Validating dense-gas models recently added to the Computational Fluid Dynamics code Eilmer3 at University of Queensland

T. Hommes
G.H.P. Campmans

January 28, 2013

Thomas Hommes S0168319
Geert Campmans S0206148

Internship location:
University of Queensland,
Queensland Geothermal Energy Centre of Excellence
Dr. Peter Jacobs
Brisbane, Australia

Period:
Thomas Hommes: 17-sept – 17-dec 2012
Geert Campmans: 3-sept – 3-dec 2012

Home university:
University of Twente, CTW, Engineering Fluid Dynamics
prof.dr.ir. H.W.M. Hoeijmakers

Contents

Chapter 1

Introduction

1.1 Abstract

At the University of Queensland in the department Queensland Geothermal Energy Centre of Excellence a Computational Fluid Dynamics code named Eilmer3 has been developed. In this code it is recently made possible to alternate the gas model for the calculations. This internship can be roughly divided in three parts: First of all Eilmer3 is used to do simulations on a shock tube with different gas models. Furthermore it is investigated whether Eilmer3 is able to cope with Non-classical gas behavior, according to some authors this might occur in a shock tube for high molecular fluids.

Secondly, at the University of Southern Queensland Ghassan Al-Doori did some experiments with a steam ejector. With Eilmer3 he tried to validate his experiments but his model did not yet give the appropriate results. By alternating the gas model it was tried to obtain better results. Only an adjusted ideal gas model did indeed give better results other gas models couldn't converge due to the thermodynamic non-equilibrium.

Finally along the way there seemed to be some problems in Eilmer3 with the subsonic inflow boundary condition and the development of the turbulent boundary layer. Both these things are investigated and the subsonic boundary condition could easily be reprogrammed however for the development of the turbulent boundary layer more research was needed which is not conducted due to a lack of time.

Chapter 2

An introduction to Eilmer3

In the 1980's most available CFD codes used finite difference methods to solve the compressible Navier-Stokes equations. However these codes did not do a good job of capturing strong shocks. So one started working on a new CFD code using the by that time promising technique of Finite volume methods. The development of the Eilmer3 code was started at NASA Langley but it is further developed during the 1990's and 2000's at the University of Queensland.

2.1 Getting Started

Nowadays Eilmer3 is a CFD code which can perform transient flow simulations in two and three dimensions based on finite volume method. The code uses many different scripts in several different programming languages to run different routines in the code, these routines together form the software. Running Eilmer3 is most convenient on a LINUX operating system. The source code to build the software is available through the University of Queensland. The source code will build the executable files for the code. Several other programs and libraries are needed to complete the program. Eilmer3 uses these libraries to call on different functions necessary for the simulations. When all the executables are in place and the relevant libraries are installed Eilmer3 is ready to be used.

2.2 Setting up a simulation

To set up a simulation one has to define the problem in such a way that Eilmer3 can interpret it. This is done using a python script called: `job.py`, where `job` is the simulation job-name. In the python script the flow problem has to be specified, therefore a lot of things have to be specified such as:

- The gas model has to be selected one can choose from different models. Amongst the possible gas models are; 'Ideal gas', 'van der Waals gas' and 'Nist Refprop', these are the gas models that are mostly used in this study.
- The fluid composition has to be defined is it a pure fluid or a mixture such as air.
- The geometry of the flow domain has to be specified. This is done by defining nodes, lines and blocks. The lines can either be straight lines or curved lines like arc sections, such that a complex geometry can be build. It is also possible to create a rotational symmetrical body. The common vertexes of the blocks will not obstruct the flow domain, all the other vertexes will be slip wall boundary conditions by default.
- The boundary conditions need to be specified for example: to subsonic or supersonic in- or out-flow conditions. The geometry and boundary conditions are defined in a rather abstract way therefore it can be difficult to see whether the correct flow problem is defined. The solution for this problem is to let the preparation file construct a drawing of the geometry and its boundary conditions, this helps a lot in verifying the geometry of the flow problem.
- The number of nodes in the spatial dimensions need to be specified for each block in order to make a mesh.
- Finally the maximum simulation time and or maximum number of time steps need to be specified. As does the initial time step size, although this will be altered by the program itself as it uses the CFL number to adjust its time step. This will be done to make sure that the solution will converge. Also set the times when the solver should save the solution so that it can be viewed later on at these times. Saving the data can be either a spatial capture of the flow solution at a specified

time and or a time series at a specified point which for example can be compared with an experimental pressure probe or temperature sensor.

2.3 The structure of Eilmer3

The total simulation job consist of three main stages; the preparation, the simulation itself and the post-processing. These stages of the simulation are called by a shell script, this script is the main script which leads the other scrips. In figure 2.3 a schematic representation of the program is shown. The preparation file *e3prep.py* will take the simulation parameters from *job.py*, and prepare the simulation. It will build the geometry, create a mesh, set the initial conditions the pressures and temperatures in the domain and the gas model. The file *e3shared.exe* will actually solve the flow problem. Finally the post processing file *e3post.py* will take the flow solutions and convert it into a readable format for e.g. Paraview or in a format such that it can be used to make plots easily in a python plotting script.

2.4 Python modules

Eilmer3 uses a lot of modules the job-file for uses for example functions to define the gas model and to define the geometry. These functions are stored in modules, if a module is imported the script can use the functions that are defined in these modules. For example *gaspy.py* is the Python module where functions like *select_gas_model*, the function that selects the gas model, and the functions for setting the boundary conditions are defined. *gaspy.py* is an example of a modules written for Eilmer3, however more general python modules are also used, such as *numpy.py*. These modules are available through the internet because python is an open source program.

2.5 Gas model

Because the Gas model is an important part of this study, the way Eilmer3 uses it will be discussed in more detail. In the *job.py* file the gas model is selected depending on the gas model type Eilmer3 looks up for the equation of state. This can be done in two ways: The required equation of state will be build from certain fluid specific constants. For example the Van der Waals

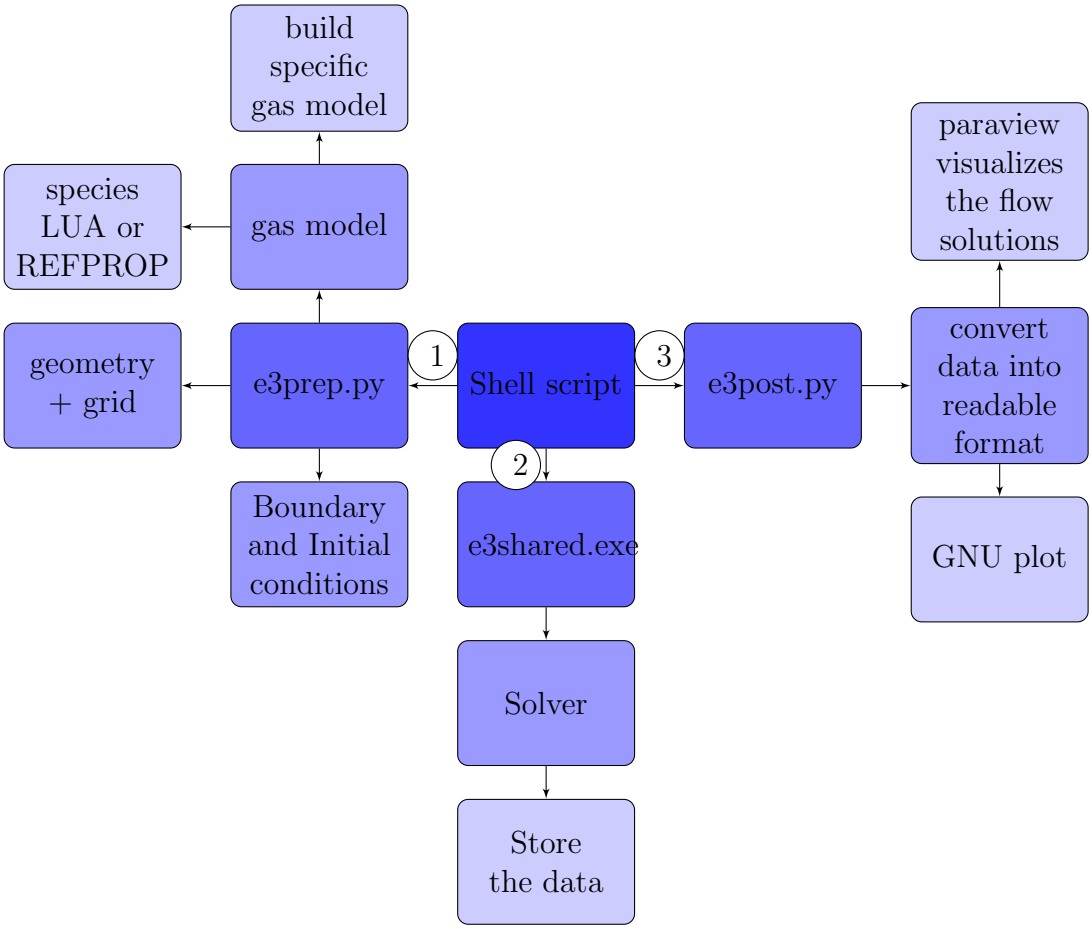


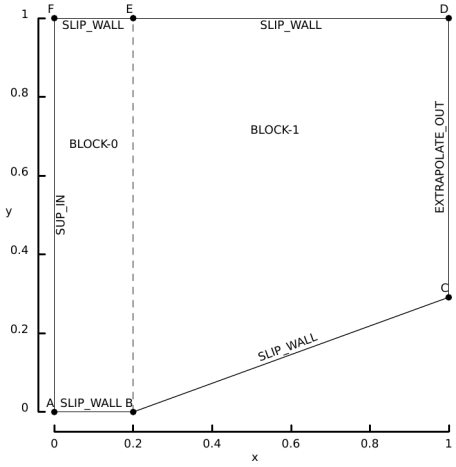
Figure 2.1: Eilmer3 schematic. The shell script will first call *e3prep.py* to prepare the simulation, then *e3share.exe* to run the simulation and finally *e3post.py* to process the data to a readable format.

gasconstants a and b can be computed from the molar mass, the critical temperature and the critical pressure. In a similar way the required gasconstants for other gas models can be computed. The required constants can be found in especially designed lua files, which are stored in the species library from Eilmer3. Each type of fluid has his own lua file with fluid specific information like molar mass, ratio of specific heats, viscosity, thermal conductivity etcetera. Depending on the gas model Eilmer3 uses these constants to build the gas model. There is one gas model available which is build in another way, that is the Nist Refprop gas model. The Nist Refprop gas model uses the program Nist Refprop to build a look up table. Nist Refprop is a database which contains for a wide variety of fluids the thermodynamic equilibrium tables and is one of the most accurate databases in circulation. The look up table is read and if necessary interpolated by Eilmer3 to obtain the other thermodynamic quantities.

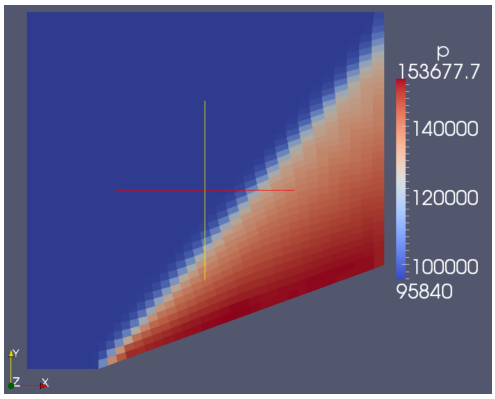
2.6 Example simulation

In this section an example is given on how to build and run a simple simulation of a 20 degree cone in a supersonic flow. A schematic overview of the flow problem is shown in Figure 2.2(a). The job-file in this case is *cone20.py* see Appendix B.1. In the python file one can see how the gas model is set line 21, the initial conditions defined lines 24-25, the geometry defined lines 32-53, the boundary conditions set 54-56, and the simulation parameters in lines 60-67.

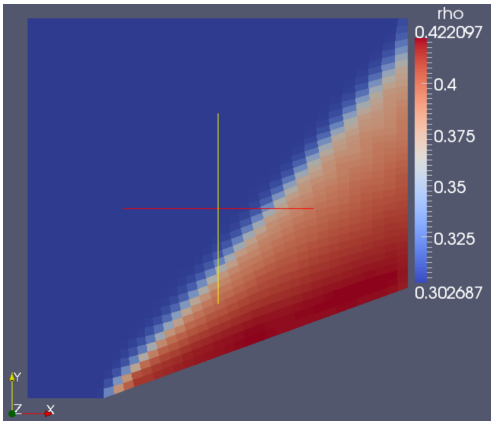
To run the simulation a shell script is used. The shell script calls the preparation file in line 6, the actual calculation in line 14 and the post-processing file in line 22. Just these three lines will do, however there are some comments added as well as some error messages in case things go wrong. If the simulation ran properly it should be possible to view the results in the visualization program Paraview. In figure 2.2 the density, pressure, and velocity are shown as Paraview displays it.



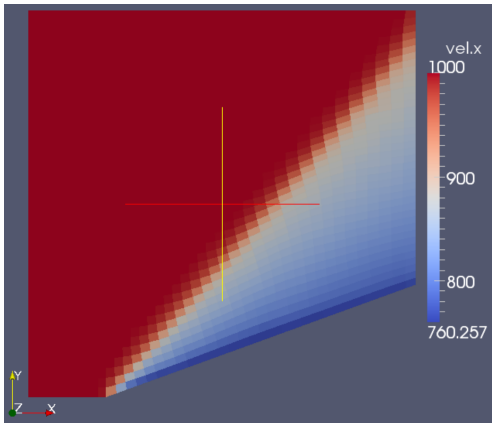
(a) Schematic overview



(b) pressure



(c) density



(d) velocity in x-direction

Figure 2.2: The results from Eilmer3's simulation visualized with Paraview

Chapter 3

Theory

During this study a lot of simulations are done with Eilmer3. Some research is done for the implementation of dense gas models in certain situations. Furthermore the use Eilmer3 for the simulation of steam ejectors will be discussed. In order to be able to cope with the results provided by Eilmer3 it is important to have a proper understanding of the theoretical background of Eilmer3. This chapter will give a brief overview of the theoretical background used by the software.

3.1 Theoretical background

The Eilmer3 software solves the integral form of the conservation equations for mass, momentum and energy numerically. For the three dimensional case the integral can be expressed as:

$$\frac{\partial}{\partial t}\int_V U dV = -\oint_S \left(\overline{F_i} - \overline{F_v}\right)\cdot \vec{n} dA + \int_V Q dV \tag{3.1}$$

Where Q is zero when modeling a non-reacting single component gas, the other quantities are:

$$U = \begin{pmatrix} \rho \\ \rho\vec{u} \\ \rho E \end{pmatrix} \tag{3.2}$$

The total energy E consists of two parts: the internal energy e and the kinetic energy $\frac{1}{2}|\vec{u}|^2$. The inviscid component is:

$$\vec{F_i} = \begin{pmatrix} \rho(\vec{u} - \vec{u_S}) \\ \rho\vec{u}(\vec{u} - \vec{u_S}) + p\overline{\overline{I}} \\ \rho E(\vec{u} - \vec{u_S}) + p\vec{u} \end{pmatrix} \tag{3.3}$$

It should be noted that in most simulations U_s equals zero because the surface of the simulated object does not move.
The viscous component is:

$$\vec{F_v} = \begin{pmatrix} \vec{0} \\ \overline{\overline{\tau}} \\ \overline{\overline{\tau}} \cdot \vec{u} + \vec{q} \end{pmatrix} \tag{3.4}$$

Both the viscous stress tensor $\overline{\overline{\tau}}$ and the heat flux vector $\vec{q}$ can be expressed in other flow field variables by so called constitutive relations. A constitutive relation is a relation which is specific to a certain fluid or material. The heat flux vector is given by: $\vec{q} = k\frac{\partial T}{\partial \vec{x}}$ where k is the thermal conductivity. For $\overline{\overline{\tau}}$ it is assumed that the fluid behaves like a Newtonian fluid, which means that the viscous stresses are related to the rate of strain. With the summation convention the viscous stresses can be written as:

$$\tau_{ij} = \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) + \lambda\delta_{ij}\frac{\partial u_k}{\partial x_k} \tag{3.5}$$

Where λ can be expressed in terms of μ by Stokes hypothesis $\lambda = -\frac{2}{3}\mu$.

So there are five equations available however there are nine unknowns: density ρ, temperature T, pressure p, internal energy e, thermal conductivity k, viscosity μ and the three velocity components $\vec{u}$. The remaining four equations will be provided by the gas library. For example the thermodynamic state principle, which states that if the chemical composition of the fluid is fixed, then the local thermodynamic state is fixed completely by two independent thermodynamic variables, together with two equations of state will supply two equations. The other two equations are the two constitutive relations for the thermal conductivity and viscosity to complete the model of the flow. Different gas models can provide the required equations of state, like the perfect gas model, van der Waals gas or a Nist Refprop gas. Later on in this report the Nist Refprop gas model will be discussed further.

In normal flow one of the underlying assumptions is that the flow is uniform and homogeneous. Which means that things like chemical reactions and mass diffusion can be neglected and the fluid has the same composition everywhere. However for hypersonic flow this is no longer the case. In order to deal with those effects additional conservation equations are added to Eilmer3 for modeling the flow. However the simulations done during this study do not concern hypersonic flow. Therefore there will be no elaboration on these equations in this report. Although Eilmer3 is equipped with modules which can cope with hypersonic flow.

3.1.1 Discretization of the equations

Eilmer3 solves the conservation equations numerically in order to be able to do so these equations need to be discretized. The volume is divided into finite-volume cells, these cells are hexahedral with six quadrilateral interfacing surfaces. At the middle of the interface the flux into the neighboring cell is calculated. Integral 3.1 is approximated by the following algebraic expression:

$$\frac{dU}{dt} = -\frac{1}{V} \sum_{cell-surface} \left(\vec{F}_i - \vec{F}_v \right) \cdot \vec{n} dA + Q \tag{3.6}$$

where U and Q are cell-average values. Expression 3.6 will be integrated in time with small time steps throughout the whole flow domain. The different physical phenomenon represented by the governing equations are decoupled and the integration in time is done in a order of operations.

3.1.2 Gas model

As already described in section 3.1 the thermodynamic state principle states that the local thermodynamic state is fully described by two independent variables. The equations which describe the thermodynamic state are called equations of state. There are a lot of different gas models available each with different equations of state. Some simulations done in this study contain dense gas models. This can be done with different gas models such as: Nist Refprop, VDW and Bender gas. The REFPROP gas model uses the Nist Refprop thermodynamic and transport property database for the evaluation of all fluid properties. Nist Refprop is a commercial software package equipped with the most accurate models and equations of state available for

gas properties. One of the main advantages of the Nist Refprop gas model is that it remains valid close to the vapor dome. Which is a requirement for some of the simulations done in this study.

3.1.3 Dense gas regime

Because some simulations are done in or close to the dense gas regime this section will give some theoretical background about this gas regime. Gas is often described as an ideal gas, however it is well known that under certain conditions gas behaves far from ideally. For example if the temperatures and pressures of the gas are near the critical point or close to saturated conditions the ideal gas model is not able to predict physically correct behavior. In this thermodynamic region certain substances composed of polyatomic molecules can exhibit non-classical gas dynamic phenomena. Such as expansion shocks and compression waves which are impossible for an ideal gas because the second law of thermodynamics will be violated. However it is proven by Thompson and coworkers that in this thermodynamic region flow phenomena as expansion shock can exist. This thermodynamic regime is called the dense gas regime.

The nonlinear dynamics of gases are described by the so called fundamental derivative of gas dynamics Γ (Thompson 1971,).

$$\Gamma = 1 + \frac{\rho}{a}\left(\frac{\partial a}{\partial \rho}\right)_s = -\frac{\nu}{2}\frac{\left(\frac{\partial^2 P}{\partial \nu^2}\right)_s}{\left(\frac{\partial P}{\partial \nu}\right)_s} \tag{3.7}$$

Positive nonlinearity occurs when $\Gamma > 0$ disturbances steepen forward and form compression shocks. In case $\Gamma < 0$ there is negative nonlinearity and disturbances steepen backward and form expansion shocks. Therefore Γ is a measure for the nature of the gas behavior. When Γ is negative non classical gas behavior occurs and this is the dense gas regime. However this non-classical gas behavior such as expansions shocks can only occur for a small range of polyatomic substances. For most substances Γ will never be negative. Figure 3.1 shows the regions where Γ is negative these regions are called the dense gas regions.

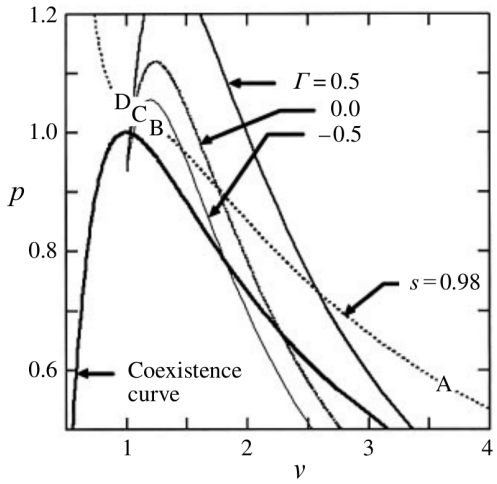


Figure 3.1: The PV-diagram from a van der Waals gas model used by B.P.Brown and B.M.Argrow [1], showing the constant-Γ contours.

Chapter 4

Dense gas shock tube simulations

In order to be able to determine the flow characteristics of a dense gas a simplified geometry is be used. A straight nozzle is chosen, which in fact is a shock tube. In figure 4 a schematic overview of the shock tube is given. The gas used within the shock tube is the refrigerant R245fa, on the left side of the diaphragm it will have a high pressure on the right side a low pressure. The diaphragm will be removed instantaneously which will result in an expansion wave into the driver gas and a shock wave into the test gas.

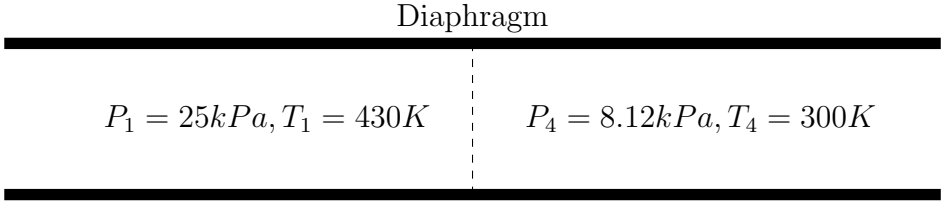


Figure 4.1: Schematic overview of a shock tube

The flow characteristics of this shock tube are computed with Eilmer3, for a dense gas. An Eilmer3 simulation file is made with a Nist Refprop gas model. In order to verify the calculations done by Eilmer3 some checks are done: first of all the analytical solution with an adjusted ideal gas is computed, furthermore the analytical solution is computed with the same Nist Refprop gas model however not with Eilmer3 but with the program written by Peter Blyton. If the different calculations give similar results it

means that Eilmer3's calculations are preformed in the way they are supposed to do. However this does not mean that the results from Eilmer3's calculation agree with the reality. In order to find that out the calculations will be compared to other data from papers.

4.1 CFD calculation of a shock tube by Eilmer3

The geometry of the shock tube is put into Eilmer3 and a NIST Refprop real gas model is used as gas model for the R245fa. The initial conditions of the driver gas are $P_1 = 25kPa, T_1 = 430K$ and for the driven gas $P_4 = 8.12kPa, T_4 = 300K$. The pressures are quite low and the temperatures relatively high, that is done to make sure that the R245FA remains a gas the whole time. Due to the expansion wave the temperature will drop, if the pressures are higher the expansion wave will be stronger so the temperature will drop even further. Furthermore at high pressures the boiling point is higher so it becomes more likely that the gas will condense if the temperature drops. Eilmer3 treats the fluid as a gas at all times, even when the conditions are changed in such a way that the fluid actually is the liquid phase. To make sure that the calculations of the shock tube are correct and to avoid any trouble with phase changes the pressures and temperatures will be chosen in such a way that the fluid will be gas the whole time.

Figure 4.2 shows the distribution of the density 1 millisecond after the instantaneous removal of the diaphragm. On first sight this results seems to be what is expected, the different flow regions can be seen. However it will be checked whether this result is truly correct by computing the analytical solutions with an adjusted ideal gas model, and with Peter Blython's program, see section 4.2.

4.2 Analytical solution

In this section the way of computing the analytical solution will be explained first. Later on the different gas models which are used will be described. The results from both analyses can be found in section 4.3.

After the instantaneous release of the diaphragm a shock will move into the test gas while an expansion wave will move into the driver gas. The flow can be divided in four regions:

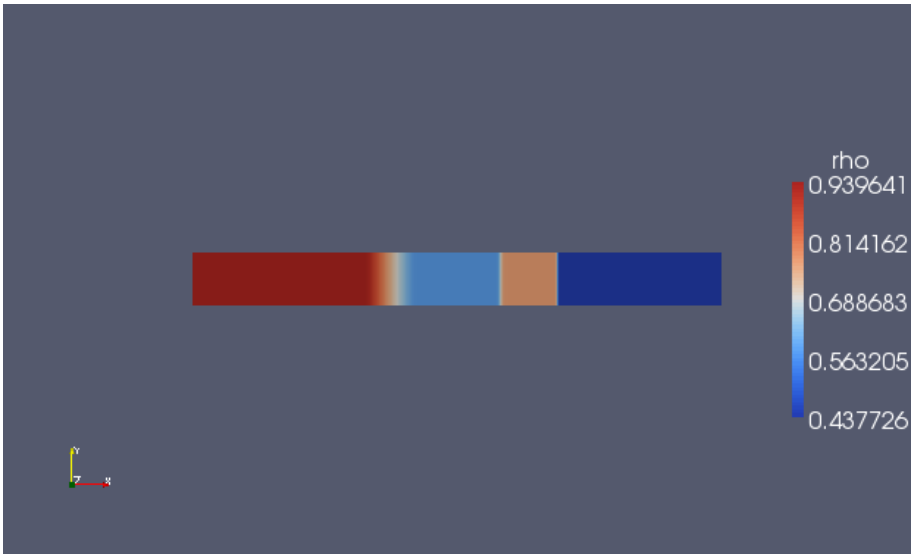


Figure 4.2: Density distribution in the shock tube at t=100 μs

- Region 1: is located in the driver gas reservoir and the gas in this region is still undisturbed.
- Region 2: is the simple-wave expansion region and is located between the first rarefaction wave and the contact surface. In this case the contact surface will travel with a constant velocity u_p which will result in the fact that the expansion wave consists of two parts. Region 2a is the centered expansion wave and region 2b is a uniform-flow region with the driver gas moving with the velocity of the contact surface.
- Region 3: is a uniform flow region between the contact surface and the shock wave with the test gas.
- Region 4: is located right of the shock wave and the gas is still undisturbed.

The General initial conditions of the shock-tube problem are:

$$\{p, T, \rho, u\} = \begin{cases} p_1, T_1, \rho_1, 0 & for \quad x < 0 \\ p_4, T_4, \rho_4, 0 & for \quad x > 0 \end{cases}$$

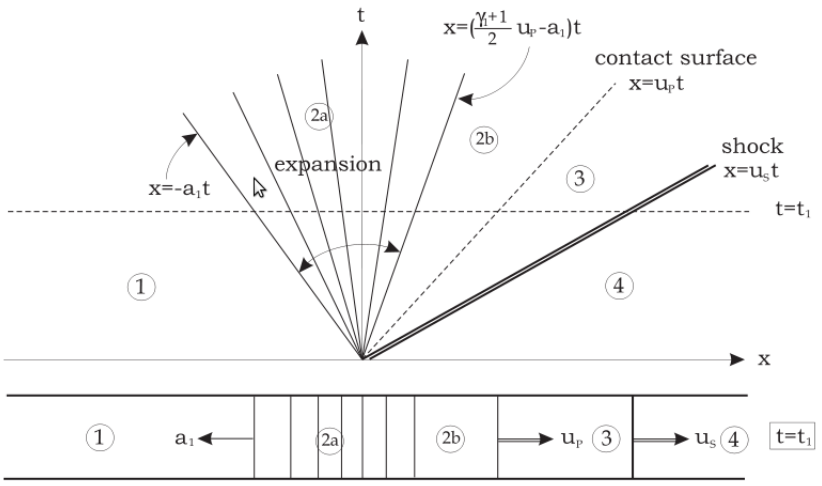


Figure 4.3: The different flow regions within the shock tube

The different flow regions are also shown in figure 4.2. The velocity, pressure, density and the speed of sound of region 1, 2 and 4 can be determined by using Riemann Invariants. This will not be done in this report only the results will be shown. The characteristics of region 3 can be expressed by using the Rankine-Hugoniot conditions.

4.2.1 Region 1

As mentioned earlier on region 1 is one of the undisturbed regions. The gas simply did not receive the information of the diaphragm removal yet. So all the parameters are still the initial ones.

$$p = p_1$$
$$T = T_1$$
$$\rho = \rho_1$$
$$u = 0$$
$$a = \sqrt{\gamma_1 \frac{p_1}{\rho_1}}$$

4.2.2 Region 2

Region 2 only contains driver gas that is accelerating or has been accelerated. This region is split into two subregions. The expanding region 2a and the uniform flow region 2b.

Region 2a

As the first rarefaction wave comes into the driver gas it starts to accelerate up to the velocity of the contact surface u_p.

$$p = p_1 \left(\frac{2}{\gamma_1 + 1} - \frac{\gamma_1 - 1}{\gamma_1 + 1} \frac{x}{a_1 t} \right)^{\frac{2\gamma_1}{\gamma_1 - 1}}$$

$$\rho = \rho_1 \left(\frac{2}{\gamma_1 + 1} - \frac{\gamma_1 - 1}{\gamma_1 + 1} \frac{x}{a_1 t} \right)^{\frac{2}{\gamma_1 - 1}}$$

$$u = \frac{2}{\gamma_1 + 1} a_1 + \frac{2}{\gamma_1 + 1} \frac{x}{t}$$

$$a = \frac{2}{\gamma_1 + 1} a_1 - \frac{\gamma - 1}{\gamma_1 + 1} \frac{x}{t}$$

Region 2b

In region 2b the driver gas has been fully expanded. And is moving at the contact surface velocity u_p. This contact surface velocity will be determined later on.

$$p = p_1 \left(1 - \frac{\gamma_1 - 1}{2} \frac{u_p}{a_1} \right)^{\frac{2\gamma_1}{\gamma_1 - 1}} \tag{4.1}$$

$$\rho = \rho_1 \left(1 - \frac{\gamma_1 - 1}{2} \frac{u_p}{a_1} \right)^{\frac{2}{\gamma_1 - 1}}$$

$$u = u_p$$

$$a = a_1 - \frac{\gamma_1 - 1}{2} u_p$$

4.2.3 Region 4

In region 4 the shock-wave did not pass yet, so the quantities are still the initial conditions.

$$p = p_4$$

$$\rho = \rho_4$$

$$u = 0$$

$$a = \sqrt{\gamma_4 \frac{p_4}{\rho_4}}$$

4.2.4 Region 3

Region 3 is the region where the test gas is moving uniformly at contact surface velocity. At region 3 the velocity is known and the other flow quantities will be determined by using the Rankine-Hugoniot relations:

$$u = u_p \tag{4.2}$$

$$U_4^2 = \gamma \frac{p_4}{\rho_4} \left(1 + \frac{\gamma + 1}{2\gamma} \left(\frac{p_3}{p_4} - 1\right)\right) > a_4^2 \tag{4.3}$$

$$\frac{p_3}{p_4} = 1 + \left(\gamma_4 \frac{u_p}{a_4}\right) \left(\frac{\gamma_4 + 1}{4\gamma_4} \left(\gamma_4 \frac{u_p}{a_4}\right) + \sqrt{1 + \left(\frac{\gamma_4 + 1}{4\gamma_4} \left(\gamma_4 \frac{u_p}{a_4}\right)\right)^2}\right) \tag{4.4}$$

$$\frac{\rho_3}{\rho_4} = \frac{1 + \frac{\gamma_4 + 1}{2\gamma_4} \left(\frac{p_3}{p_4} - 1\right)}{1 + \frac{\gamma_4 - 1}{2\gamma_4} \left(\frac{p_3}{p_4} - 1\right)} \tag{4.5}$$

Now the frame of reference will be changed and we will move along with the shock wave, as shown in figure 4.2.4.

Velocity U_3 and U_4 can be expressed in terms of u_s and u_p, if these and equation 4.4 are substituted in equation 4.3 and some rewriting is done equation 4.6 for the shock speed follows.

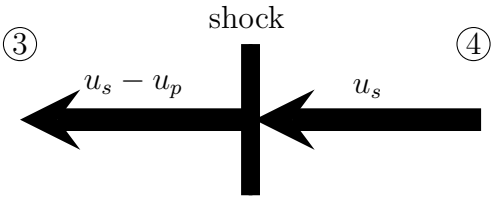


Figure 4.4: Frame of reference when moving with the shock

$$u_s = a_4 \left[\frac{\gamma_4 + 1}{4\gamma_4} \left(\gamma_4 \frac{u_p}{a_4} \right) + \sqrt{1 + \left(\frac{\gamma_4 + 1}{4\gamma_4} \left(\gamma_4 \frac{u_p}{a_4} \right) \right)^2} \right] \tag{4.6}$$

Now all flow characteristics are expressed in terms of the velocity of the contact surface u_p. This system can be solved by stating that the pressure in region 2b should be equal to the pressure in region 3, in other words the static pressure across a contact surface should be constant. Stating this will result in the nonlinear equation for 4.7 u_p:

$$\frac{p_1}{p_4} = \frac{1 + \left(\gamma_4 \frac{u_p}{a_4} \right) \left[\frac{\gamma_4 + 1}{4\gamma_4} \left(\gamma_4 \frac{u_p}{a_4} \right) + \sqrt{1 + \left(\frac{\gamma_4 + 1}{4\gamma_4} \left(\gamma_4 \frac{u_p}{a_4} \right) \right)^2} \right]}{\left(1 + \frac{\gamma_1 - 1}{2} \frac{u_p}{a_1} \right)^{\frac{2\gamma_1}{\gamma_1 - 1}}} \tag{4.7}$$

From equation 4.7 u_p can be solved iteratively by Newton's iterative solution procedure. When the velocity of the contact surface is known then all the other quantities can be determined. With this system of equations the analytical solution will be computed, however a gas model is required to determine the thermodynamic relations. Two different gas models are used to validate Eilmer3's calculations, an ideal gas model and a Nist Refprop gas model.

4.2.5 Ideal gas model

The gas is assumed to behave as an ideal gas and therefore obeys the thermal equation of state for a perfect gas:

$$Pv = RT \quad \text{with } R = c_p - c_v \tag{4.8}$$

However there is no specific kind of fluid chosen, the gas properties like R and γ are chosen in such a way that the results are similar to Eilmer3's calculations. If the initial conditions are changed so do the gas properties. This can be done because the purpose of the analytical solution, with an ideal gas model or Nist Refprop gas model, is to validate Eilmer3's calculations. So not to validate the gas model it self. That will be done by comparing the results from Eilmer3 with other data.

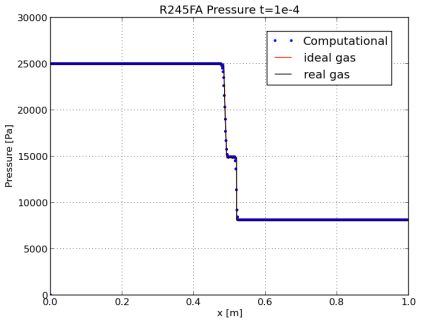
4.2.6 Nist Refprop gas model

Peter Blython, a member of the Queensland geothermal energy center of excellence, has written a program in python which makes it possible to calculate the analytical solution of the shock tube problem with a dense gas model. To verify Eilmer3's calculations this program is used together with the same Nist refprop gas model which is used for Eilmer3. The calculations are done in the same way as for the analytical solution with the ideal gas model. The results are shown in section 4.3.

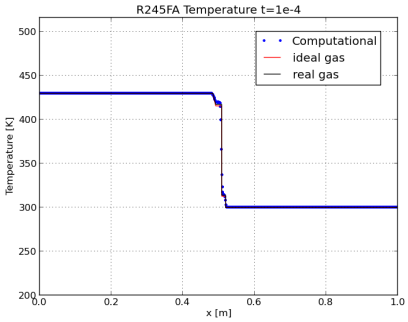
4.3 Results

In order to compare the different methods for the dense gas shock tube problem all results are shown in a (P,x), (T,x), (u_x,x) and a (ρ,x) diagram for t=100μs, as can be seen in figure 4.5.

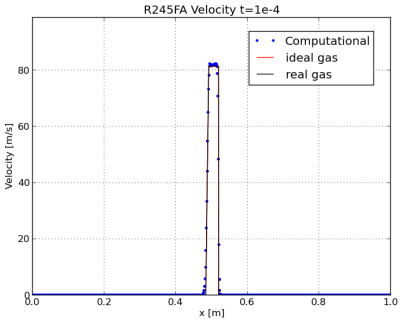
From these graphs it can be concluded that the solution from Eilmer3 looks very similar to the analytical solutions. The shape of the graphs is exactly as expected for all the three solution methods. The main difference is the transition to different phases of the flow, for the analytical solutions these transitions are very rigid. Whereas for Eilmer3's solution these transitions are smoother. It is remarkable that the analytical solution with ideal gas model is so similar to the other solutions. This can be explained by the relatively low pressures and temperatures because the features which are not modeled by the ideal gas model become more prominent and more influential at lower temperatures and higher pressures. Furthermore the properties of this ideal gas are adjusted to fit Eilmer3's solutions, as described in section 4.2.5. When the pressures and temperature at t=0 are adjusted this gas model will no longer be so similar.



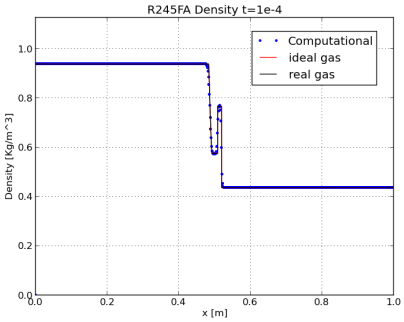
(a) (P,x) diagram



(b) (T,x) diagram



(c) (u_x,x) diagram



(d) (ρ,x) diagram

Figure 4.5:

From these graph it can be concluded that Eilmer3 performs the calculations itself in the right way. Because the analytical solution with the same Nist Refprop gas model as used in Eilmer3 gives almost the same results. So the calculations from Eilmer3 are therefore validated, however based on these results we cannot say whether the gas model is correct. This is due to the fact that both methods use the same gas model. To validate the gas models the results should be compared to other data available in literature.

Chapter 5

Non classical gas behavior

5.1 Verification of the gas model

When modelling high molecular fluids, like refrigerants, are concerned non classical gas behavior can occur. A.A Borisov *et al.* wrote a paper on the experiments with Freon-13 gas. They claim to have found experimental proof for non classical gas behavior such as expansion shocks. In this section we will first do some Eilmer3 calculations with a Nist Refprop gas model for R13 in the gassy regime and later on in the dense gas regime to see whether Eilmer3 reproduces non classical gas behavior.

5.1.1 The Freon-13 experiment

Freon-13 is the trivial name of the refrigerant chlorotrifluoromethane and is used by A.A. Borisov *et al.* for there experiments in a shock tube. The experiments in the shock tube will be simulated with Eilmer3, therefore Eilmer3's shock tube file will be used and a Nist Refprop gas model will be used. The first goal is to determine whether the Nist Refprop gas model will work at all with Eilmer3, for this gas. Therefore the initial conditions are chosen to be in the very gassy domain.

Furthermore it will be checked whether phenomena such as expansion shocks will occur when the solution goes to the dense gas regime, therefore other initial conditions will be chosen. The initial conditions for both situations are shown in Table 5.1.1. Moreover the solutions are drawn in figure 5.1, where the black line is the vapor dome of Freon-13. It is assumed that the shape of the negative fundamental derivative gas region is the same as

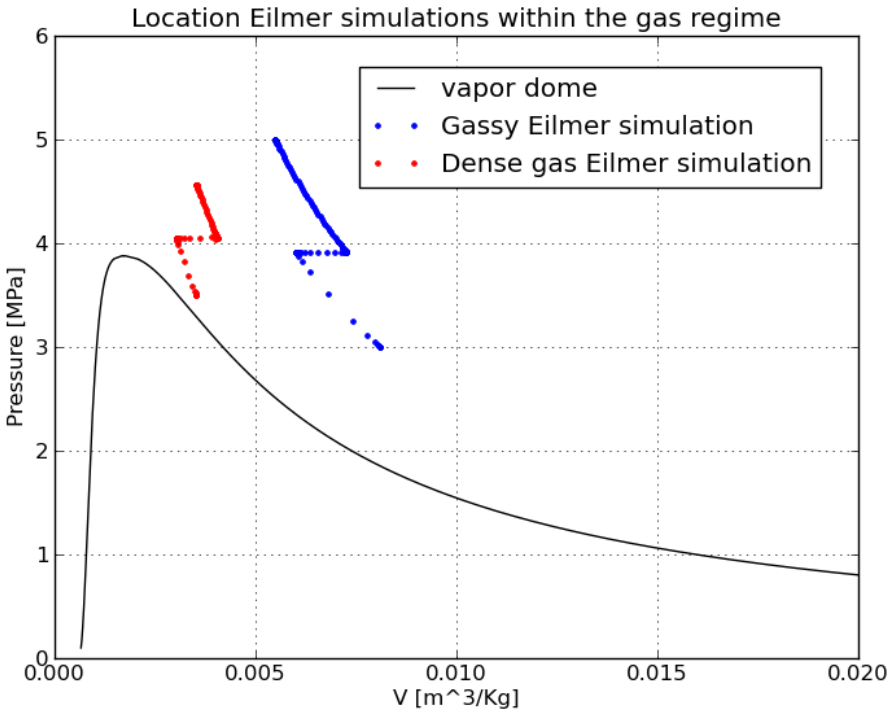
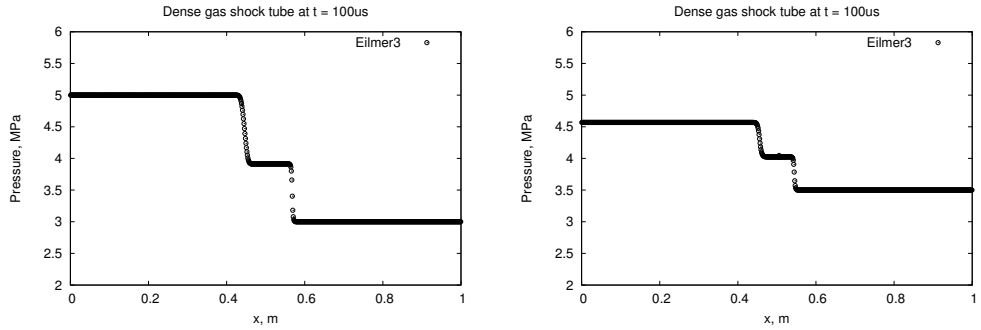


Figure 5.1: The P-v diagram for Freon-13 with the solutions from Eilmer3 drawn in there

shown in figure 3.1. This leads to the conclusion that non classical gas behavior is expected for the initial conditions corresponding to the dense gas regime.

Gassy gas regime	Driver gas	P=5e6 Pa	T=500 K
	Test gas	P=3e6 Pa	T=400 K
Dense gas regime	Driver gas	P=4.568e6 Pa	T=335 K
	Test gas	P=3.5e6 Pa	T=300 K

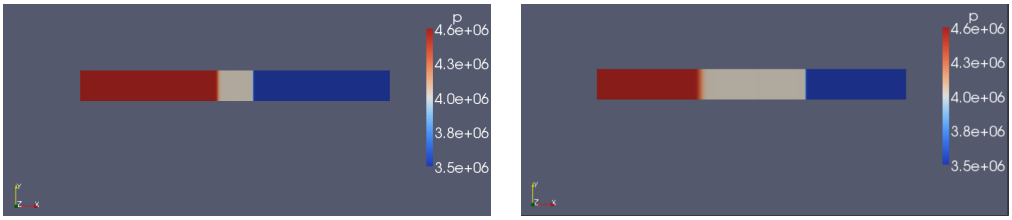


(a) Gassy regime (b) Dense gas regime

Figure 5.2: (P,x) diagram from Eilmer3's solution

Both simulations did run and got results as shown in figure 5.2. The results from the simulation in the gassy regime with the Nist Refprop model as shown in figure 5.2(a) seem to be as expected. The shock is clearly visible at x=0.57 and the expansion shock behaves as expected. The results produced by Eilmer3 seems to be quite promising. Therefore it can be concluded that Eilmer3 can cope with the Nist Refprop gas model in the gassy regime. Figure 5.2(b) shows a (P,x) diagram for the dense gas simulation. As one can see the shape is very similar to the gassy situation. Around x=0.505 something seems to be going on the contact surface is located at this location. The pressure should be continuous along the contact surface perhaps these vibrations in the solution are caused by numerical errors. The results also show a compression shock which does not correspond with Borisov his paper.

One of the arguments A.A. Borisov *et al.* use in their paper is that the expansion shock has the same thickness over time. An expansion wave will become wider as it linearly grows in time. In their experiments a constant



(a) The shock tube after $0.5\mu s$ (b) The shock tube after $2.0\mu s$

Figure 5.3: The development of the expansion wave in the shock tube

thickness of the expansion wave is observed and therefore it will be an expansion shock according to them. The development of the expansion wave in the results from Eilmer3 are shown in 5.3. If one looks at the expansion wave one can see that it becomes larger over time while the compression shock does not. Therefore it can be concluded that it is an expansion wave instead of an expansion shock. This means that there is no non classical gas behavior observed. This can be due to two things: either Eilmer3 does not cope with non classical gas behavior or the Nist Refprop gas model doe not have a dense gas regime. A.A. Borisov *et al.* say they have seen expansion shocks while using R13 but it is questioned by others. Therefore it will be investigated whether Freon 13 has a region where fundamental derivative is negative according to the the data provided by Nist Refprop, this is done in section 5.2.1. Further research needs to be done to make sure that Eilmer3 gives reliable results in this gas regime, however this is not done in this report.

So in conclusion the simulations done with Eilmer3 with a Nist Refprop gas model seems to work in the gassy regime. However in the dense gas regime the results do not agree with experiments observed by A.A. Borisov *et al.* Whether this is due to Eilmer3 or has an other cause needs to be investigated further.

5.2 Fundamental derivative

To check whether non classical gas behavior can occur when R13 is used the sign of the fundamental derivative will be checked. Based on the data supplied by Nist Refprop for R13 it will be checked if Γ is negative in a certain region. If so non-classical gas behavior is expected in that region.

5.2.1 Numerical approximation

As mentioned earlier the sign of the fundamental derivative determines whether non-classical gas behavior occurs or not. Since all other terms in equation 3.7 besides the second order derivative will not change sign equation 5.1 will fulfil.

$$sign\ \Gamma = sign\left(\frac{\partial^2 P}{\partial V^2}\right)_s \tag{5.1}$$

From NIST-REFPROP a data file of R13 was created with varying density at constant entropy. This data file is used to investigate in what regions the fundamental derivative is negative. Therefore an approximation of the second order derivative has to be made. This has been done by using the finite difference method. After discretization the second order derivative becomes equation 5.2

$$\left(\frac{\partial^2 P}{\partial V^2}\right)_s \approx \frac{\frac{P_{i+1}-P_i}{dV_{i+1/2}} - \frac{P_i-P_{i-1}}{dV_{i-1/2}}}{\frac{1}{2}\left(dV_{i-\frac{1}{2}} + dV_{i+\frac{1}{2}}\right)} \tag{5.2}$$

Where $dV_{i+\frac{1}{2}} = V_i - V_{i+1}$ from which the order of subtraction might look weird. Note that density is a monotonic increasing variable in this table, and thus after taking its reciprocal volume is thus a monotonic decreasing variable. Due to the decreasing volume this order is used to make sure the sign does not change. Before the results will be shown, there will be some elaboration on the expected results.

5.2.2 Expected results

Borisov *et al.* (1983) [4] experimentally observed an expansion shock with $CClF_3$ or Freon-13. The objective was to run this experiment with the Eilmer3 software but no non-classical behavior occurred. Γ should be negative in a certain region in order to have non classical gas behavior. Due to the fact that Borisov et AL claim they have found an expansion shock it will be expected that there is a negative region as shown in 5.7. The region just above and to the right of the vapor dome is the negative region and is therefore suspected to be the interesting region for R-13 as well.

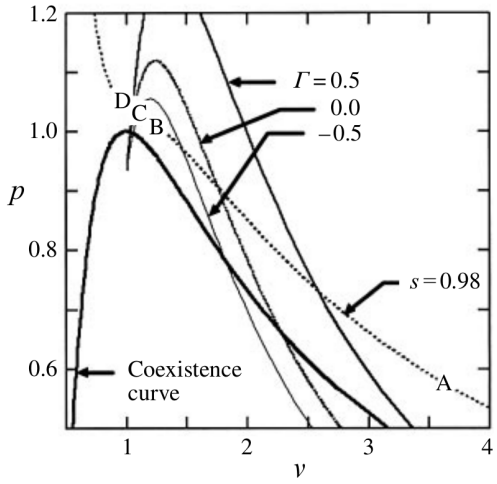
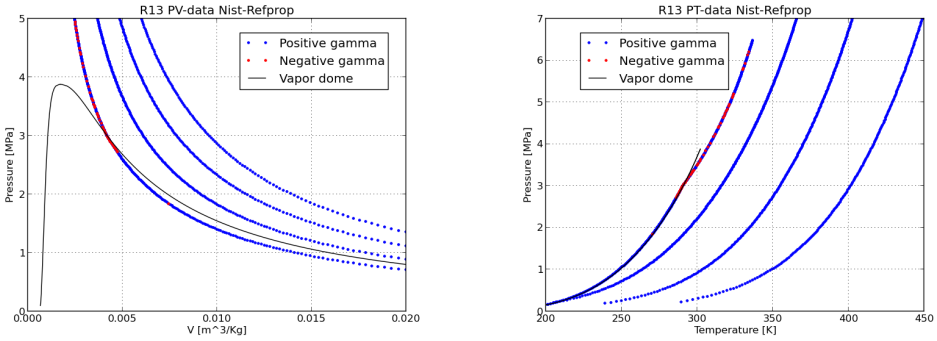


Figure 5.4: The PV-diagram from a van der Waals gas model used by B.P.Brown and B.M.Argrow [1], showing the constant-Γ contours. Note that the pressure and volume are normalized with the critical point values.

5.2.3 Results

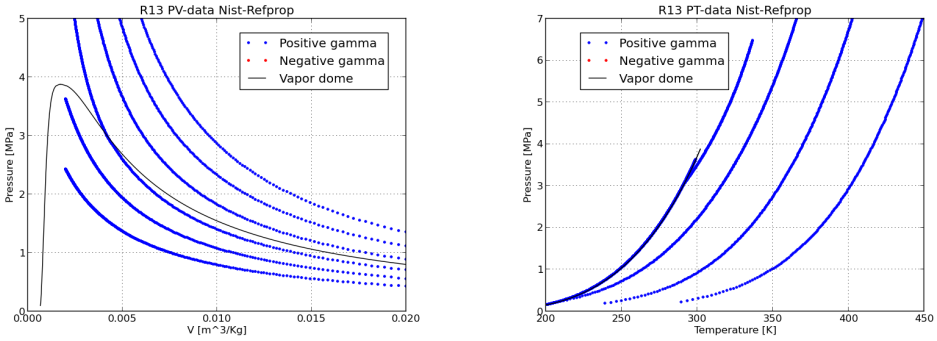
The first results of the numerical approach of the second order derivative seemed to be a bit strange. The sign appeared to be more or less swapping randomly every now and then see fig:5.5. To verify the numerical approximation some simple polynomials where used to check if this would give appropriate results. By doing this the following conclusion can be drawn: in order to get descent results the data on which the numerical derivative is applied should be fairly accurate. Nist Refprop rounded its output to five decimals, this was as can be seen in figure 5.5 not enough. Therefore Nist Refprop was set to give its output to 12 decimals and the result is shown in figure 5.6. As one can see the randomly swapping of the sign is resolved. Which makes sense, the rounding off can be seen as a "noise" signal. Taking the derivative of noise gives large amplitudes as result.The last step is to find the region where Γ is negative and figure 5.7 shows the result. As one can see Γ is not negative in the region where it is expected which is quite a surprising result. Γ is negative in the two phase region and this is not in agreement with the findings of Borisov *et al.* The data provided by the Nist Refprop database is amongst the most accurate in the world, so according to this analysis there is no valid dense gas regime for R13. Which will contradict

the paper by Borisov *et al.* Moreover there are more researched who doubt their findings. Due to the fact that this is beyond the scope of this study it will not be investigated further.



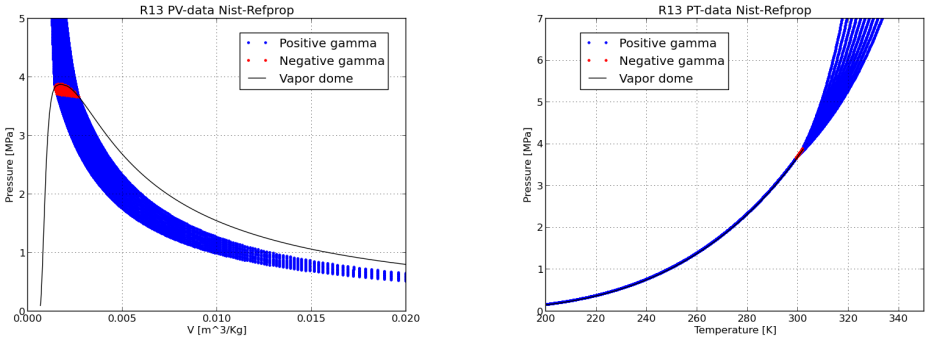
(a) PV-diagram (b) PT-diagram

Figure 5.5: Constant entropy lines; s=[1.3:0.1:1.6]



(a) PV-diagram (b) PT-diagram

Figure 5.6: Constant entropy lines; s=[1.1:0.1:1.6]



(a) PV-diagram (b) PT-diagram

Figure 5.7: Constant entropy lines; s=[1.17:0.01:1.26]

Chapter 6

The steam ejector

A lot of cooling and heating systems use refrigerants or non-natural working fluids such as chlorofluorocarbons (e.g. R-12) or hydrofluorocarbons (e.g. R134a). These fluids are very harmful for the environment. They can cause problems like depletion of the stratospheric ozone layer and produce large amounts of greenhouse gases. Furthermore most of these systems need electricity which strongly increases the demand for fossil fuels. Fossil fuels are limited and most of them harm the environment as well. A solution for these problems can be solar energy. Steam ejector refrigeration technology can be a part of a solar powered refrigeration system which makes this technology a very sustainable way for compressing fluids. However still a lot of research needs to be done on this technology.

The ejector itself is the crucial part of this system, it provides the compression effect for the cycle. At the University of Southern Queensland Ghassan Al-Doori has build an experimental rig to investigate the flow phenomena in the ejector. Furthermore mister Al-Doori is trying to do CFD simulations with Eilmer3 to check whether these results cope with the experimental results. However the results from the simulations do not yet agree with the experimental results. A possible cause of the error could be the used gas model, that will be investigated in this chapter. First a brief description of the Steam ejector will be given followed by the explanation of the experimental setup and the experimental results. The last part of this chapter is about the computational work, what is available so far and how are the results for different gas model.

6.1 Steam ejector

Figure 6.1 gives a schematic view of the steam ejector. The primary flow enters the ejector through the converging diverging nozzle. The flow accelerates through the nozzle to supersonic speeds which causes a low pressure at the end of the nozzle. This pressure is lower than the pressure in the evaporator therefore the vapor starts to flow into the steam ejector, this is called the secondary flow. The mixing process will start and that process will be finished at the beginning of the diffuser. When the flow reaches the diffuser the flow will be subsonic again, there will be a compression shock in the supersonic mixing flow. After the shock compression process the velocity will be reduced in the diffuser which will result in a further increase in pressure.

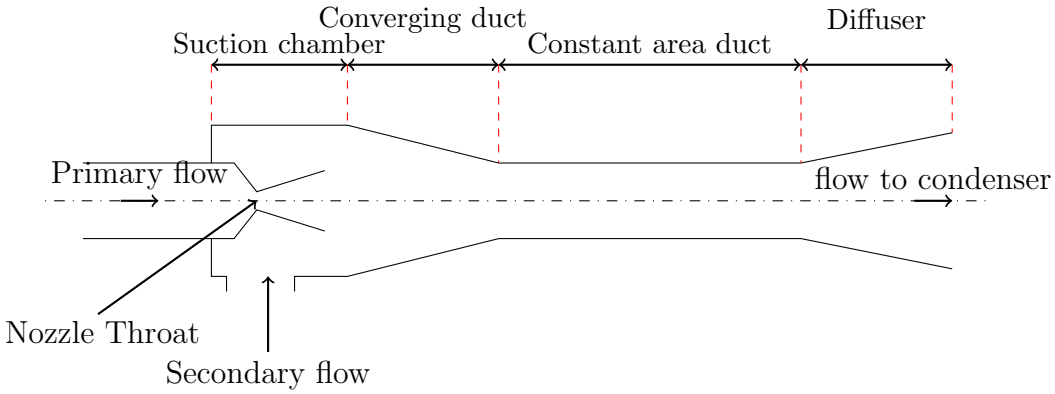


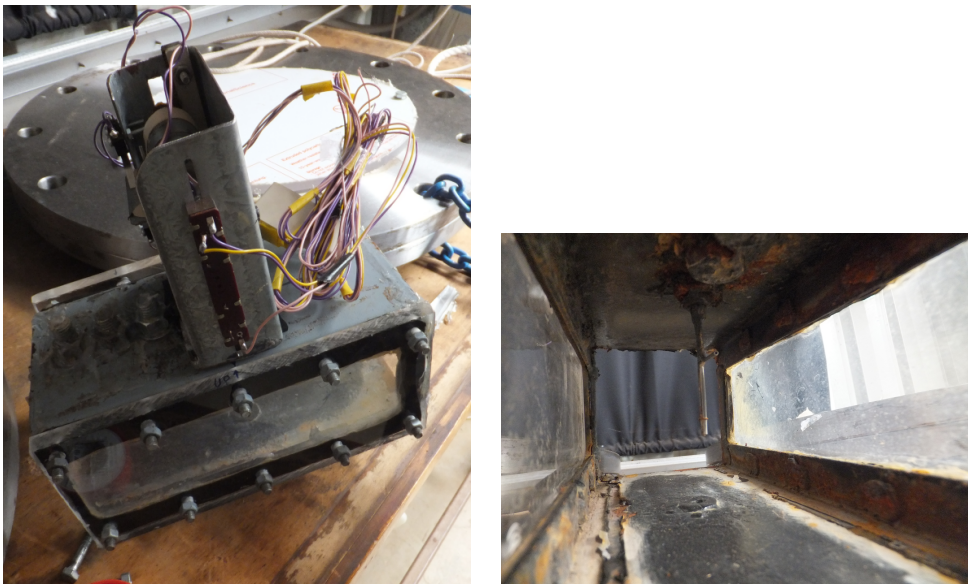
Figure 6.1: Schematic steam ejector

6.2 Experiments

As already stated Ghassan Al-Doori has done experiments with a steam ejector. His experimental rig is situated in Toowoomba in one of the test facilities of the Southern University of Queensland. The experimental rig he used is shown in figure 6.2. The experimental setup looks quite complicated therefore the different components will be explained. On the right side in the back the insulated boiler is situated, the boiler provides the steam for the setup. Through the insulated horizontal pipe the steam is transported to the nozzle of the steam ejector. After the steam is ejected through the nozzle the primary flow will induce the vapor from the tank, on the right, into the steam ejector. Where the mixed flow usually goes to a converging duct and a constant area duct has Al-Doori only used a constant area section.

Figure 6.2: The experimental setup from mister Al-Doori.

Figure 6.3 shows the rectangular duct which is used for the measurements. In the duct a pitot tube is placed, it can move in vertical direction and is actuated above the duct by an electro motor. Note that the pitot tube is extended downward in order to have the same blocking effect on the flow for each measurement position. The static pressure is measured in the side of the rectangular duct. In this way it is possible to measure the pitot pressure at different heights in a convenient way. When the measurements were performed there was no secondary flow.



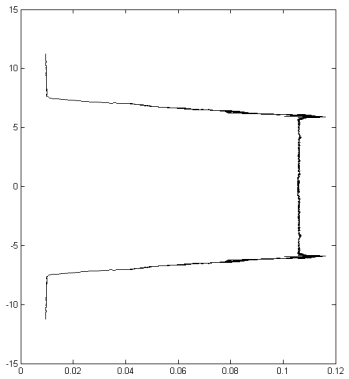
(a) The rectangular duct (b) The duct from the inside

Figure 6.3: The rectangular constant area section of the steam ejector

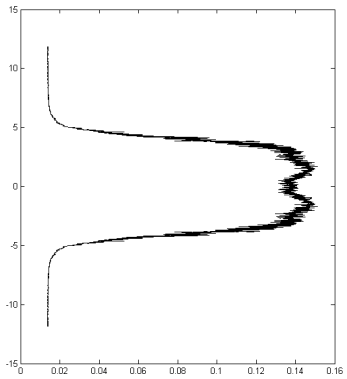
The flow conditions which occur in the flow are quite extreme. The steam enters the nozzle at 270 Kpa and 130 degrees Celsius, under these conditions the steam is only just superheated. The steam expands very rapidly through the nozzle and it reaches a very low pressure and temperature. At these conditions the steam should actually condense but because the steam is expanded so rapidly the steam has no time to condense. So the steam is not in equilibrium and is so called supersaturated. The fact that the steam is in non equilibrium gives a lot of difficulties with the CFD calculations as will be explained later on.

In one of the experiments the pressure ratio profile was measured. The

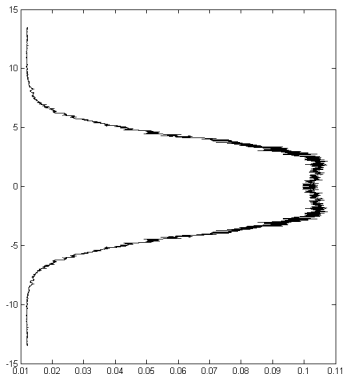
pressure ratio profile was measured at 1mm, 25mm 50mm and 70mm from the nozzle exit. At these positions the pitot tube was used to measure the pitot pressure for the entire height of the rectangular duct. As described in the experimental setup. The static pressure is measured by a hole in the side of the duct. The pressure ratio of the pitot pressure over the total pressure is shown in figure 6.4.



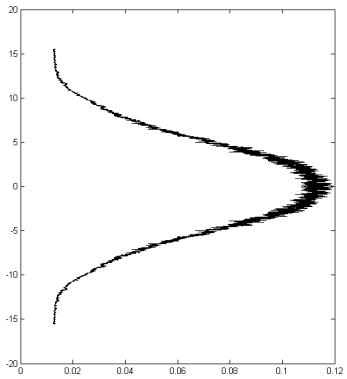
(a) 1mm after nozzle outlet



(b) 25mm after nozzle outlet



(c) 50mm after nozzle outlet



(d) 70mm after nozzle outlet

Figure 6.4: The pressure ratio $\frac{P}{P_0}$ from the experiments

6.3 Computational work done so far

As already mentioned mister Al-Doori has done some CFD simulations on the steam ejector in Eilmer3. The geometry used in this simulation is shown in figure 6.5 furthermore rotational symmetric flow is assumed to reduce the computational time. In order to be able to assume rotational symmetry the square duct is assumed to be cylindrical with a hydraulic diameter based on the square duct. This is quite a big assumption and the validity should be investigated in a later stadium.

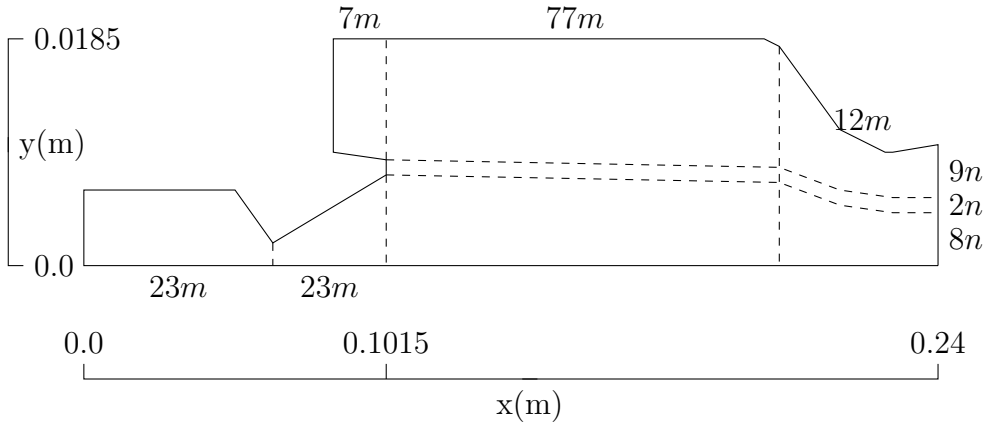
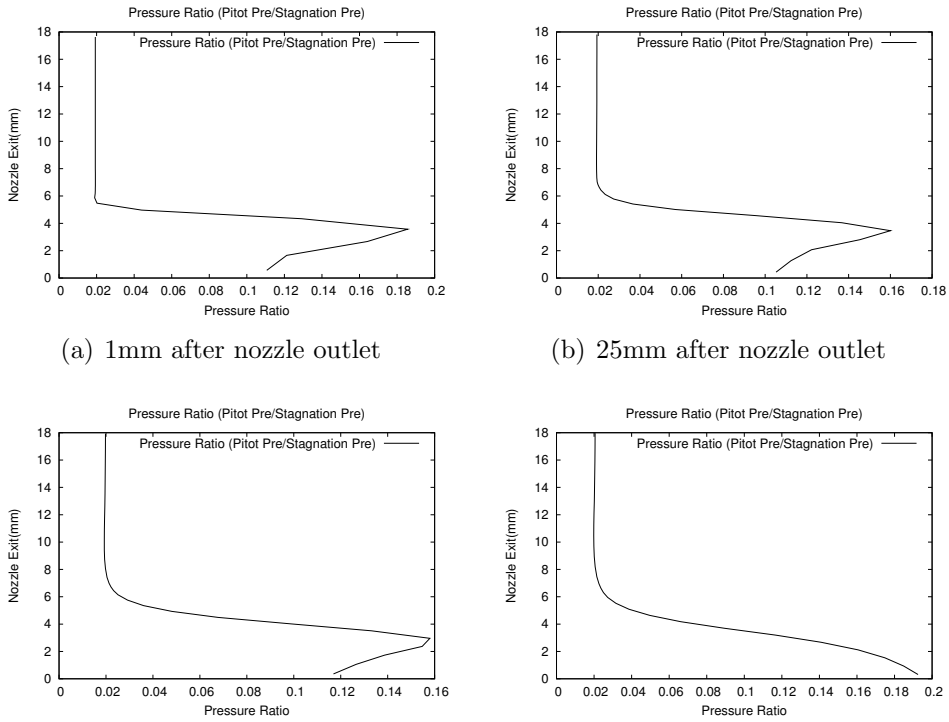


Figure 6.5: The grid used during the computation

The grid which is used in the simulation is as shown in figure 6.5 with n and m equal to 1.5. An ideal gas model is assumed for the steam with R=461.5 and γ=1.329 as the required constants. The python file which formulates the computation which is performed can be found in appendix A.1.

The results are shown in figure 6.6 and as one can see the results do not match with the experimental results. This might be due to the gas model, the incorrect geometry used in the computation or it can have another explanation. The gas is difficult to model due to the fact that is not in equilibrium, in the following part other gas models will be tried.



(a) 1mm after nozzle outlet

(b) 25mm after nozzle outlet

(c) 50mm after nozzle outlet

(d) 70mm after nozzle outlet

Figure 6.6: The pressure ratio from Al-Doori's simulation.

6.4 Computational work

Due to the fact that an incorrect gas model might be the cause for the incorrect results. Some other gas models are implemented in the python file. In this section the application of the different gas models, which are tried to get appropriate results from the CFD calculation, will be described. One acknowledgement should be made there is some doubt about the subsonic inflow boundary condition which is used in this simulation. This Boundary condition will be investigated further in chapter 7.

6.4.1 Nist Refprop gas model

The Nist Refprop gas model, which is one of the most accurate look up gas model available, was tried at first. However using this gas model the simulation could not finish. This is due to the fact that Nist Refprop represents the fluid during equilibrium for certain pressures and temperatures. At the pressures concerned in the steam ejector water is supposed to be liquid or maybe even ice, however during the experiments it is still a vapor. So Nist Refprop will give the quantities as if the fluid is in equilibrium at the concerned pressures and temperatures. As soon as the gas starts to expand unexpected results for e.g. the temperature occur and the simulation will stop. The following conclusion can be drawn: a Nist Refprop gas model can only be used for the conditions where Nist Refprop has actually the data for. Although there is quite a lot of data available for water no data has been found for the conditions in the steam ejector during operation.

6.4.2 Van der Waals gas

The Nist Refprop database could not cope with the still gassy steam while it actually should be condensing if it was in equilibrium. The Van der Waals gas model will be tried to use for the steam ejector simulations. In that way the steam will continue to stay gassy in because the van der Waals gas model does not include phase change. The van der Waals constants a and b are based on the Nist Refprop data near the vapor dome, because in that region Nist Refprop gives the most accurate data. A python program was written to compute the best a and b based on the data extracted from Nist Refprop. The van der Waals equation of state is shown in equation 6.1.

$$\left(p + \rho^2 a\right)\left(\frac{1}{\rho} - b\right) - RT = 0 \tag{6.1}$$

The program guesses the parameters a and b and computes the temperature with the van der Waals equation for a chosen patch of values for density and pressure. The error between the computed temperature and the temperature according to the Nist Refprop database is squared for each point in the patch and the errors are added together to get the squared error. This procedure is repeated for different values for a and b. Every time the squared error is lower than the lowest error recorded so far the values a,b and the lowest error are stored. In this way the best a and b will be found to fit the Nist Refprop data. The set of values which are tried for a and b are the elements from the rows A and B shown in expressions 6.2 and 6.3, they are tried in each combination. The grid for a and b was originally coarser, but was manually refined such that exacter results were obtained.

$$A = [0 : 10 : 10000] \tag{6.2}$$

$$B = [0.01 : 0.001 : 0.02] \tag{6.3}$$

The patch of Nist Refprop data, used for fitting the parameters a and b, is chosen around the initial conditions of the steam before it expands through the nozzle of the steam ejector. These conditions are $P_t = 2.76 \cdot 10^5\ Pa$ and $T_t = 417\ K$. The initial conditions together with the patch used to fit the van der Waals constants are shown in figure 6.7. The temperature of the steam will decrease rapidly when it goes through the steam ejector, so the data used to fit the van der Waals constants is quite close to the vapor dome. However in figure 6.7 this does not seem to be the case but in this figure a log-log scale is used, so in reality all values are fairly close to the vapor dome.

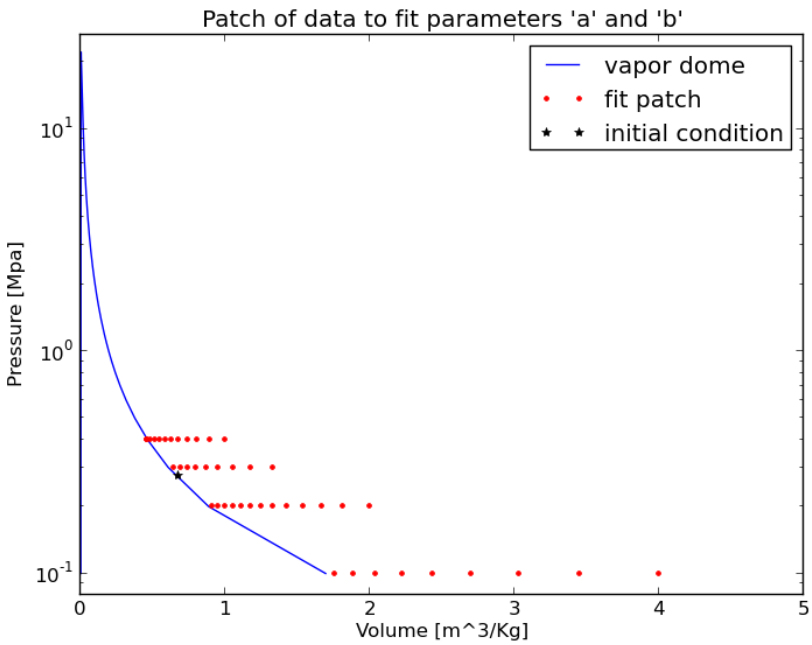
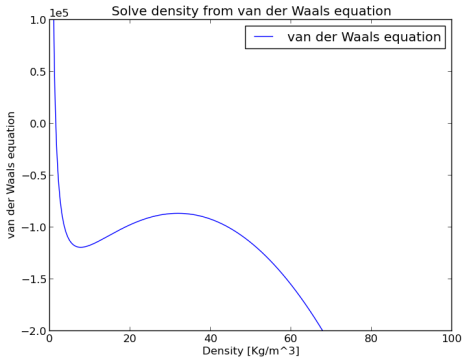


Figure 6.7: The data patch used from Nist Refprop to fit a van der Waals gas model.

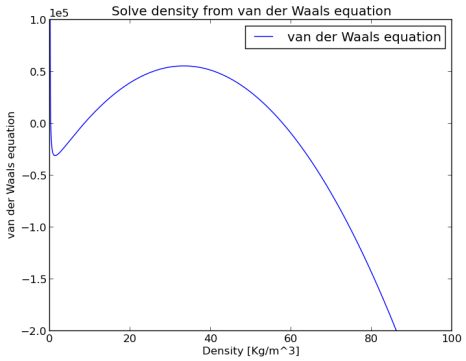
The best values for a and b are shown in table 6.1. When the flow is expanded the van der Waals equation becomes multivalued. The van der Waals equation is plotted in figure 6.8.

a	6100
b	0.015

Table 6.1: The van der Waals constants with the smallest squared error



(a) Initial state: $P = 276$ KPa, $T = 417$ K



(b) Expanded state: $P = 10$ KPa, $T = 100$ K

Figure 6.8: Van der Waals equation for best a and b

As one can see there are three values of the density which obey the van der Waals equation in this case are given in table 6.2. When the density is calculated with the ideal gas model for these conditions the density would be $\rho_{ideal} = 0.217$ Kg/m^3. So Eilmer3 should pick ρ_1 from the three different values for the density that obey the van der Waals equation. When the expanded conditions are given to Eilmer3 it indeed picks the right density of the three. By simply checking this by putting in the expanded conditions and reading out the density Eilmer3 indeed picks the correct value for the density. When the van der Waals gas model is used with Eilmer3 to do the

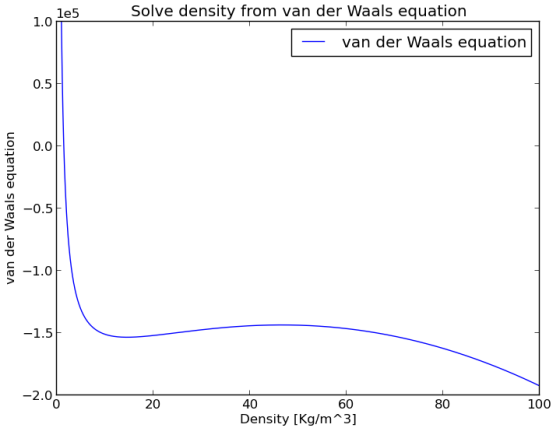
	density in $[kg/m^3]$
ρ_1	57.97
ρ_2	8.47
ρ_3	0.22

Table 6.2: The three density values obeying the van der Waals equation

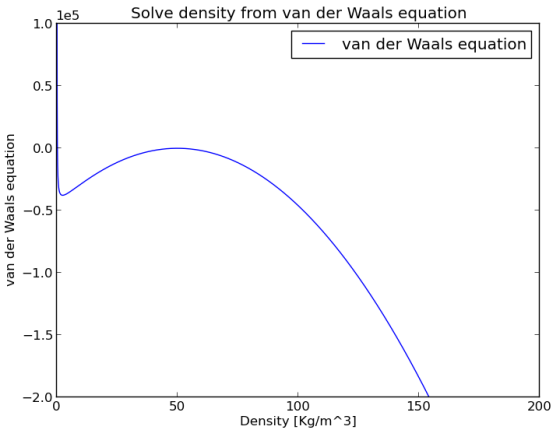
simulations for the steam ejector, the simulation stopped running at a certain point during the expansion. We think this is due to the fact that Eilmer3 has problems with the fact that the van der Waals equation is multivalued when the gas is in the expanded state. Especially when the point in the expansion is reached where the van der Waals model goes from a single valued solution to multivalued solutions is suspected to be the cause of the difficulties. To cope with this problem the program which determines a and b is changed. The program to determine the variables a and b from the van der Waals model is now slightly changed. It computes the derivative of the van der Waals model with respect to density, equation 6.4 and it uses the expanded conditions $T = 100\ K$ and $P = 10\ KPa$. The program looks where the derivative is zero because that are the local minima or maxima at these points the function value of the van der Waals equation will be computed. If both of these values have the same sign it will mean that the van der Waals equation has only a single valued solution for the density that obeys the equation. Again the a and b are chosen with the minimum squared error.

$$\frac{d}{d\rho}\left(\left(p + \rho^2 a\right)\left(\frac{1}{\rho} - b\right) - RT\right) = -\frac{p}{\rho^2} + a - 2\rho ab \tag{6.4}$$

However this formulation is not sufficient because it does not include the case when there is no local minimum or maximum and the derivative



(a) Initial state: $P = 276$ KPa, $T = 417$ K

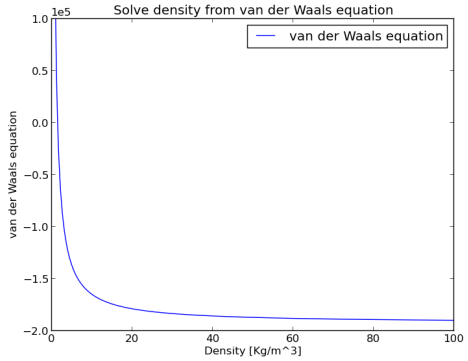


(b) Expanded state: $P = 10$ KPa, $T = 100$ K

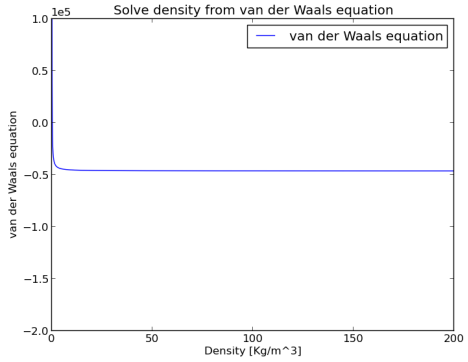
Figure 6.9: a and b best fitting the patch while the local minimum and maximum are both just beneath zero such that the equation has only a single solution.

is negative at all times. So the program is extended with the option of an always negative derivative because in that case the solution is single valued at all the times as well. So the program will look for the optimal a and b where one of these conditions is true, to make sure the van der Waals equation will be single valued. It turns out that the smallest error occurs for a and b

both equal to zero, meaning that the van der Waals equation turned to the ideal gas law 6.10. However when the Van der Waals gas model is used in Eilmer3 the constants a and b will be obtained through the critical pressure and temperature. When the critical values are chosen in such a way that a and b are supposed to be zero the Van der Waals gas model should give the same results as the ideal gas model, however this is not the case. Therefore the conclusion can be drawn that there is a bug somewhere in the van der Waals gas model. This will not be investigated further in this study but will be solved by Peter Jacobs or Rowan Gollan in time.



(a) Initial state: $P = 276$ KPa, $T = 417$ K

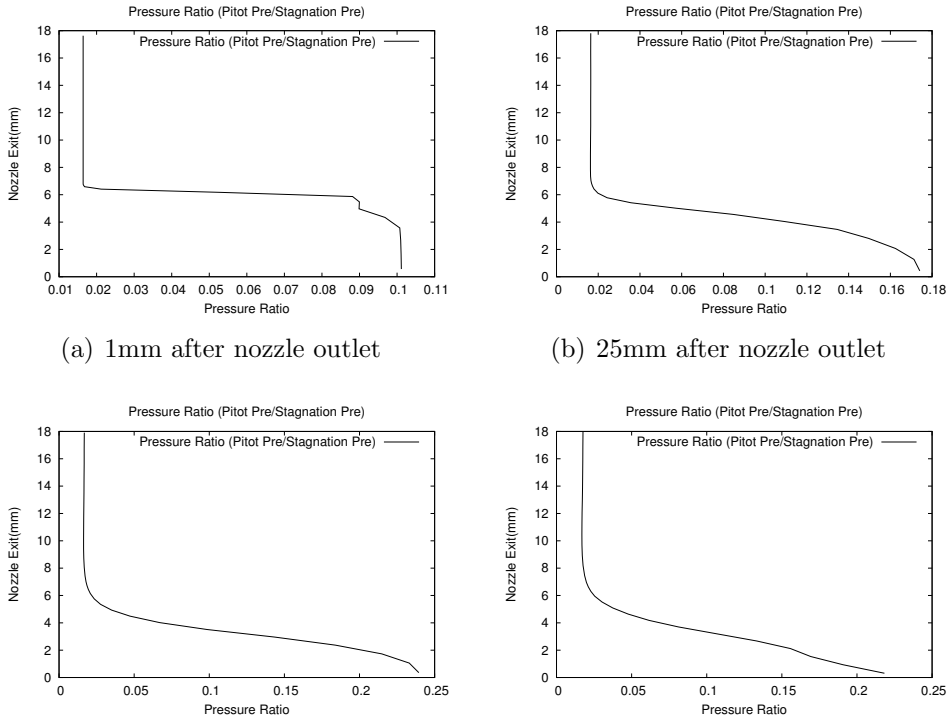


(b) Expanded state: $P = 10$ KPa, $T = 100$ K

Figure 6.10: When also including the case that the derivative is negative at all times the minimum misfit error occurs for a and b zero, which means it is the ideal gas law.

6.4.3 Adjusted ideal gas model

Dr Paul Petrie-Repar pointed out that an ideal gas model for supersaturated steam can be a proper way to simulate the gas when the polytropic index is adjusted. The lowering of γ corresponds to the absorbed energy through condensation, a γ of 1.11 is used in this simulation. The advantage of this gas model is that the computation does not encounter problems when the fluid actually should turn into liquid. However one should notice that by lowering γ something is done to simulate condensation but the real condensation is not modeled. The rest of the simulation is similar to the simulation performed by Ghassan Al-Doori. The simulation will run for 300 ms and for these cases the solution will be steady at that time. The results are shown in figure 6.11.



(a) 1mm after nozzle outlet

(b) 25mm after nozzle outlet

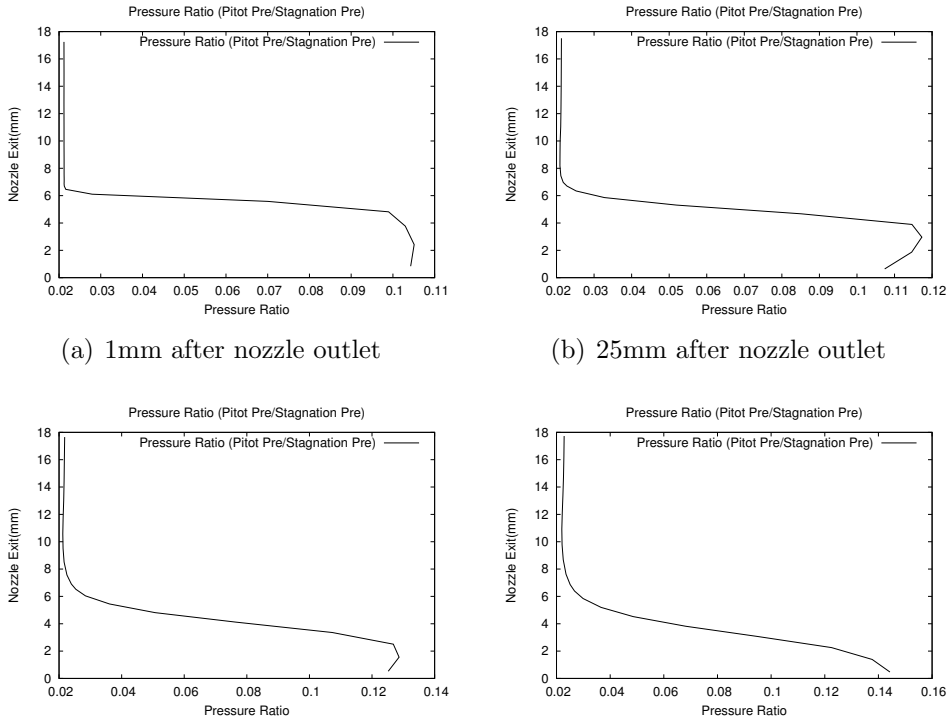
(c) 50mm after nozzle outlet

(d) 70mm after nozzle outlet

Figure 6.11: The pressure ratio from the simulation with n=1.5 and m=1.5 and an adjusted ideal gas model

The results shown by 6.11 are still not the same as the experimental

results. They also differ from Al-Doori his simulation and the shape seems to be a bit more similar to the experimental results. Another simulation has to be done with another grid size because the solution can not be depend on the grid size. So n and m are chosen to be equal to 1.0 to confirm the results. The grid size is chosen to be coarse to make sure that the simulation will be faster. The results from this simulation are shown in figure 6.12.



(a) 1mm after nozzle outlet

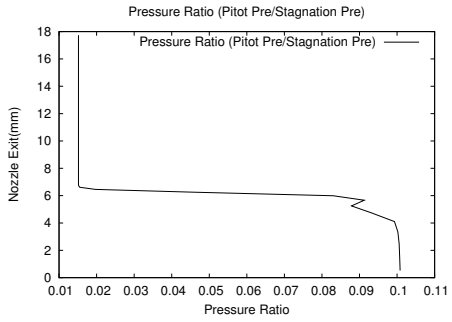
(b) 25mm after nozzle outlet

(c) 50mm after nozzle outlet

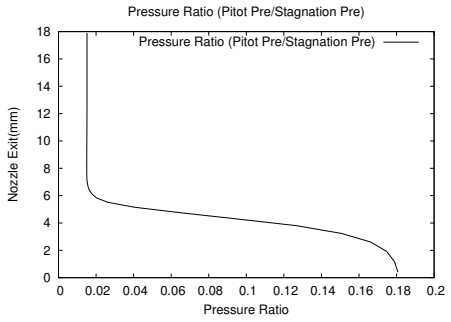
(d) 70mm after nozzle outlet

Figure 6.12: The pressure ratio from the simulation with n=1.0 and m=1.0 and an adjusted ideal gas model

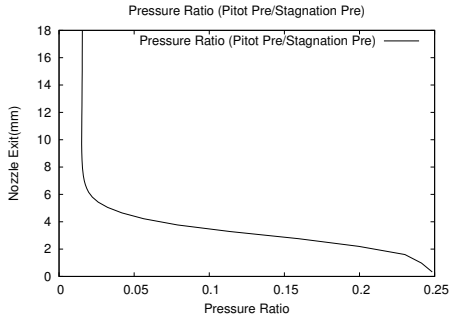
Surprisingly the results shown in figure 6.12 look very similar to the experimental results. The magnitude of the pressure ratio is as well closer to the magnitude which it is supposed to be. However the results are not similar to the simulation with the finer grid size where n and m equal 1.5. For both cases the solution was steady so the conclusion can be drawn that the solution is not yet converged. Therefore the simulation will be done with a finer grid n and m will equal 1.7. The simulation time does increase substantially and will take around 6 days on the available super computer from the university of Queensland. The results are shown in figure 6.13.



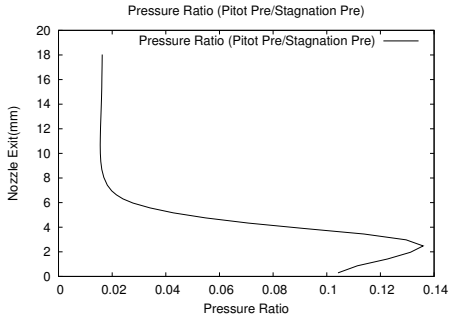
(a) 1mm after nozzle outlet



(b) 25mm after nozzle outlet



(c) 50mm after nozzle outlet



(d) 70mm after nozzle outlet

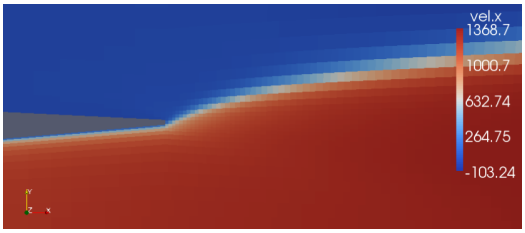
Figure 6.13: The results from Simulation with n=1.7

The results for 1mm,25mm and 50mm look very similar to the solution with n and m=1.5, the magnitude of the pressure ratio is also very close to each other. However 70mm shows truly different behavior one thing is quite unexpected that is the magnitude of the pressure ratio, it seems to be quite small compared to figure 6.11. But the order of magnitude is quite close to the experimental data.

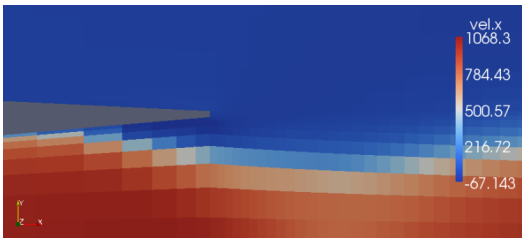
6.5 Flow around the nozzle exit of the steam ejector

With Paraview the simulation with the adjusted ideal gas model for n and m equal to 1.7 is visualized. Shortly after the primary flow enters the steam ejector a snapshot is taken and shown in figure 6.14(a). When the solution

is completely steady so at t= 300ms another snapshot is taken and shown in figure 6.14(b). As in figure 6.14 can be seen the flow is in the beginning under



(a) t=4ms



(b) t=300ms

Figure 6.14: Snapshots of the simulation at the nozzle exit of the steam ejector.

expanded which is not surprisingly because there is a very large pressure difference. However according to Eilmer3 the solution becomes over expanded if this in reality indeed is the case in reality the nozzle will be quite efficient but the jet can be unstable which can cause damage to the nozzle. It will be difficult to see whether the nozzle is over expanded during the experiments but it might be worth further investigation.

Chapter 7

The subsonic inflow boundary condition

Due to the fact that the CFD calculation still does not agree with the experimental results the model should be investigated further.One of the boundary conditions is the subsonic inflow boundary condition at the begin of the primary flow. The functioning of this boundary condition is questioned. To check whether it functions properly two checks will be executed. The first check will be to see if the mass flow throughout a nozzle will remain the same. The second check will investigate if the total pressure and total temperature at the subsonic inflow boundary correspond to the specified total pressure and temperature.

7.1 The massflow

To see whether the mass flow trough the subsonic inflow BC remains is the same as in the rest of the flow domain a simple example is used. The nozzle as shown in figure 7.1 is used for this check. On the left side of the subsonic region at x=-0.26 the inflow BC is located. At x=-0.26, x=-0.125 and just before and after the throat is the data recorded and at those position the mass flow will be computed. The data for all nodes at a constant x-position is obtained and integrated along the surface. The results are shown in table 7.1. The following conclusion can be drawn: when the solution is steady the mass flow is constant trough the nozzle. The small differences are due to numerical errors.

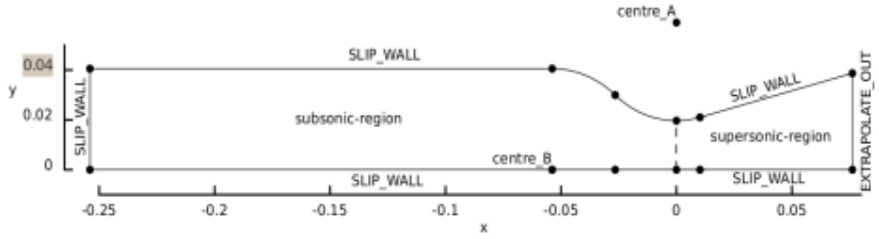


Figure 7.1: Schematic view of the Nozzle

Table 7.1: massflow check

x-position in (m)	Mass flow in (m^3/s)
x=-0.26	0.4916
x=-0.125	0.4916
x=-0.0007	0.4901
x=0.002	0.4913

7.2 Total pressure and total temperature

In the subsonic inflow boundary condition it self the flow properties are computed by using the initial conditions. The check that is performed is checking whether these calculated flow properties indeed correspond to the correct total pressure and temperature. Because the total pressure and temperature are the given quantities. This is done by extracting the data from the simulation for the first cells of the first block, so at the inflow. This data is used to compute the velocity, total pressure and total temperature at each node. This is done through the following equations:

$$|v|^2 = v_x^2 + v_y^2 + v_z^2 \tag{7.1}$$

$$P_t = p + \frac{1}{2}\rho|v|^2 \tag{7.2}$$

$$T_t = T + \frac{|v|^2}{2C_p} \tag{7.3}$$

As inflow conditions the total pressure and temperature are set at $2.76 \cdot$

Table 7.2: Total pressure and temperature

node	Total pressure (in Pa)	Total temperature (in K)	Velocity (in m/s)
1	244373	416.605	27.73
2	244372	416.605	27.73
3	244372	416.605	27.73
4	244373	416.605	27.74
5	244374	416.605	27.74
6	244375	416.605	27.75
7	244376	416.606	27.76
8	244372	416.616	27.76
9	244375	416.631	27.65
10	244374	416.404	27.49
11	244383	414.291	28.03
12	244511	405.805	29.58
13	244588	362.811	32.19

10^5 Pa and $417K$ respectively. However the calculated values for the total pressure are quite off, they are around $2.44 \cdot 10^5$ Pa while the total temperature is $416K$. The exact values are given in table 7.2, where node 1 is in the center of the flow and node 13 is at the wall. As one can see the total pressure seems to be rather constant however the value is incorrect. The total temperature is quite close to the actual total temperature and the reason that the total temperature decreases close to the wall is due to the fact that the wall temperature is fixed at 308 K. However these results show some strange behavior regarding the velocity, because the velocity does not decrease near to the wall. And this should be the case because there is a no slip condition at the boundary. The subsonic inflow boundary condition is recently changed in Eilmer3 so therefore the results from Ghassan's simulation, which have used the old subsonic inflow BC are checked. The same quantities are calculated and the old subsonic boundary condition seems to have better results, as can be seen in table 7.2. The total pressure is almost correct. The total temperature and the velocity decreases for the nodes closer to the wall. So the conclusion can be drawn that the Subsonic inflow BC is not working properly yet for the updated version of Eilmer3.

Table 7.3: Total pressure and temperature

node	Total pressure (in Pa)	Total temperature (in K)	Velocity (in m/s)
1	275937	416.636	31.12
2	275937	416.636	31.13
3	275937	416.636	31.12
4	275936	416.636	31.11
5	275939	416.635	31.17
6	275943	416.633	31.28
7	275926	416.641	30.89
8	275868	416.663	29.58
9	275839	416.451	28.87
10	275716	413.514	26.11
11	275544	400.812	19.74
12	275284	363.591	7.59

7.3 Subsonic inflow BC based on ideal gas laws

In order to do simulations this problem needs to be solved. At the Univeristy of Queensland Rowan Gollan will change the way the boundary condition is computed and implement this in the code. However this will take some time and therefore we will continue to work with a subsonic inflow boundary condition based on the ideal gas laws. What this boundary condition does is it obtains the inflow velocity through the conservation equations, based on the initial difference in pressure. With this velocity it computes the other thermodynamic quataties such as: speed of sound a, both static and total temperature T, Mach number M, both static and total pressure p, the density ρ and the internal energy e. All these quantaties are calculated through formula's derived from the ideal gas law. The new quantaties will be used to compute a new inflow velocity and the thermodynamic quantaties will be updated again. This will be done untill the solution is steady and the inflow velocity and thermodynamic quantaties do not change anymore. The assumptions made by using this boundary condition are the same as for using an ideal gas model. Due to the fact that the simulations of the steam ejector use an ideal gas model this boundary condition can be used with-

out compromising the accuracy further. In addition the boundary condition which R.Gollan will implement will work in a similar fashion however with isentropic laws instead of ideal gas laws.

7.4 The outflow boundary condition

The inflow boundary condition is already discussed but how does the outflow boundary condition work and does it work properly. The outflow boundary condition is an extrapolation boundary condtion, which means that the value within the ghost cells will be estimated based on the values of the cells within the boundary condition. For the steam ejector simulation the outflow requirement is that the outflow should be supersonic. Because if the outflow is supersonic no information can travel from upstream of the exit into the steam ejector. Figure 7.2 shows the speed of sound at the exit and figure 7.3 shows the velocity in x-direction at the exit. So as one can see the flow is supersonic everywhere except at the wall. This is clarified by the boundary layer due to the no slip condition. The conclusion can be drawn that the outflow boundary condition works as it is supposed to do. Figure 7.3 shows the exit velocity in x-direction for the simulation with the adjusted ideal gas model with a grid with n and m equal to 1.5. The outflow boundary condition is checked for all the simulations and seems to work for all the simulations.

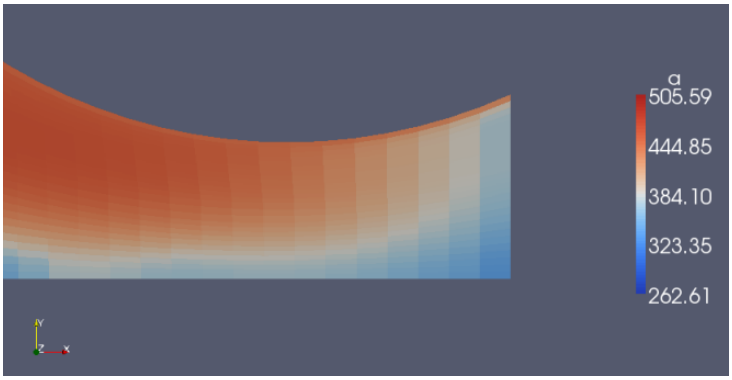


Figure 7.2: The speed of sound at the exit of the steam ejector

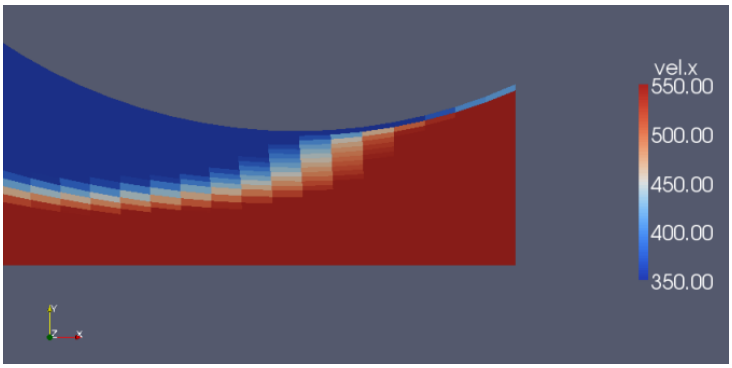


Figure 7.3: The velocity in the x-direction at the exit of the steam ejector

Chapter 8

Boundary layer in the nozzle of the steam ejector.

In section 6.4.3 the results of the simulation with the adjusted ideal gas model are shown. As one can see the results differ quite a lot when the grid size changes. A possible explanation for this phenomenon can be the turbulence model. The turbulence model may overestimate the effect of viscous dissipation which results in a boundary layer which is to thick. Furthermore Al-Doori has done some more simulations on the steam ejector and he is encountering big differences directly behind the nozzle already. Therefore the nozzle of the steam ejector will be simulated and the influence of the turbulence model will be investigated in this section.

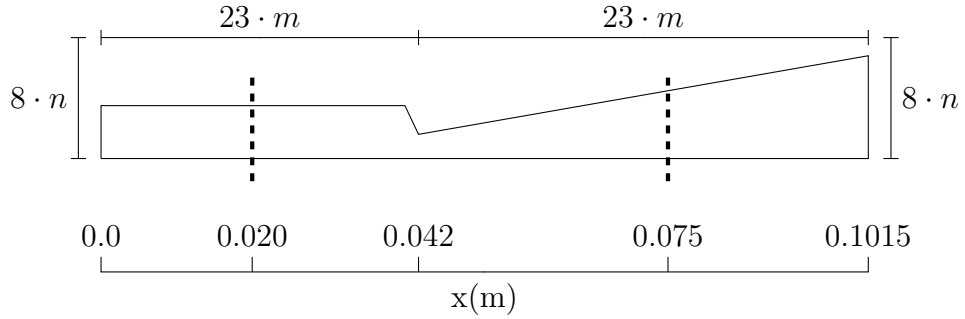


Figure 8.1: Schematic drawing of the steam ejector nozzle

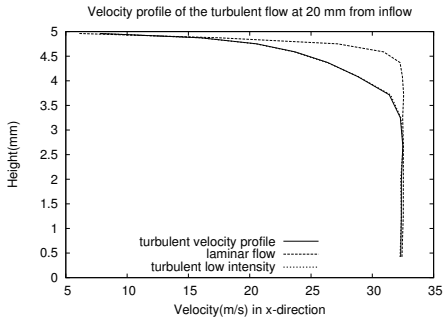
8.1 Simulation of the nozzle

For the simulation of the steam ejector a k-ω turbulence model is used. Sometimes the effect of viscous dissipation is overestimated by the turbulence model which results in a thick boundary layer. To check whether the boundary layer is indeed too thick two simulations will be done: one with a k-ω turbulence model and one without a turbulence model so a laminar flow situation. for both simulation the velocity profile will be obtained at two locations: one before the throat at x=0.02m and one after the throat at x=0.075m. Furthermore to investigate the effect of the boundary layer the total pressure and total temperature will be obtained along the centerline and close to the wall. Close to the wall means in this case the second node besides the wall in y direction. Figure 8 is schematic representation of the steam ejector nozzle, underneath the figure the dimensions are given in meters. Apart from that the refinement of the grid is given by $8 \cdot n, 23 \cdot m$ where different values for n and m will be used. The steam will enter the nozzle at the same conditions as it would enter the steam ejector so:$P = 2.76e5, T = 417$. All the simulations done in this study will run for 3ms to make sure that the solution is steady. There is a subsonic inflow boundary condition based on the ideal gas laws and the outflow boundary condition is an extrapolate out. Furthermore there is a no slip condition at the wall and the temperature is fixed at the wall at 308 K. Before any conclusions will be drawn from the simulations the subsonic inflow boundary condition will be checked and for all simulations it turned out to work fine.

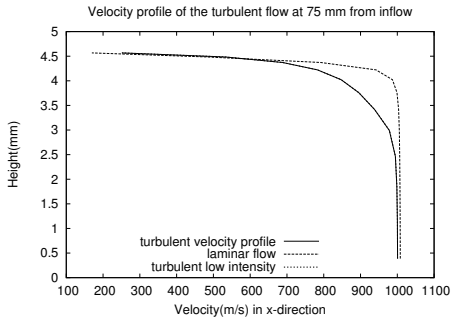
8.2 Boundary layer thickness in the nozzle

The first simulation is done with n and m equal to 1.5 for both the laminar and turbulent case. Figure 8.2(a) and 8.2(b) show the velocity profiles for the laminar and turbulent case at 20 and 75 mm. In appendix C.1 the bash script is shown which produces figure 8.2(a) and 8.2(b) similar scripts are often used during this study for different figures. While figure 8.2(d) and 8.2(c) shows respectively the total temperature and temperature at the centerline and close to the wall.

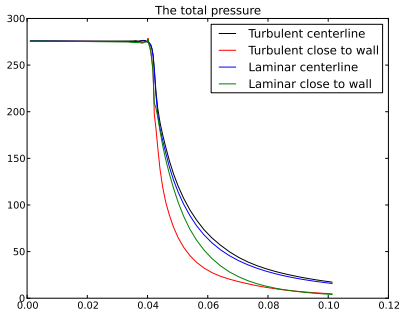
By definition the boundary layer ends if the velocity is $0.99 \cdot U_{mean}$ and for both before and after the nozzle the turbulent boundary layer is much thicker than the laminar boundary layer. Figure 8.2(c) shows that the difference in



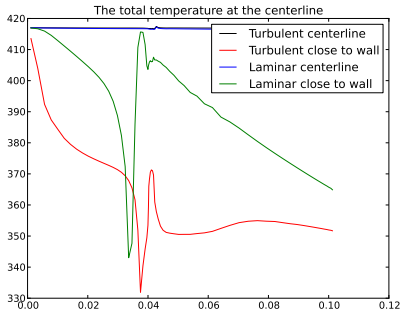
(a) velocity profile at 20mm



(b) velocity profile at 75mm



(c) The total pressure



(d) The total temperature

Figure 8.2: Results form the nozzle simulation with n,m=1.5 at t=3ms

total pressure at the centerline and close to the wall for the laminar case are more similar from which the conclusion can be drawn that in the turbulent case the boundary layer has influence on a bigger part of the flow. The graph of the total temperature along the wall shows a decrease in total temperature close to the throat due to the fact that the wall temperature is 308 K which causes the static temperature to drop. Then the flow starts to accelerate fast which causes the total temperature to rise again. For the turbulent case this behavior goes a bit smoother which is caused by the fact that in turbulent flow there is more heat transfer because the flow mixes more. Figure 8.2(a) and 8.2(b) give an idea of the boundary layer thickness according to Eilmer3, to give some kind of idea on the order of magnitude of the boundary layer the boundary layer is estimated for a flat plate. Which is off course a incorrect assumption but it tells us something about the order of magnitude, is the

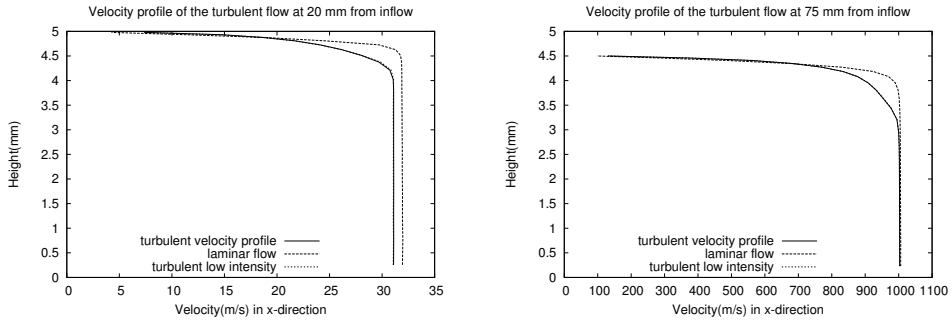
boundary layer centimeters or micrometers thick. The simple equations of Blasius for the boundary layer on a flat plate are used which are shown in equation 8.1. Where δ is the boundary layer thickness for the laminar case and δ_t for the turbulent case. All the quantities such as: ν, u_0, ρ, μ are averaged on the free stream cells, the results are in 8.2.

$$\delta = 4.91 \cdot \sqrt{\frac{\nu x}{u_0}} \qquad \delta_t = 0.382 \cdot x \cdot \left(\frac{\rho u_0 x}{\mu}\right)^{\frac{1}{5}} \tag{8.1}$$

	20 mm	75 mm
δ in (mm)	0.3972	0.3349
δ_t in (mm)	0.4028	0.3452

8.2.1 Grid size and initial conditions

In figure 8.2 one can see that the boundary layer thickness for the laminar case has the same order of magnitude as the computed value by Blasius equation. The turbulent boundary layer seems to be quite thick and therefore it can be concluded that the viscous effects are probably overestimated by the turbulence model. This overestimation by the turbulence model can be caused by to large initial conditions or a grid size which is too coarse. Therefore two things are done the initial conditions are changed to check whether this makes a difference, by initial conditions the initial turbulence intensity and the laminar to turbulent viscosity ratio are meant. A lot of different values are tried and the case for which both initial conditions are divided by 20 is called the low turbulent intensity case. In figure 8.2(a) and 8.2(b) this case is also plotted and as one can see it hardly differs from the turbulent case. Different values for n and m are also used to refine the grid and it seems that when the grid is finer the turbulent boundary layer becomes smaller. Figure 8.3 shows the velocity profiles for the finest grid with n and m equal to 2.5. It is expected that grid refinement can be a solution to eliminate the overestimation of the viscous effects. However due to a lack of time no finer grids are researched a simulation will take at least four days and there is no time for that.



(a) velocity profile at 20mm (b) velocity profile at 75mm

Figure 8.3: Results form the nozzle simulation with n,m=2.5 at t=3ms

Chapter 9

Conclusion

During this internship several subjects have been addressed in this section the most important conclusions will be summarized.

- In general the conclusion can be drawn that the van der waals gas model and the Nist Refprop gas model work in Eilmer3 under certain conditions. The gas should be in equilibrium to obtain realistic results. Furthermore when the Nist Refprop gas model is concerned the simulation should be in the domain where the Nist Refprop database has data available. If this is not the case the simulation will not converge.

- The simulations of the steam ejector still do not cope with the experimental results. An adjusted gas model is the best gas model for the simulation. In order to come up with better results the turbulence model should be alternated to make sure the turbulent boundary layer has the appropriate thickness.

- The geometry of the steam ejector is a rectangular duct, however it is modeled as a circular duct in order to have rotational symmetry. By making this assumption the amount of cells required to do a simulation is much smaller. Although it can alternate the solution quite a bit. Due to a lack of time the computational geometry is not adjusted.

- Non classical gas behavior does not seem to occur in Eilmer3. However in literature the existence of this behavior is still doubted.

- The subsonic inflow boundary condition did not work properly and this will be fixed by Rowan Gollan an employee of the University of Queensland. However this happened after this internship ended so it is not used for new simulations.

Bibliography

[1] B.P.Brown and B.M.Argrow, *Two-dimensional shock tube flow for dense gases*, Department of Aerospace Engineering Science, University of Colorado, Boulder, 1997.

[2] B.M.Argrow, *Computational analysis of dense gas shock tube flow*, Department of Aerospace Engineering Science, University of Colorado, Boulder, 1995.

[3] A.C. Aldo, B.M.Argrow (1995) *Dense gas flow in minimum length nozzles*, Journal of Fluid Engineering 117:270

[4] A.A. Borisov, A.I.A. Borisov, S.S. Kutateladze, V.E. Nakoryakov (1981) *Rarefaction shock wave near the critical liquid-vapour point*, Journal of Fluid Mechanics 126:59

[5] P.A. Thompson (1971) *A fundamental derivative in gasdynamics*, Physics of Fluids 14:1843

[6] P.A. Thompson, K.C. Lambrakis (1973) *Negative shock waves*, Journal of Fluid Mechanics 60:187

[7] P.A. Jacobs, R.J. Gollan (2008/07) *The Eilmer3 Code: User Guide and Example Book*, Mechanical Engingeering Report 2008/07

[8] P.A. Jacobs, R.J. Gollan, A.J. Denman, B.T. O'Flaherty, D.F. Potter, P.J. Petrei-Repar and I.A. Johnston (2010/09) *Eilmer's Theory Book: Basic Models for Gas Dynamics and Thermochemistry*, Mechanical Engingeering Report 2010/09

Appendix A

A.1 Nozzle.py

```
# Actual Steam nozzle_test.py
# 28 Aug 2012
# Water as a working fluid.
# Input SubsonicInBC(inflow)
# Output is ExtrapolateOutBC()

gdata.title = "Flow through an ejector."
print gdata.title

# Parameters that we are likely to change
m =1.5; n = 1.5 # cell refinement control

nx0 = int(23*m); nx1 = int(33*m); nx2 = int(77*m);nx3 = int(7*m)
    ;nx6 = int(12*m);

ny0 = int(8*n); ny1 = int(2*n); ny2 = int(9*n)

# Define X locations of each pressure transeducer at roof of the
    rectangular duct
xt1= 0.101
xt2= 0.2035

# Accept defaults for water - treated as ideal gas for now
select_gas_model(model='ideal gas', species=['H2O'])
#select_gas_model(model='ideal gas', species=['air'])
#select_gas_model(fname="Water_200_LUT.lua.gz")

#==========specifiy the boundary conditions==========

```

```
#----------initial flow-------
p_ini= 500.0
u_ini= 10.0
v_ini= 0.0
T_ini= 283.0
Itub_ini= 0.002  #turbulence intensity for initial flow
tur_lam_ratio_ini =  1.0 #(Mt/Ml)turbulent-to-laminar viscosity
    ratio for initial flow
rho_ini = p_ini /(461.5 * T_ini)  # Uses R from steam

#----------secondary flow--------
p_sec= 3000.0
u_sec= 20.0
T_sec= 293.0
Itub_sec= 0.02  #turbulence intensity for secondary flow
tur_lam_ratio_sec =  1000.0  #(Mt/Ml)turbulent-to-laminar
    viscosity ratio secondary flow
rho_sec = p_sec /(461.5 * T_sec)  # Uses R from steam

#----------Primary flow----------

p_pri= 2.76e5
u_pri= 30.0
T_pri= 417.0
Itub_pri= 0.01  #turbulence intensity for primary flow
tur_lam_ratio_pri =  1000.0  #(Mt/Ml)turbulent-to-laminar
    viscosity ratio primary flow
rho_pri = p_pri/(461.5 * T_pri)  # Uses R from steam


# Sutherland's viscosity law
S= 1064.0 ; T_ref= 350.0; mu_ref=1.12e-5 #steam sutherland's
    constant from visocus fuild flow pp29 White(1991)


#========= Estimate turbulence intensity and turbulent-to-
    laminar viscosity ratio==========

# firstly initial flow
Sutherland_mu_ini= mu_ref * (T_ini / T_ref)**1.5 * (T_ref + S)/(
    T_ini + S)#secondary Sutherland's viscousity
# turbulent kinetic energy law = 1.5 *(Itub *u)**2  where :Itub=
     turbulence intensity & u = velocity
tke_ini = 1.5 * (Itub_ini * u_ini)**2 # turbulent kinetic energy
     for primary flow
```

```
omega_ini = rho_ini * tke_ini /(tur_lam_ratio_ini * Sutherland_mu_ini)

#secondaly secondary flow
Sutherland_mu_sec= mu_ref * (T_sec / T_ref)**1.5 * (T_ref + S)/(T_sec + S)#secondary Sutherland's viscousity
# turbulent kinetic energy law = 1.5 *(Itub *u)**2  where :Itub= turbulence intensity & u = velocity
tke_sec = 1.5 * (Itub_sec * u_sec)**2 #turbulent kinetic energy for secondary flow
omega_sec = rho_sec * tke_sec /(tur_lam_ratio_sec * Sutherland_mu_sec)

# finally primary flow
Sutherland_mu_pri= mu_ref * (T_pri / T_ref)**1.5 * (T_ref + S)/(T_pri + S)# primary Sutherland's viscousityNone,
# turbulent kinetic energy law = 1.5 *(Itub *u)**2  where :Itub= turbulence intensity & u = velocity
tke_pri = 1.5 * (Itub_pri * u_pri)**2 # turbulent kinetic energy for primary flow
omega_pri = rho_pri * tke_pri / (tur_lam_ratio_pri * Sutherland_mu_pri)

print "%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%"
print "initial Sutherland viscousity = ",Sutherland_mu_ini
print "secondary Sutherland viscousity = ",Sutherland_mu_sec
print "primary Sutherland viscousity = ",Sutherland_mu_pri
print "%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%"

print " density of initial flow",rho_ini
print " density of secondary flow",rho_sec
print " density of primary flow",rho_pri
print "%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%"

print " turbulent kinetic energy for initial flow",tke_ini
print " turbulent kinetic energy for secondary flow",tke_sec
print " turbulent kinetic energy for primary flow",tke_pri
print "%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%"

print "initial omega = ",omega_ini
print "secondary omega = ",omega_sec
print "primary omega = ",omega_pri

#==============Initial and inflow conditions===============

```

```
#initial = ExistingSolution('Nozzle', '.', 6, 43)
initial = FlowCondition(p=p_ini,u=u_ini, v=v_ini, T=T_ini, tke=
    tke_ini, omega=omega_ini)# initial conditions
inflow_sec = FlowCondition(p=p_sec, T=T_sec,tke= tke_sec, omega=
    omega_sec)  # secondary conditions
inflow_pri = FlowCondition(p=p_pri, T=T_pri,tke= tke_pri, omega=
    omega_pri)   #primary conditions
# The following Nodes will be rendered in the SVG file.

# Node group A
a0 = Node(0.0, 0.0)
a1 = Node(0.042, 0.0)
a2 = Node(0.1015, 0.0)
a3 = Node(0.21, 0.0)
a4 = Node(0.24, 0.0)


# Nodes group B
b0 = Node( 0.0000000, 0.005000,label="B0")
b1 = Node(0.03855000, 0.005000,label="B1") # First point
b2 = Node(0.04020000, 0.0035000,label="B2")# Second point
centre_bc1= Node(0.03855000, 0.00335000, label="centre_B1")#
    Centre curve1
b3 = Node(0.042000, 0.0016000) # Second point (Throat of the
    primary nozzle)
centre_bc2= Node(0.04200052, 0.00340124, label="centre_B2")#
    Centre curve2
b4 = Node(0.1015, 0.0068)
b5 = Node(0.21, 0.0058)
b6 = Node(0.22, 0.0035)
b7 = Node(0.24, 0.0047)

# Nodes group C

#c0a= Node(0.0148, 0.010)
#c0b= Node(0.0268, 0.010)
c1 = Node(0.0895, 0.00768955)
c2 = Node(0.1015, 0.0071)
c3 = Node(0.21, 0.0071)
c4 = Node(0.22, 0.0047)
c5 = Node(0.24, 0.0055)


# Node group D
#d0 = Node(0.0148, 0.0185)
```

```
d1 = Node(0.0895, 0.0185)
d2 = Node(0.1015, 0.0185)
d3 = Node(0.205, 0.0185) # First point
centre_dc = Node(0.205, 0.0135)# Centreline point
d4 = Node(0.21, 0.0135)  # Second point
d5 = Node(0.22, 0.0085)
d6 = Node(0.24, 0.01)



# Block-0
north0 = north0 = Polyline([Line(b0, b1), Arc(b1,b2,centre_bc1),
    Arc(b2,b3,centre_bc2)])
east0 = Line(a1, b3)
south0 =Line(a0, a1)
west0 = Line(a0, b0)


# Block-1
north1 = Line(b3, b4)
east1 = Line(a2, b4)
south1 = Line(a1, a2)


# Block-2
north2 = Line(d1, d2)
east2 = Line(c2, d2)
south2 = Line(c1, c2)
west2 = Line(c1, d1)


# Block-3
north3 = Line( b4, b5)
east3 = Line( a3, b5)
south3 = Line(a2, a3)

# Block-4
north4 = Line(c2, c3)
east4 = Line(b5,c3)
west4 =Line(b4,c2)

# Block-5
north5 = Polyline([Line(d2, d3), Arc(d3,d4,centre_dc)])
east5 = Line(c3, d4)

```

```
184
185 # Block 6
186 north6 = Arc3(b5, b6, b7)
187 east6 = Line(a4, b7)
188 south6 = Line(a3, a4)
189
190 # Block 7
191 north7 = Arc3(c3, c4, c5)
192 east7 = Line(b7, c5)
193
194 # Block 8
195 north8 = Arc3(d4, d5, d6)
196 east8 = Line(c5,d6)
197
198
199
200
201
202
203 # Define the blocks, boundary conditions and set the
      discretisation.
204
205 # Inlet Nozzle ( primary flow)
206 blk0 = Block2D(make_patch(north0, east0, south0, west0),
207                           nni=nx0, nnj=ny0,
208                           fill_condition=initial,
209                           label="block0",
210                           cf_list=[RobertsClusterFunction(0, 1,
                                1.08),
211                                    RobertsClusterFunction(0, 1,
                                        1.04),
212                                    RobertsClusterFunction(0, 1,
                                        1.08),
213                                    RobertsClusterFunction(0, 1,
                                        1.04)],
214                           bc_list=[FixedTBC(308), SlipWallBC(),
215                           SlipWallBC(), SubsonicInBC(inflow_pri)
                                ],
216                           hcell_list=[(1,1)])
217
218
219 # Outlet Nozzle
220 blk1 = Block2D(make_patch(north1, east1, south1, east0),
221                           nni=nx0, nnj=ny0,
222                           fill_condition=initial,
```

```
223                                 label="block1",
224                                 cf_list=[RobertsClusterFunction(1, 1,
                                        1.08),
225                                         RobertsClusterFunction(0, 1,
                                            1.04),
226                                         RobertsClusterFunction(1, 1,
                                            1.08),
227                                         RobertsClusterFunction(0, 1,
                                            1.04)],
228                                 bc_list=[FixedTBC(308), SlipWallBC(),
229                                 SlipWallBC(), SlipWallBC()],
230                                 hcell_list=[(1,1)])
231
232 # Mixing chamber1 (No Secondary flow)
233
234 blk2 = Block2D(make_patch(north2, east2, south2, west2),
235                                 nni=nx3, nnj=ny2,
236                                 fill_condition=initial,
237                                 label="block2",
238                                  cf_list=[RobertsClusterFunction(0, 1,
                                        1.08),
239                                         RobertsClusterFunction(1, 0,
                                            1.04),
240                                         RobertsClusterFunction(0, 1,
                                            1.08),
241                                         RobertsClusterFunction(1, 0,
                                            1.04)],
242                                 bc_list=[FixedTBC(308), SlipWallBC(),
243                                 FixedTBC(308), FixedTBC(308)])
244  # Mixing chamber1( Rectanguler Duct with hydrulic Diameter)
245
246 blk3 = Block2D(make_patch(north3, east3, south3, east1),
247                                nni=nx2, nnj=ny0,
248                                fill_condition=initial,
249                                label="block3",
250                                 cf_list=[RobertsClusterFunction(1, 0,
                                        1.08),
251                                         None,
252                                         RobertsClusterFunction(1, 0,
                                            1.08),
253                                         RobertsClusterFunction(0, 1,
                                            1.04)],
254                                bc_list=[SlipWallBC(), ExtrapolateOutBC
                                        (),
255                                 SlipWallBC(), SlipWallBC()])
```

```
256
257
258
259
260 blk4 = Block2D(make_patch(north4, east4, north3,west4),
261                            nni=nx2, nnj=ny1,
262                            fill_condition=initial,
263                           label="block4",
264                             cf_list=[RobertsClusterFunction(1, 0,
                                    1.08),
265                                      None,
266                                      RobertsClusterFunction(1, 0,
                                          1.08),
267                                      RobertsClusterFunction(1, 1,
                                          1.04)],
268                            bc_list=[SlipWallBC(), ExtrapolateOutBC
                                ()],
269                             SlipWallBC(), FixedTBC(308)])
270
271
272 blk5 = Block2D(make_patch(north5, east5, north4, east2),
273                            nni=nx2, nnj=ny2,
274                            fill_condition=initial,
275                            label="block5",
276                             cf_list=[RobertsClusterFunction(1, 0,
                                    1.08),
277                                      None,
278                                      RobertsClusterFunction(1, 0,
                                          1.08),
279                                      RobertsClusterFunction(1, 0,
                                          1.04)],
280                             bc_list=[FixedTBC(308),
                                    ExtrapolateOutBC(),
281                             SlipWallBC(), SlipWallBC()])
282
283
284 # Group bell mouth inlet
285
286 blk6 = Block2D(make_patch(north6, east6,south6,east3),
287                            nni=nx6, nnj=ny0,
288                            fill_condition=initial,
289                           label="block6",
290                            cf_list=[None,
291                                      None,
292                                      None,
```

```
                                        None],
                                bc_list=[SlipWallBC(), ExtrapolateOutBC(),
                                  SlipWallBC(), SlipWallBC()])


blk7 = Block2D(make_patch(north7, east7, north6, east4),
                               nni=nx6, nnj=ny1,
                               fill_condition=initial,
                               label="block7",
                                  cf_list=[None,
                                           None,
                                           None,
                                           None],
                               bc_list=[SlipWallBC(), ExtrapolateOutBC(),
                               SlipWallBC(), SlipWallBC()])


blk8 =Block2D(make_patch(north8, east8, north7, east5),
                               nni=nx6, nnj=ny2,
                               fill_condition=initial,
                               cf_list=[None,None,None,None],
                               bc_list=[FixedTBC(308),
                                   ExtrapolateOutBC(),
                                SlipWallBC(), SlipWallBC()],
                                label="block8")




identify_block_connections()


# Do a little more setting of global data.
#gdata.dimensions=2
gdata.axisymmetric_flag = 1
gdata.viscous_flag = 1
gdata.turbulence_flag = 1
gdata.turbulence_model = "k_omega"
gdata.flux_calc = ADAPTIVE
gdata.max_time = 30.0e-3  # seconds
gdata.max_step = 7000000
#gdata.cfl= 0.4
```

```
335 #gdata.cfl_count=3
336 gdata.dt = 1.0e-12
337 gdata.dt_plot = 0.1e-3
338 gdata.dt_history = 1.0e-4
339
340 sketch.xaxis(-0.01, 0.4, 0.1, -0.005)
341 sketch.yaxis( 0.0, 0.05, 0.005, -0.015)
342 sketch.window(-0.011, -0.006, 0.4, 0.2, 0.02, 0.01, 0.22, 0.65)
```

Appendix B

B.1 cone20.py

```
## \file cone20.py
## \brief Simple job-specification file for e3prep.py
## \author PJ, 08-Feb-2005
##
## 15-Sep-2008 -- simplified version for Eilmer3
##
## We have set this file up very much like the cone20.sit
## file so that users may more-easily see the correspondence
## between the Tcl and Python elements.

job_title = "Mach 1.5 flow over a 20 degree cone."
print job_title

# We can set individual attributes of the global data object.
gdata.dimensions = 2
gdata.title = job_title
gdata.axisymmetric_flag = 1
gdata.stringent_cfl = 1  # to match the old mb_cns behaviour

# Accept defaults for air giving R=287.1, gamma=1.4
select_gas_model(model='ideal gas', species=['air'])

# Define flow conditions
initial = FlowCondition(p=5955.0,  u=0.0,    v=0.0, T=304.0)
inflow  = FlowCondition(p=95.84e3, u=1000.0, v=0.0, T=1103.0)

# Set up two quadrilaterals in the (x,y)-plane be first defining
# the corner nodes, then the lines between those corners and
# then the boundary elements for the blocks.
```

```
30 # The labelling is not significant; it is just to make the SVG
31 # picture look the same as that produced by the Tcl scriptit
       program.
32 a = Node(0.0, 0.0, label="A")
33 b = Node(0.2, 0.0, label="B")
34 c = Node(1.0, 0.29118, label="C")
35 d = Node(1.0, 1.0, label="D")
36 e = Node(0.2, 1.0, label="E")
37 f = Node(0.0, 1.0, label="F")
38
39 # lower boundary including cone surface
40 ab = Line(a, b); bc = Line(b, c)
41 # upper boundary
42 fe = Line(f, e); ed = Line(e, d)
43 # vertical lines
44 af = Line(a, f); be = Line(b, e); cd = Line(c, d)
45
46 # Define the blocks, boundary conditions and set the
       discretisation.
47 nx0 = 10; nx1 = 30; ny = 40
48 blk_0 = Block2D(make_patch(fe, be, ab, af), nni=nx0, nnj=ny,
49                 fill_condition=initial, label="BLOCK-0")
50 blk_1 = Block2D(make_patch(ed, cd, bc, be,"AO"), nni=nx1,
51                 nnj=ny, fill_condition=initial, label="BLOCK-1",
52                 hcell_list=[(9,0)], xforce_list=[0,0,1,0])
53 identify_block_connections()
54 blk_0.bc_list[WEST] = SupInBC(inflow, label="inflow-boundary")
55 # one way to set a BC
56 blk_1.set_BC(EAST, EXTRAPOLATE_OUT, label="outflow-boundary")
57 # another way
58
59 # Do a little more setting of global data.
60 gdata.viscous_flag = 1
61 gdata.flux_calc = ADAPTIVE
62 gdata.compression_tolerance = -0.05 # the old default value
63 gdata.max_time = 5.0e-3  # seconds
64 gdata.max_step = 3000
65 gdata.dt = 1.0e-6
66 gdata.dt_plot = 1.5e-3
67 gdata.dt_history = 10.0e-5
68
69 sketch.xaxis(0.0, 1.0, 0.2, -0.05)
70 sketch.yaxis(0.0, 1.0, 0.2, -0.04)
71 sketch.window(0.0, 0.0, 1.0, 1.0, 0.05, 0.05, 0.17, 0.17)
```

B.2 cone20_run.sh

```
#! /bin/sh
# cone20_run.sh

# Prepare the simulation input files (parameter, grid and
    initial flow data).
# The SVG file provides us with a graphical check on the
    geometry
e3prep.py --job=cone20 --do-svg
if [ "$?" -ne "0" ] ; then
    echo "e3prep.py ended abnormally."
    exit
fi

# Integrate the solution in time,
# recording the axial force on the cone surface.
time e3shared.exe -f cone20 --run --verbose
if [ "$?" -ne "0" ] ; then
    echo "e3shared.exe ended abnormally."
    exit
fi

# Extract the solution data and reformat.
# If no time is specified, the final solution found is output.
e3post.py --job=cone20 --vtk-xml

echo "At this point, we should have a solution that can be
    viewed with Paraview."
```

Appendix C

C.1 Velocity profile.sh

```
#! /bin/sh
# Extract the solution data and reformat.
# If no time is specified, the final solution found is output.

## The different data files are obtained from the solution
    provided by Eilmer3.

e3post.py --job=laminar --tindx=9999 --add-pitot-p --add-mach --
    output-file=lam_20mm.data \
      --slice-list="0,18,:,0"
e3post.py --job=laminar --tindx=9999 --add-pitot-p --add-mach --
    output-file=lam_75mm.data \
      --slice-list="1,30,:,0"
e3post.py --job=turbulent --tindx=9999 --add-pitot-p --add-mach
    --output-file=turb_20mm.data \
      --slice-list="0,18,:,0"
e3post.py --job=turbulent --tindx=9999 --add-pitot-p --add-mach
    --output-file=turb_75mm.data \
      --slice-list="1,30,:,0"
e3post.py --job=turbulent_low_intensity --tindx=9999 --add-pitot
    -p --add-mach --output-file=turb_low_20mm.data \
      --slice-list="0,18,:,0"
e3post.py --job=turbulent_low_intensity --tindx=9999 --add-pitot
    -p --add-mach --output-file=turb_low_75mm.data \
      --slice-list="1,30,:,0"

## Plot the velocity profiles 20mm after inflow with GNU plot.

gnuplot <<EOF
```

```
23 set term postscript eps enhanced 20
24 set output "pitot_plots/Velocity_profile_20mm_turb.eps"
25 set style line 1 linetype 1 linewidth 3.0
26 set style line 2 linetype 2 linewidth 5.0
27 set style line 3 linetype 3 linewidth 4.0
28 set title "Velocity profile of the turbulent flow at 20 mm from
       inflow"
29 set xlabel "Velocity(m/s) in x-direction"
30 set ylabel "Height(mm)"
31 set key bottom left
32 plot "turb_20mm.data" using (\$6):(\$2*1000) title "turbulent
       velocity profile" with line, "lam_20mm.data" using (\$6):(\$2
       *1000) title "laminar flow" with line 2, "turb_low_20mm.data"
        using (\$6):(\$2*1000) title "turbulent low intensity" with
       line 3
33 EOF
34
35
36 ## Plot the velocity profiles 75mm after inflow with GNU plot.
37
38 gnuplot <<EOF
39 set term postscript eps enhanced 20
40 set output "pitot_plots/Velocity_profile_75mm_turb.eps"
41 set style line 1 linetype 1 linewidth 3.0
42 set style line 2 linetype 2 linewidth 5.0
43 set style line 3 linetype 3 linewidth 4.0
44 set title "Velocity profile of the turbulent flow at 75 mm from
       inflow"
45 set xlabel "Velocity(m/s) in x-direction"
46 set ylabel "Height(mm)"
47 set key bottom left
48 plot "turb_75mm.data" using (\$6):(\$2*1000) title "turbulent
       velocity profile" with line, "lam_75mm.data" using (\$6):(\$2
       *1000) title "laminar flow" with line 2, "turb_low_75mm.data"
        using (\$6):(\$2*1000) title "turbulent low intensity" with
       line 3
49 EOF
```