
LINE d5d3 d5 d3
LINE d3d2 d3 d2
LINE d2d0 d2 d0
LINE d6d4 d6 d4
LINE d4d7 d4 d7
LINE d7d1 d7 d1
LINE d0s2 d0 s2
LINE d1d10 d1 d10
LINE d10s3 d10 s3

LINE s2s3 s2 s3
```

```
LINE s2s4 s2 s4
LINE s3s5 s3 s5
LINE s4s5 s4 s5
LINE s4s6 s4 s6
LINE s5s7 s5 s7
LINE s6s7 s6 s7
LINE s6s8 s6 s8
LINE s7s9 s7 s9
LINE s8s9 s8 s9
LINE s8s10  s8  s10
LINE s9s11  s9  s11
LINE s10s11 s10 s11
LINE s10s12 s10 s12
LINE s11s13 s11 s13
LINE s12s13 s12 s13
LINE s12s14 s12 s14
LINE s13s15 s13 s15
LINE s14s15 s14 s15
LINE s14s16 s14 s16
LINE s15s17 s15 s17
LINE s16s17 s16 s17
LINE s16s18 s16 s18
LINE s17s19 s17 s19
LINE s18s19 s18 s19
LINE s18s20 s18 s20
LINE s19s21 s19 s21
LINE s20s21 s20 s21
LINE s20e1 s20 e1
LINE s21e4 s21 e4
BEZIER e1e4 e1 e2 e3 e4

BEZIER e4h e4 f g h
LINE e1h0 e1 h0
LINE h0h  h0 h
LINE hi   h  i
BEZIER ik i j j k
LINE km   k  m
LINE h0k0 h0 k0
LINE k0k  k0 k
LINE k0n0 k0 n0
LINE n0m  n0 m
LINE mn1  m  n1
LINE n3n2 n3 n2
LINE n2n1 n2 n1
LINE n3n4 n3 n4
LINE n4n5 n4 n5
LINE n1n5 n1 n5

LINE mt1  m  t1
LINE n0t0 n0 t0
LINE n1t2 n1 t2
LINE n5t3 n5 t3
LINE t0t1 t0 t1
LINE t1t2 t1 t2
LINE t2t3 t2 t3
LINE t0t4 t0 t4
```

```
LINE t1t5 t1 t5
LINE t2t6 t2 t6
LINE t3t7 t3 t7
LINE t4t5 t4 t5
LINE t5t6 t5 t6
LINE t6t7 t6 t7

POLYLINE north0 2 + d17d15 + d15d6
POLYLINE south0 2 + d16d14 + d14d5
POLYLINE east0  1 + d5d6
POLYLINE west0  1 + d16d17

POLYLINE north1 1 + d6d4
POLYLINE south1 1 + d5d3
POLYLINE east1  1 + d3d4

POLYLINE north2  2 + d4d7 + d7d1
POLYLINE south2  2 + d3d2  + d2d0
POLYLINE east2   1 + d0d1

POLYLINE north3  1 + d1d10 + d10s3
POLYLINE south3  1 + d0s2
POLYLINE east3   1 + s2s3

POLYLINE north4  1 + s3s5
POLYLINE south4  1 + s2s4
POLYLINE east4   1 + s4s5

POLYLINE north5  1 + s5s7
POLYLINE south5  1 + s4s6
POLYLINE east5   1 + s6s7

POLYLINE north6  1 + s7s9
POLYLINE south6  1 + s6s8
POLYLINE east6   1 + s8s9

POLYLINE north7  1 + s9s11
POLYLINE south7  1 + s8s10
POLYLINE east7   1 + s10s11

POLYLINE north8  1 + s11s13
POLYLINE south8  1 + s10s12
POLYLINE east8   1 + s12s13

POLYLINE north9  1 + s13s15
POLYLINE south9  1 + s12s14
POLYLINE east9   1 + s14s15

POLYLINE north10 1 + s15s17
POLYLINE south10 1 + s14s16
POLYLINE east10  1 + s16s17

POLYLINE north11 1 + s17s19
POLYLINE south11 1 + s16s18
POLYLINE east11  1 + s18s19
```

```
  POLYLINE north12 1 + s19s21
  POLYLINE south12 1 + s18s20
  POLYLINE east12  1 + s20s21

  POLYLINE north13 1 + s21e4
  POLYLINE south13 1 + s20e1
  POLYLINE east13  1 + e1e4

  POLYLINE north14 1 + e4h
  POLYLINE south14 1 + e1h0
  POLYLINE east14  1 + h0h

  POLYLINE north15 2 + hi + ik
  POLYLINE south15 1 + h0k0
  POLYLINE east15  1 + k0k

  POLYLINE north16 1 + km
  POLYLINE south16 1 + k0n0
  POLYLINE east16  1 + n0m

  POLYLINE north17 1 + mt1
  POLYLINE south17 1 + n0t0
  POLYLINE east17  1 + t0t1

  POLYLINE north18 1 + n1t2
  POLYLINE east18  1 + t1t2
  POLYLINE west18  1 + mn1

  POLYLINE north19 1 + n5t3
  POLYLINE east19  1 + t2t3
  POLYLINE west19  1 + n1n5

  POLYLINE north20 1 + n4n5
  POLYLINE south20 2 + n3n2 + n2n1
  POLYLINE west20  1 + n3n4

  POLYLINE north21 1 + t1t5
  POLYLINE south21 1 + t0t4
  POLYLINE east21  1 + t4t5

  POLYLINE north22 1 + t2t6
  POLYLINE east22  1 + t5t6

  POLYLINE north23 1 + t3t7
  POLYLINE east23  1 + t6t7

END_GEOMETRY

BEGIN_FLOW

  # --- [aug98] N2 driving N2 ---
  #GAS_TYPE PERF_N2
  GAS_TYPE LUTN
  GAS_STATE driver   3.25e6 0.0 0.0 310.00 1.0 0.0 0.0 0.0 0.0
  GAS_STATE driven   3.00e4 0.0 0.0 296.00 0.0 1.0 0.0 0.0 0.0
  GAS_STATE dumptank 4.00e2 0.0 0.0 296.00 0.0 0.0 1.0 0.0 0.0
```

```
# --- [feb00] He driving N2 ---
#GAS_TYPE PERF_HE_N2
#GAS_STATE driver   5.60e6 0.0 0.0 305.00 1.0 0.0 0.0 0.0 0.0
#GAS_STATE driven   6.14e4 0.0 0.0 296.00 0.0 1.0 0.0 0.0 0.0
#GAS_STATE dumptank 4.00e2 0.0 0.0 296.00 0.0 0.0 1.0 0.0 0.0
# --- [end] ---

DISCRETISE north0  140 0 0 0.0
DISCRETISE south0  140 1 0 1.6
DISCRETISE east0   80  0 1 1.1
DISCRETISE west0   80  0 1 1.1

DISCRETISE north1  220 0 0 0.0
DISCRETISE south1  220 0 0 0.0
DISCRETISE east1   80  0 1 1.1

DISCRETISE north2  220 0 0 0.0
DISCRETISE south2  220 0 0 0.0
DISCRETISE east2   80  0 1 1.1

DISCRETISE north3  240 0 0 0.0
DISCRETISE south3  240 0 0 0.0
DISCRETISE east3   80  0 1 1.1

DISCRETISE north4  240 0 0 0.0
DISCRETISE south4  240 0 0 0.0
DISCRETISE east4   80  0 1 1.1

DISCRETISE north5  240 0 0 0.0
DISCRETISE south5  240 0 0 0.0
DISCRETISE east5   80  0 1 1.1

DISCRETISE north6  240 0 0 0.0
DISCRETISE south6  240 0 0 0.0
DISCRETISE east6   80  0 1 1.1

DISCRETISE north7  240 0 0 0.0
DISCRETISE south7  240 0 0 0.0
DISCRETISE east7   80  0 1 1.1

DISCRETISE north8  240 0 0 0.0
DISCRETISE south8  240 0 0 0.0
DISCRETISE east8   80  0 1 1.1

DISCRETISE north9  240 0 0 0.0
DISCRETISE south9  240 0 0 0.0
DISCRETISE east9   80  0 1 1.1

DISCRETISE north10 240 0 1 1.3
DISCRETISE south10 240 0 1 1.3
DISCRETISE east10  80  0 1 1.1

DISCRETISE north11 240 0 0 0.0
DISCRETISE south11 240 0 0 0.0
DISCRETISE east11  80  0 1 1.1
```

```
DISCRETISE north12 240 0 0 0.0
DISCRETISE south12 240 0 0 0.0
DISCRETISE east12  80  0 1 1.1

DISCRETISE north13 240 0 0 0.0
DISCRETISE south13 240 0 0 0.0
DISCRETISE east13  80  0 1 1.1

DISCRETISE north14 120 0 1 1.2
DISCRETISE south14 120 0 1 1.2
DISCRETISE east14  80  0 1 1.1

DISCRETISE north15 120 0 0 0.0
DISCRETISE south15 120 0 0 0.0
DISCRETISE east15  80  0 1 1.1

DISCRETISE north16 200 1 0 1.2
DISCRETISE south16 200 1 0 1.2
DISCRETISE east16  80  0 1 1.1

DISCRETISE north17 100 0 0 0.0
DISCRETISE south17 100 0 0 0.0
DISCRETISE east17  80  0 0 0.0

DISCRETISE north18 100 0 0 0.0
DISCRETISE east18  10  0 0 0.0
DISCRETISE west18  10  0 0 0.0

DISCRETISE north19 100 0 0 0.0
DISCRETISE east19  60  1 0 1.2
DISCRETISE west19  60  1 0 1.1

DISCRETISE north20 60  0 1 1.2
DISCRETISE south20 60  0 1 1.2
DISCRETISE west20  60  1 0 1.1

DISCRETISE north21 80  1 0 1.2
DISCRETISE south21 80  1 0 1.2
DISCRETISE east21  80  0 0 0.0

DISCRETISE north22 80  1 0 1.2
DISCRETISE east22  10  0 0 0.0

DISCRETISE north23 80  1 0 1.2
DISCRETISE east23  60  1 0 1.1

BOUNDARY_SPEC north0  FIXED_T 296.0
BOUNDARY_SPEC north1  FIXED_T 296.0
BOUNDARY_SPEC north2  FIXED_T 296.0
BOUNDARY_SPEC north3  FIXED_T 296.0
BOUNDARY_SPEC north4  FIXED_T 296.0
BOUNDARY_SPEC north5  FIXED_T 296.0
BOUNDARY_SPEC north6  FIXED_T 296.0
BOUNDARY_SPEC north7  FIXED_T 296.0
BOUNDARY_SPEC north8  FIXED_T 296.0
BOUNDARY_SPEC north9  FIXED_T 296.0
```

```
BOUNDARY_SPEC north10 FIXED_T 296.0
BOUNDARY_SPEC north11 FIXED_T 296.0
BOUNDARY_SPEC north12 FIXED_T 296.0
BOUNDARY_SPEC north13 FIXED_T 296.0
BOUNDARY_SPEC north14 FIXED_T 296.0
BOUNDARY_SPEC north15 FIXED_T 296.0
BOUNDARY_SPEC north16 FIXED_T 296.0
BOUNDARY_SPEC north19 SUP_OUT
BOUNDARY_SPEC north20 SUP_OUT
BOUNDARY_SPEC north23 SUP_OUT

BLOCK driver1 + north0  + east0  + south0  + west0
BLOCK driver2 + north1  + east1  + south1  + east0
BLOCK driver3 + north2  + east2  + south2  + east1
BLOCK shock1  + north3  + east3  + south3  + east2
BLOCK shock2  + north4  + east4  + south4  + east3
BLOCK shock3  + north5  + east5  + south5  + east4
BLOCK shock4  + north6  + east6  + south6  + east5
BLOCK shock5  + north7  + east7  + south7  + east6
BLOCK shock6  + north8  + east8  + south8  + east7
BLOCK shock7  + north9  + east9  + south9  + east8
BLOCK shock8  + north10 + east10 + south10 + east9
BLOCK shock9  + north11 + east11 + south11 + east10
BLOCK shock10 + north12 + east12 + south12 + east11
BLOCK shock11 + north13 + east13 + south13 + east12
BLOCK nozzle1 + north14 + east14 + south14 + east13
BLOCK nozzle2 + north15 + east15 + south15 + east14
BLOCK nozzle3 + north16 + east16 + south16 + east15
BLOCK test1   + north17 + east17 + south17 + east16
BLOCK test2   + north18 + east18 + north17 + west18
BLOCK test3   + north19 + east19 + north18 + west19
BLOCK test4   + north20 + west19 + south20 + west20
BLOCK test5   + north21 + east21 + south21 + east17
BLOCK test6   + north22 + east22 + north21 + east18
BLOCK test7   + north23 + east23 + north22 + east19

CONNECT_BLOCKS driver1 east  driver2 west
CONNECT_BLOCKS driver2 east  driver3 west
CONNECT_BLOCKS driver3 east  shock1  west
CONNECT_BLOCKS shock1  east  shock2  west
CONNECT_BLOCKS shock2  east  shock3  west
CONNECT_BLOCKS shock3  east  shock4  west
CONNECT_BLOCKS shock4  east  shock5  west
CONNECT_BLOCKS shock5  east  shock6  west
CONNECT_BLOCKS shock6  east  shock7  west
CONNECT_BLOCKS shock7  east  shock8  west
CONNECT_BLOCKS shock8  east  shock9  west
CONNECT_BLOCKS shock9  east  shock10 west
CONNECT_BLOCKS shock10 east  shock11 west
CONNECT_BLOCKS shock11 east  nozzle1 west
CONNECT_BLOCKS nozzle1 east  nozzle2 west
CONNECT_BLOCKS nozzle2 east  nozzle3 west
CONNECT_BLOCKS nozzle3 east  test1   west
CONNECT_BLOCKS test1   north test2   south
CONNECT_BLOCKS test2   north test3   south
CONNECT_BLOCKS test3   west  test4   east
```

```
  CONNECT_BLOCKS test1   east  test5   west
  CONNECT_BLOCKS test2   east  test6   west
  CONNECT_BLOCKS test3   east  test7   west
  CONNECT_BLOCKS test5   north test6   south
  CONNECT_BLOCKS test6   north test7   south

  FILL_BLOCK driver1 driver
  FILL_BLOCK driver2 driver
  FILL_BLOCK driver3 driver
  FILL_BLOCK shock1  driven
  FILL_BLOCK shock2  driven
  FILL_BLOCK shock3  driven
  FILL_BLOCK shock4  driven
  FILL_BLOCK shock5  driven
  FILL_BLOCK shock6  driven
  FILL_BLOCK shock7  driven
  FILL_BLOCK shock8  driven
  FILL_BLOCK shock9  driven
  FILL_BLOCK shock10 driven
  FILL_BLOCK shock11 driven
  FILL_BLOCK nozzle1 driven
  FILL_BLOCK nozzle2 driven
  FILL_BLOCK nozzle3 dumptank
  FILL_BLOCK test1   dumptank
  FILL_BLOCK test2   dumptank
  FILL_BLOCK test3   dumptank
  FILL_BLOCK test4   dumptank
  FILL_BLOCK test5   dumptank
  FILL_BLOCK test6   dumptank
  FILL_BLOCK test7   dumptank

END_FLOW

BEGIN_CONTROL

  TITLE Entire Drummond Tunnel Simulation (Diaphragm Rupture, Mach 4 Nozzle)
  # progressive blocks can be activated in mb_special_init.c
  CASE_ID 100

  AXISYMMETRIC
  VISCOUS
  TURBULENT shock1
  TURBULENT shock2
  TURBULENT shock3
  TURBULENT shock4
  TURBULENT shock5
  TURBULENT shock6
  TURBULENT shock7
  TURBULENT shock8
  TURBULENT shock9
  TURBULENT shock10
  TURBULENT shock11
  TURBULENT nozzle1
  TURBULENT nozzle2
  TURBULENT nozzle3
  FLUX_CALC  adaptive
```

```
  MAX_TIME   1.0e-2
  MAX_STEP   600000
  TIME_STEP  0.2e-8
  DT_PLOT    1.0e-4
  DT_HISTORY 1.0e-6
  HISTORY_CELL driver2 220 80
  HISTORY_CELL shock2  106 80
  HISTORY_CELL shock9  150 80
  HISTORY_CELL shock11 32  80
  HISTORY_CELL nozzle1 1   1
  # --- [aug98] N2 driving N2 ---
  HISTORY_CELL test1 14  1
  HISTORY_CELL test1 14  22
  HISTORY_CELL test1 14  30
  HISTORY_CELL test1 14  52
  # --- [feb00] He driving N2 ---
  #HISTORY_CELL test1   1  1
  #HISTORY_CELL test1   1  22
  #HISTORY_CELL test1   1  30
  #HISTORY_CELL test1   1  52
  # --- [end] ---

END_CONTROL

BEZIER_FILE drummond_tunnel.bez
PARAM_FILE drummond_tunnel.p
BUILD

EXIT
```

```
# drummond_tunnel_blanked_80.sit
# Complete Drummond Tunnel Simulation
# Blanked end, new (1998) driver,
# N2 driving N2 case
# Version 7.4 (06/02/03)

BEGIN_GEOMETRY

 # Driver Section Nodes
  NODE d0  -3.02100 0.00000
  NODE d1  -3.02100 0.03110
  NODE d2  -3.05701 0.00635
  NODE d3  -3.40600 0.00635
  NODE d4  -3.40600 0.02950
  NODE d5  -3.79100 0.00635
  NODE d6  -3.79100 0.02950
  NODE d7  -3.03900 0.02950
  NODE d10 -3.01160 0.03110
  NODE d14 -3.81850 0.02480
  NODE d15 -3.79600 0.03700
  NODE d16 -4.02100 0.02480
  NODE d17 -4.02100 0.03700

  # Shock Tube Nodes
  NODE s2  -2.60130 0.00000
  NODE s3  -2.60130 0.03110
  NODE s4  -2.25160 0.00000
  NODE s5  -2.25160 0.03110
  NODE s6  -1.90190 0.00000
  NODE s7  -1.90190 0.03110
  NODE s8  -1.55210 0.00000
  NODE s9  -1.55210 0.03110
  NODE s10 -1.20240 0.00000
  NODE s11 -1.20240 0.03110
  NODE s12 -0.85270 0.00000
  NODE s13 -0.85270 0.03110

  NODE s14 -0.60830 0.00000
  NODE s15 -0.60830 0.03110

  NODE s16 -0.45520 0.00000
  NODE s17 -0.45520 0.03110
  NODE s18 -0.30210 0.00000
  NODE s19 -0.30210 0.03110
  NODE s20 -0.14910 0.00000
  NODE s21 -0.14910 0.03110
  NODE s22 -0.002000 0.00000
  NODE s23 -0.002000 0.03110


  LINE d17d15 d17 d15
  LINE d15d6  d15 d6
  LINE d16d14 d16 d14
  LINE d14d5  d14 d5
  LINE d16d17 d16 d17
  LINE d5d6   d5  d6
```

```
LINE d0d1 d0 d1
LINE d3d4 d3 d4
LINE d5d3 d5 d3
LINE d3d2 d3 d2
LINE d2d0 d2 d0
LINE d6d4 d6 d4
LINE d4d7 d4 d7
LINE d7d1 d7 d1
LINE d0s2 d0 s2
LINE d1d10 d1 d10
LINE d10s3 d10 s3

LINE s2s3 s2 s3
LINE s2s4 s2 s4
LINE s3s5 s3 s5
LINE s4s5 s4 s5
LINE s4s6 s4 s6
LINE s5s7 s5 s7
LINE s6s7 s6 s7
LINE s6s8 s6 s8
LINE s7s9 s7 s9
LINE s8s9 s8 s9
LINE s8s10  s8  s10
LINE s9s11  s9  s11
LINE s10s11 s10 s11
LINE s10s12 s10 s12
LINE s11s13 s11 s13
LINE s12s13 s12 s13
LINE s12s14 s12 s14
LINE s13s15 s13 s15
LINE s14s15 s14 s15
LINE s14s16 s14 s16
LINE s15s17 s15 s17
LINE s16s17 s16 s17
LINE s16s18 s16 s18
LINE s17s19 s17 s19
LINE s18s19 s18 s19
LINE s18s20 s18 s20
LINE s19s21 s19 s21
LINE s20s21 s20 s21
LINE s20s22 s20 s22
LINE s21s23 s21 s23
LINE s22s23 s22 s23

POLYLINE north0 2 + d17d15 + d15d6
POLYLINE south0 2 + d16d14 + d14d5
POLYLINE east0  1 + d5d6
POLYLINE west0  1 + d16d17

POLYLINE north1 1 + d6d4
POLYLINE south1 1 + d5d3
POLYLINE east1  1 + d3d4

POLYLINE north2  2 + d4d7 + d7d1
POLYLINE south2  2 + d3d2 + d2d0
POLYLINE east2   1 + d0d1
```

```
  POLYLINE north3  1 + d1d10 + d10s3
  POLYLINE south3  1 + d0s2
  POLYLINE east3   1 + s2s3

  POLYLINE north4  1 + s3s5
  POLYLINE south4  1 + s2s4
  POLYLINE east4   1 + s4s5

  POLYLINE north5  1 + s5s7
  POLYLINE south5  1 + s4s6
  POLYLINE east5   1 + s6s7

  POLYLINE north6  1 + s7s9
  POLYLINE south6  1 + s6s8
  POLYLINE east6   1 + s8s9

  POLYLINE north7  1 + s9s11
  POLYLINE south7  1 + s8s10
  POLYLINE east7   1 + s10s11

  POLYLINE north8  1 + s11s13
  POLYLINE south8  1 + s10s12
  POLYLINE east8   1 + s12s13

  POLYLINE north9  1 + s13s15
  POLYLINE south9  1 + s12s14
  POLYLINE east9   1 + s14s15

  POLYLINE north10 1 + s15s17
  POLYLINE south10 1 + s14s16
  POLYLINE east10  1 + s16s17

  POLYLINE north11 1 + s17s19
  POLYLINE south11 1 + s16s18
  POLYLINE east11  1 + s18s19

  POLYLINE north12 1 + s19s21
  POLYLINE south12 1 + s18s20
  POLYLINE east12  1 + s20s21

  POLYLINE north13 1 + s21s23
  POLYLINE south13 1 + s20s22
  POLYLINE east13  1 + s22s23

END_GEOMETRY

BEGIN_FLOW

  #GAS_TYPE perf_n2
  GAS_TYPE LUTN
  GAS_STATE driver   3.20e6 0.0 0.0 310.00 1.0 0.0 0.0 0.0 0.0
  GAS_STATE driven   3.00e4 0.0 0.0 296.00 0.0 1.0 0.0 0.0 0.0

  DISCRETISE north0  140 0 0 0.0
  DISCRETISE south0  140 1 0 1.6
```

```
DISCRETISE east0   80  0 1 1.1
DISCRETISE west0   80  0 1 1.1

DISCRETISE north1  220 0 0 0.0
DISCRETISE south1  220 0 0 0.0
DISCRETISE east1   80  0 1 1.1

DISCRETISE north2  220 0 0 0.0
DISCRETISE south2  220 0 0 0.0
DISCRETISE east2   80  0 1 1.1

DISCRETISE north3  240 0 0 0.0
DISCRETISE south3  240 0 0 0.0
DISCRETISE east3   80  0 1 1.1

DISCRETISE north4  240 0 0 0.0
DISCRETISE south4  240 0 0 0.0
DISCRETISE east4   80  0 1 1.1

DISCRETISE north5  240 0 0 0.0
DISCRETISE south5  240 0 0 0.0
DISCRETISE east5   80  0 1 1.1

DISCRETISE north6  240 0 0 0.0
DISCRETISE south6  240 0 0 0.0
DISCRETISE east6   80  0 1 1.1

DISCRETISE north7  240 0 0 0.0
DISCRETISE south7  240 0 0 0.0
DISCRETISE east7   80  0 1 1.1

DISCRETISE north8  240 0 0 0.0
DISCRETISE south8  240 0 0 0.0
DISCRETISE east8   80  0 1 1.1

DISCRETISE north9  240 0 0 0.0
DISCRETISE south9  240 0 0 0.0
DISCRETISE east9   80  0 1 1.1

DISCRETISE north10 240 0 1 1.3
DISCRETISE south10 240 0 1 1.3
DISCRETISE east10  80  0 1 1.1

DISCRETISE north11 240 0 0 0.0
DISCRETISE south11 240 0 0 0.0
DISCRETISE east11  80  0 1 1.1

DISCRETISE north12 240 0 0 0.0
DISCRETISE south12 240 0 0 0.0
DISCRETISE east12  80  0 1 1.1

DISCRETISE north13 240 0 0 0.0
DISCRETISE south13 240 0 0 0.0
DISCRETISE east13  80  0 1 1.1

BOUNDARY_SPEC north0  FIXED_T 296.0
```

```
BOUNDARY_SPEC north1  FIXED_T 296.0
BOUNDARY_SPEC north2  FIXED_T 296.0
BOUNDARY_SPEC north3  FIXED_T 296.0
BOUNDARY_SPEC north4  FIXED_T 296.0
BOUNDARY_SPEC north5  FIXED_T 296.0
BOUNDARY_SPEC north6  FIXED_T 296.0
BOUNDARY_SPEC north7  FIXED_T 296.0
BOUNDARY_SPEC north8  FIXED_T 296.0
BOUNDARY_SPEC north9  FIXED_T 296.0
BOUNDARY_SPEC north10 FIXED_T 296.0
BOUNDARY_SPEC north11 FIXED_T 296.0
BOUNDARY_SPEC north12 FIXED_T 296.0
BOUNDARY_SPEC north13 FIXED_T 296.0

BLOCK driver1 + north0  + east0  + south0  + west0
BLOCK driver2 + north1  + east1  + south1  + east0
BLOCK driver3 + north2  + east2  + south2  + east1
BLOCK shock1  + north3  + east3  + south3  + east2
BLOCK shock2  + north4  + east4  + south4  + east3
BLOCK shock3  + north5  + east5  + south5  + east4
BLOCK shock4  + north6  + east6  + south6  + east5
BLOCK shock5  + north7  + east7  + south7  + east6
BLOCK shock6  + north8  + east8  + south8  + east7
BLOCK shock7  + north9  + east9  + south9  + east8
BLOCK shock8  + north10 + east10 + south10 + east9
BLOCK shock9  + north11 + east11 + south11 + east10
BLOCK shock10 + north12 + east12 + south12 + east11
BLOCK shock11 + north13 + east13 + south13 + east12

CONNECT_BLOCKS driver1 east  driver2 west
CONNECT_BLOCKS driver2 east  driver3 west
CONNECT_BLOCKS driver3 east  shock1  west
CONNECT_BLOCKS shock1  east  shock2  west
CONNECT_BLOCKS shock2  east  shock3  west
CONNECT_BLOCKS shock3  east  shock4  west
CONNECT_BLOCKS shock4  east  shock5  west
CONNECT_BLOCKS shock5  east  shock6  west
CONNECT_BLOCKS shock6  east  shock7  west
CONNECT_BLOCKS shock7  east  shock8  west
CONNECT_BLOCKS shock8  east  shock9  west
CONNECT_BLOCKS shock9  east  shock10 west
CONNECT_BLOCKS shock10 east  shock11 west

FILL_BLOCK driver1 driver
FILL_BLOCK driver2 driver
FILL_BLOCK driver3 driver
FILL_BLOCK shock1  driven
FILL_BLOCK shock2  driven
FILL_BLOCK shock3  driven
FILL_BLOCK shock4  driven
FILL_BLOCK shock5  driven
FILL_BLOCK shock6  driven
FILL_BLOCK shock7  driven
FILL_BLOCK shock8  driven
FILL_BLOCK shock9  driven
FILL_BLOCK shock10 driven
```

```
  FILL_BLOCK shock11 driven

END_FLOW

BEGIN_CONTROL

  TITLE Entire Drummond Tunnel Simulation (Diaphragm Rupture, Blanked End)
  # progressive blocks can be activated in mb_special_init.c
  CASE_ID 101

  AXISYMMETRIC
  VISCOUS
  TURBULENT shock1
  TURBULENT shock2
  TURBULENT shock3
  TURBULENT shock4
  TURBULENT shock5
  TURBULENT shock6
  TURBULENT shock7
  TURBULENT shock8
  TURBULENT shock9
  TURBULENT shock10
  TURBULENT shock11
  FLUX_CALC  adaptive
  MAX_TIME   1.0e-2
  MAX_STEP   400000
  TIME_STEP  0.2e-8
  DT_PLOT    1.0e-4
  DT_HISTORY 1.0e-6
  #Rake at 1015mm position
  HISTORY_CELL shock6  88 1
  HISTORY_CELL shock6  88 20
  HISTORY_CELL shock6  88 46
  HISTORY_CELL shock6  88 48
  #Rake at 524mm position
  HISTORY_CELL shock8  76 1
  HISTORY_CELL shock8  76 20
  HISTORY_CELL shock8  76 46
  HISTORY_CELL shock8  76 48
  #Shock tube wall points A and B
  HISTORY_CELL shock10 24 80
  HISTORY_CELL shock11 134 80

END_CONTROL

BEZIER_FILE drummond_tunnel.bez
PARAM_FILE drummond_tunnel.p
BUILD

EXIT
```

APPENDIX C

MB_CNS Special Case Files

```
/* mb_special_init.inc
 *
 * Section of code that deals with the initialisation of
 * special cases.
 *
 * This file is "included" in mb_cns.c.
 */
if (Case_ID == DRUMMOND_TUNNEL_M4NOZZLE) {
    /* Primary Diaphragm Rupture Model:
     * this is the number (the first one being zero) of the block
     * upstream from the diaphragm
     */
    diaphragm_block = 2;

    /* assuming a linear profile of ruptured are versus time, the total
     * opening time in seconds
     */
    diaphragm_rupture_time = 0.94*200.0e-6;
    diaphragm_rupture_diameter = 57.0e-3;

    sprintf(msg_text, "\n... activating diaphragm rupture
      parameters: block %d, rupture time = %fus,
      rupture diameter = %fmm\n",
      diaphragm_block, diaphragm_rupture_time*1.0e6,
      diaphragm_rupture_diameter*1.0e3);
    log_message (msg_text, 1);

    /* Secondary Diaphragm:
     * Initially set blocks[16]-[23] as inactive and put reflective
     * boundary conditions between block[15] and block[16] (as the secondary
     * diaphragm).
     */
    bd[15].bc_E = 3;
    bd[16].bc_W = 3;
    secondary_diaphragm_ruptured = 0;
    G.active[16] = 0;
    G.active[17] = 0;
    G.active[18] = 0;
    G.active[19] = 0;
    G.active[20] = 0;
    G.active[21] = 0;
    G.active[22] = 0;
    G.active[23] = 0;
```

```
    /* Progressive Shock Tube Blocks:
     * Using progressive switching on of blocks along the shock tube.
     * Initially only the three driver blocks and the first shock tube
     * block are turned on and the boundaries are opened.
     */

    /* use progressive blocks? */
    drummond_progressive = 1;

    if (drummond_progressive == 1) {
        bd[3].bc_E = 3;
        for (block_counter = 4; block_counter < 15; block_counter++) {
            tube_block_activated[block_counter] = 0;
            G.active[block_counter] = 0;
            bd[block_counter].bc_E = 3;
            bd[block_counter].bc_W = 3;
        }
        bd[15].bc_W = 3;
        G.active[15] = 0;
        tube_block_activated[15] = 0;
    }
}   /* end of DRUMMOND_TUNNEL_M4NOZZLE */


if ( Case_ID == DRUMMOND_TUNNEL_BLANKED ) {
    /* Primary Diaphragm Rupture Model:
     * this is the number (the first one being zero) of the block
     * upstream from the diaphragm
     */
    diaphragm_block = 2;

    /* assuming a linear profile of ruptured are versus
     * time, the total
     * opening time in seconds
     */
    diaphragm_rupture_time = 0.94*200.0e-6;
    /*   diaphragm_rupture_time = 50.0e-6; */
    diaphragm_rupture_diameter = 57.0e-3;

    sprintf(msg_text, "\n... activating diaphragm rupture
      parameters: block %d, rupture time = %fus,
      rupture diameter = %fmm\n",
      diaphragm_block, diaphragm_rupture_time*1.0e6,
      diaphragm_rupture_diameter*1.0e3);
    log_message (msg_text, 1);

    /* Using progressive switching on of blocks along the shock tube.
     * Initially only the three driver blocks and the first shock tube
     * block are turned on and the boundaries are opened.
     */

    /* use progressive blocks? */
    drummond_progressive = 1;

    if (drummond_progressive == 1) {
        bd[3].bc_E = 3;
```

```
        for (block_counter = 4; block_counter < 13; block_counter++) {
            tube_block_activated[block_counter] = 0;
            G.active[block_counter] = 0;
            bd[block_counter].bc_E = 3;
            bd[block_counter].bc_W = 3;
        }
        bd[13].bc_W = 3;
        G.active[13] = 0;
        tube_block_activated[13] = 0;
    }
}   /* end if DRUMMOND_TUNNEL_BLANKED */
```

```
/* mb_special_step.inc
 *
 * Special-case code for inclusion at the start of a time step.
 */
if (Case_ID == DRUMMOND_TUNNEL_M4NOZZLE) {
 /* Check for secondary diaphragm rupture */
 pp = bd[15].Ctr[bd[15].ixmax][bd[15].iymin].gas.p;
 if ( secondary_diaphragm_ruptured == 0 && pp > 1.0e5 ) {
    secondary_diaphragm_ruptured = 1;
    sprintf (msg_text, "\n... Secondary diaphragm ruptured at p = %e\n", pp);
    log_message (msg_text, 1);
    G.active[16] = 1;
    G.active[17] = 1;
    G.active[18] = 1;
    G.active[19] = 1;
    G.active[20] = 1;
    G.active[21] = 1;
    G.active[22] = 1;
    G.active[23] = 1;
    bd[15].bc_E = 0;
    bd[16].bc_W = 0;
 }

 /* progressive blocks: activate tube blocks if the x velocity in the cell 5
  * from the end of the block before it in the tube rises over 1.0m/s
  */
 if (drummond_progressive == 1) {
   for (block_counter = 4; block_counter < 16; block_counter++) {
      if (tube_block_activated[block_counter] == 0) {
        /*
         * So far, this is an inactive block.
         * Look a few cells into the upstream block to see if the shock
         * in coming (as indicated by a velocity rise).
         */
         pp = bd[block_counter-1].Ctr[bd[block_counter-1].ixmax-5][bd[block_counter-1].iymin].u;
         if (pp >= 1.0) {
          sprintf (msg_text, "\n... Switching on block %d with %em/s\n", block_counter, pp);
          log_message (msg_text, 1);
           /* then activate this block and the boundary before it */
           bd[block_counter-1].bc_E = 0;
           G.active[block_counter] = 1;
           bd[block_counter].bc_W = 0;
           tube_block_activated[block_counter] = 1;
         }
      }
   }
 }
}   /* end of DRUMMOND_TUNNEL_M4NOZZLE */


if (Case_ID == DRUMMOND_TUNNEL_BLANKED) {
 /* progressive blocks: activate tube blocks if the x velocity in the cell 5
  * from the end of the block before it in the tube rises over 1.0m/s
  */

 if (drummond_progressive == 1) {
```

```
    for (block_counter = 4; block_counter < 14; block_counter++) {
        if (tube_block_activated[block_counter] == 0) {
          /*
           * So far, this is an inactive block.
           * Look a few cells into the upstream block to see if the shock
           * in coming (as indicated by a velocity rise).
           */
          pp = bd[block_counter-1].Ctr[bd[block_counter-1].ixmax-5][bd[block_counter-1].iymin].u
          if (pp >= 1.0) {
      sprintf (msg_text, "\n... Switching on block %d with %ems-1\n", block_counter, pp);
         log_message (msg_text, 1);
              /* then activate this block and the boundary before it */
              bd[block_counter-1].bc_E = 0;
      G.active[block_counter] = 1;
         bd[block_counter].bc_W = 0;
              tube_block_activated[block_counter] = 1;
           }
        }
     }
 }
} /* if DRUMMOND_BLANKED_BLANKED end */
```