University of Stavanger

Faculty of Science and Technology

MASTER'S THESIS

Study program/ Specialization: Offshore Technology/ Risk Management	Spring semester, 2015 Open
Writer: Mohammad Ikram Hossain Talukdar	…… (Writer's signature)
Faculty supervisor: Bjørn H. Hjertager External supervisor: Knut Erik Giljarhus	
Title of thesis: Numerical simulation of underexpanded air jet using OpenFOAM	
Credits (ECTS): 30	
Key words: • rhoCentralFoam • FVM • underexpanded • CFD • OpenFOAM • Mach disk	Pages: 53 +enclosure: 38 + CD Stavanger, 15/06/2015

Acknowledgement

This thesis has been conducted in alliance with Llyod's Register Consulting whom I would like to appreciate for granting me the opportunity to accomplish their proposal under risk management program. I would like to express my profound appreciation to my internal supervisor Bjørn Helge Hjertager of University of Stavanger contemplating granting me for this position and according worthwhile suggestions. Also, I would like to express my gratitude to my external supervisor Knut Erik Giljarhus of Llyod's Register Consulting for being a mentor all over the thesis execution. His positive attitude of conveying team spirit has constituted this thesis more interesting and compelling to me. Special thanks to my family and friends for their love and support.

Abstract

It is of utmost importance for the awareness of safety issues involved in high pressure gas storage to perceive the adjacent field of high pressure gas jet release for the establishment of the decomposition laws in the far field. The numerical simulations of the first cell of an underexpanded gas jet can be performed executing finite volume solver which can be validated later by means of available literature source. The prominence of OpenFoam is irrefutable fact especially in the research field of fluid dynamics. At the same time it is indisputable that the lack of proper documentation about the authentication of OpenFoam solver may yield ambiguity about the simulation result. Therefore it requires experimental validations and resemblance with the results of other available commercial CFD software packages where the analogy can be drawn with the OpenFoam simulation results for credibility. This report has pursued the assessment of an executable OpenFoam solver known as rhoCentralFoam in terms of its competence of dealing with supersonic air flow due to leakage from high pressure reservoir and compared it to the outcome of practical experiment and other commercial CFD software like KFX and FLACS that are programmed on the basis of some simplified methods.

Table of Contents

1. Introduction

The situation of supersonic jet are often encountered while dealing with risk assessment due to leakage in the offshore and process industry. Any kind of accidental release of supersonic jet through an orifice of the high pressure reservoir requires the insight of flow properties for the establishment of appropriate safety standards. Due to the lack of proper knowledge and quantitative information of the near field of the under expanded supersonic jet it is challenging to measure and simulate supersonic flow in the shock structured region which requires simplified model of the supersonic flow. Such concept of simplified methods are proposed by Birch et al. (1984) and Birch et al. (1987) which are developed base on laws of conservation of mass, momentum, energy and ideal gas law.

It is crucial to study the resemblance between the numerical and experimental aspect of any prototype hazardous situation in offshore industry such as fire and ignition consequence due to leakage and dispersion especially in offshore modules in order to minimize the labor. There are many commercial CFD software like FLACS and KFX that are available to predict the fluid flow due to leakage which are very expensive and requires overcoming hurdle such as expensive hardware and commercial license to model the leakage dispersion and fire explosions for the enhancement of the risk quantification process. But it is always challenging to perform the similar task in open source CFD software like OpenFOAM which is becoming more popular day by day for its efficiency in handling aerodynamic investigations. It is consist of numerous solvers and utilities capable of handling wide range of problems. Unfortunately there are not enough evidence that are documented consisting the verification of numerical data from OpenFOAM with the experimental data because open source software requires more knowledge and proficiency of the user as compare to other commercial CFD software. This report attempts to find the comparison between the CFD data obtained from OpenFAOM with some experimental results. The underexpanded supersonic flow are of great interest in this report the axisymmetric CFD model of which will be simulated and investigated through the measurement of flow properties and some shock structure parameters which will be compared with the experimental results as well as some simplified models employed by the commercial CFD software like KFX and FLACS. For this the one dimensional shock tube case simulation is considered as an initiation to appraise the capability of the OpenFOAM solver.

1.1 Previous Work

This section is inspired and retrieved partly from Vembe et al. (2001), Menon (2009), and Gribben (1999). Jet flow on the sonic and supersonic nozzles were a great subject of interest in the fifties and sixties decade due to the use of jet, rocket engines and recoilless rifles. Also there was a great investigation on both numerical and experimental aspects of underexpanded jets in the last decade especially on the oil and gas industries. The commencement of investigation over the underexpanded jet flow through axisymmetric nozzle was performed over the past half century by Love at al. (1959) and the characteristics method developed to determine the near field characteristics of the under expanded jet by Pack (1950). The initial inclination of the underexpanded jet also had been studied in the past by Love (1956) and Love (1958) for different pressure ratios. The solution approach of Euler equation applied by Katanoda et al. (2000) and Navier-Stokes equation by Dash et al. (1985), McDaniel et al. (2002) and Gribben et al. (2000) in the cylindrical coordinates are also applied to determine the flow properties in the axisymmetric jet. The formation of under expanded jet is due to the conically tapered shape of the nozzle. The position of Mach disk in a shock structure formed by the jet from a sonic conically convergent nozzle was predicted by an empirical formula provided by Addy (1981). Later Birch et al. (1984) and Birch et al. (1987) establish the method to determine the position and diameter of the Mach disk.

Numerous attempt had been devised to calculate numerically the flow field of the shock structure of underexpanded jet. Specially the use of Euler and Navier-Stokes solver to obtain numerical solution for the under expanded jet plume had been employed with impressive outcome by Prudhomme (1994), Cumber et al. (1995), Hsu (1991) and Birkby et al. (1996) which demonstrates a better agreement with the experimental results up to an wide range of conditions considering several parameter such as the location and diameter of the Mach disk and centerline velocity of a complex shock structure. Now a days, many commercial and open source software are available that employs these solvers that comprises of finite volume discretization scheme to predict the high flow characteristics.

1.2 Thesis Objective

The objective of this thesis is the OpenFOAM simulation of supersonic air flow due to leakage from a high pressure reservoir numerically and investigate the flow properties downstream of the first cell of the shock structure produce by leakage adopting the density based OpenFOAM solver called rhoCentralFoam. The OpenFOAM simulation will be conducted in three phases. In the first phase a one dimensional shock tube case is simulated and its results are demonstrated as a general assessment of the accuracy of the solver and its discretization scheme. The second phase consist of axisymmetric simulation of a shock structure and it's comparison with the experimental results known as Ladenburg experiment (Greenshields et al. 2009) through some benchmark such as the location and diameter of Mach disk of the shock structure. The third phase consist of simulation of air jet shock structure for a different reservoir pressure and the OpenFOAM simulation properties of this case are compared with the results of commercial software used for the similar purpose such as KFX and FLACS that adopt the simplified method proposed by Birch et al. (1984) and Birch et al. (1987).

2. Theoretical background

The results of hydrocarbon leakage from the high pressure gas system is more interesting when investigating the consequences rather than the dispersion detail of the shock structure formed by the jet. But a detail study of the shock structure is necessary when scrutinizing the near-filed effect close to the outlet. This chapter will argue on the theoretical feature of the shockwave formation, its numerical simulation methodology and experimental validation followed by some simplified mathematical jet model which forms the basis for some commercial gas jet modeling software.

2.1 Overview of a shock wave and Mach disks formation

Theoretical part of this section is inspired from Bayeh (2009) and Vembe et al. (2000). When air leaked from a high pressure reservoir to the atmosphere through the throat of a supersonic nozzle, the pressure along the nozzle declines due to its expansion and acceleration. The reservoir pressure is greater than the ambient pressure thus defines the nozzle as underexpanded. The leaked underexpanded gas jet expands to the atmospheric pressure by means of an expansion fan. The free air jet boundary then reflects the expansion waves off as compression waves.

The accumulated compression waves at a point then form an oblique shock wave. Due to the assumption of axisymmetric flow for the nozzle flow, the formed oblique shock wave is again reflected off the center line of the flow. This actually causes the formation of a Mach reflection prior to the intersection point with the centerline that produces a shock triple point as in the Figure 1 where the compression wave, oblique shock and centerline coincide. The incoming oblique shock wave is the first Mach reflection. The second shock is Mach reflection that is normal one to the centerline termed as a normal shock that forms the Mach disk for axisymmetric flow. The third Mach reflection is another oblique shock wave that is reflected back off the constant pressure free air jet boundary as an expansion fan.

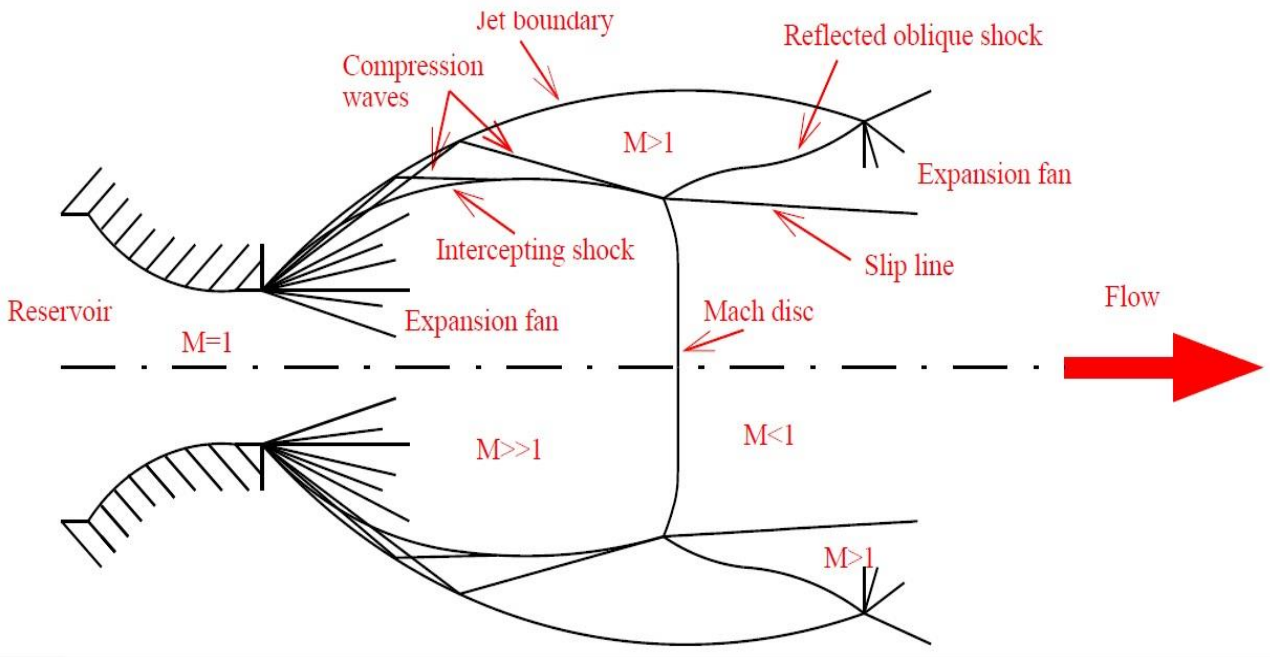


Figure 1. Schematics of the first cell of an underexpanded gas jet (Vembe et al., 2001)

The expansion wave that is formed by the reflected off oblique shock from the constant pressure free air jet boundary would then continue repeating the process like the Fig. 1 but does not occur with an infinite number of cells because of the isentropic losses due to shocks and viscous effect of the fluid flow. The escalation of the pressure and temperature occur as the flow passes through the normal shock or Mach disk for axisymmetric flow. So the Mach disks are the high pressure region in an underexpanded air jet that are formed through a repeating and decaying series of shock waves and expansions waves formed by the extensive difference between the reservoir pressure and the ambient pressure. Figure 2 demonstrates a physical view of the Mach disk ignition formed by an aircraft jet engine.

Figure 2. A series of Mach disks in the exhaust plume of jet engine (Bayeh, 2009).

There are several parameters effecting the flow pattern of this type of jet flow discharged from a straight or converged nozzle are the pressure ratio, nozzle exit pressure and the ambient pressure. When the pressure ratio is more than two then the jet is termed as highly underexpanded jet (Vembe et al, 2001).

2.2 One dimensional shock process-the shock tube problem

The dissertation of this section is inspired and retrieved from Sod (1978), Ksibi (2008), Greenshields et al. (2009), Tsangaris (2000) and Zhang (2014). Experimentally, shock tubes are studied to observe the shock process, their interaction and effects. It is also analyzed as a simplified model of real life prototype for the research purpose where simply designed shock tube with few meters length are adopted with varying compressed air pressures.

As introduced by Ksibi (2008), conventionally, a shock tube is a strong smooth wall steel pipe with a circular or rectangular cross section divided into two different pressure compartments initially at different pressure value s and separated by a diaphragm. Assuming the viscous effect are negligible along the tube and the tube length is large enough to avoid reflections at it ends, Euler equations can be solved to achieve the exact solution considering simple wave analysis as a basis. When the diaphragm bursts due to the pressure difference, leftward and rightward moving waves are created due to the breakage of the two initial stages. As mentioned by Sod (1978), the shock wave moves to the low pressure region and refraction wave moves to the high pressure region. Figure 3 represents an analytical overview of shock tube with a burster diaphragm.

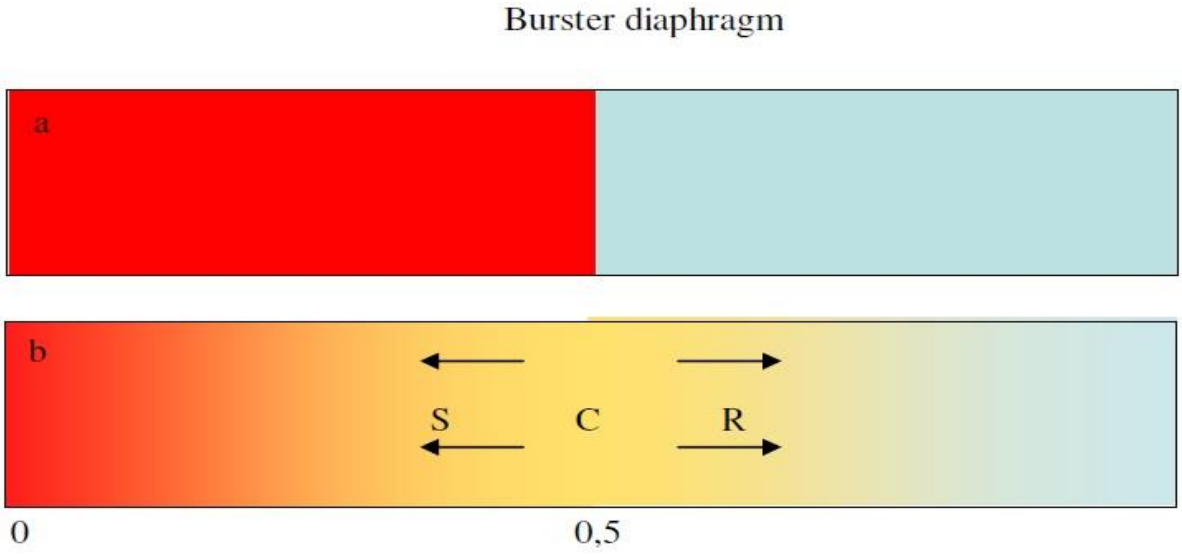


Figure 3. Primary condition of shock tube with discontinuity at mid; a: t = 0.s; b = t>0 s; S = Shock wave, R = Refraction wave, C = Contact discontinuity (Ksibi, 2008).

The shock tube problem is considered as an attractive test case from the numerical perspective because of the knowledge of exact time dependent analytical solution that can be contrasted with the computed solution achieved from the application of numerical approximations. Initial configuration of this test case consist of two uniform states of different pressure and temperature separated by a discontinuity situated at the mid center of the tube. Assuming the tube to be closed at both ends, both the left and right side boundaries of the computational domain are considered ideal solid walls. Both the shock and refraction wave never reach to the end walls due to the shortage of the integration time.

Followed by Greenshields et al. (2009) that the numerical solution domain is one dimensional in the range -5m < x < 5m considering the location of the diaphragm is at x = 0. The one dimensional shock tube case is run with a fixed time step corresponding to a Courant-Friedrichs-Lewy (CFL) number 0.2 below the stability limit of 0.5 of the central schemes. CFL number is a numerical constraint that determine the information propagation through one or several cells. Table 1 shows the initial case set up for the shock tube problem and Figure 4 demonstrates the exact analytical and numerical results at t = 7ms. The hypothetical diaphragm that separated the two chamber has been removed initially at time t = 0 and the inviscid equations are time integrated to 0.007 s with CFL 0.2. The edge of the experimental result shows some smearing and that is the only difference from the analytical result. The smooth edge can be sharpened by introducing finer initial numerical discretization scheme.

Table 1. Initial set up of the shock tube case (Greenshields et al. 2009)

Region (-5<x=0<5)	Pressure (Pa)	Temperature (K)	Density ($kg.m^{-3}$)
Left (-5<x=0)	10^5	348.4	1.0
Right (x=0<5)	10^4	278.7	0.125

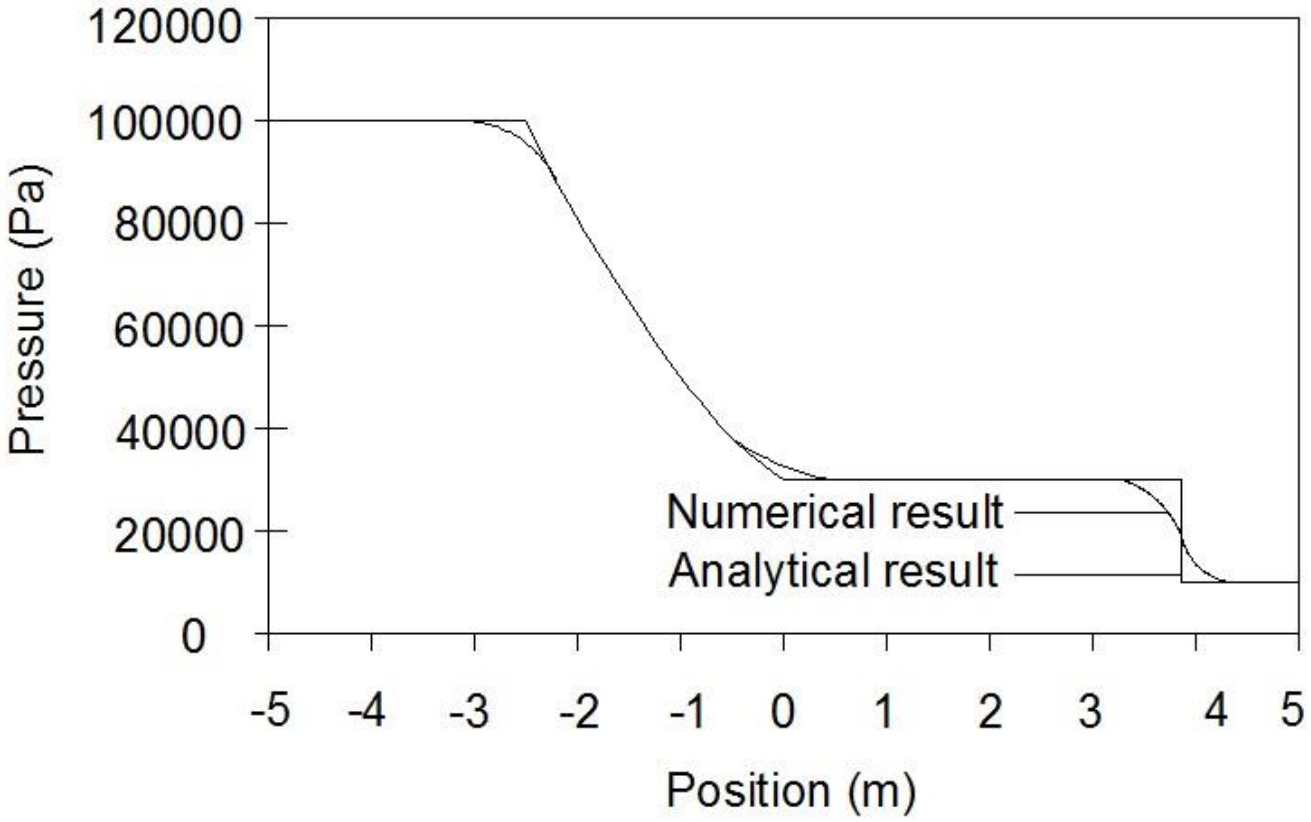


Figure 4. Shock tube analytical and numerical pressure profile at t = 7ms (Tsangaris, 2000)

Due to the high pressure difference of the compartments, an unsteady flow with the shock wave is formed due to the sudden diaphragm rupture. As reported by Zhang (2014), the growth of boundary layer attenuates the shock waves which is the major reason for the decaying of the shock wave propagation which is obvious from figure 4. Also viscous stresses attenuates both the shock wave velocity and propagation. Shock tube with smaller diameter leads the shock wave attenuation more than that of larger diameter because of the thicker boundary layer.

2.3 Supersonic air jet experiment (Ladenburg experiment)

The two dimensional supersonic air jet experiment performed by the use of Mach interferometer and the density distribution in the axially symmetric jet is achieved by Ladenburg et al. (1949). The supersonic air jet problem is well known as Ladenburg experiment. Greenshields (2009) demonstrates an overview of this experiment where dry air was discharged through a circular tapered nozzle shown in Figure 5 of 10mm orifice diameter from a high pressure reservoir tank. As the tapered circular hole of the nozzle is bored out of a cylindrical block, the exit orifice plane consist of flat solid wall. This experimental case will be simulated numerically using OpenFOAM application in the next chapter considering following data that are given below:

Insider pressure of the tank is 4.14 bar

Properties at the nozzle throat or inlet condition p = 2.72 bar, **u** = (315.6, 0, 0) and T = 247.1 K

Free stream condition or outlet condition p = 1.01 bar, **u** = (0, 0, 0) and T = 297 K

Ratio of specific heat $\gamma = 1.4$

Prandtl number $P_r = \mu c_p/k = 0.75$ (assumed)

For dry air, gas constant R = 287 J/kgK

From Sutherland's Law, viscosity $\mu = A_s [T^{1.5}/(T+T_s)]$

Where A_s = constant = $1.458* 10^{-6}$ Pa/$K^{0.5}$ and T_s = Sutherland's temperature = 110.4K

,

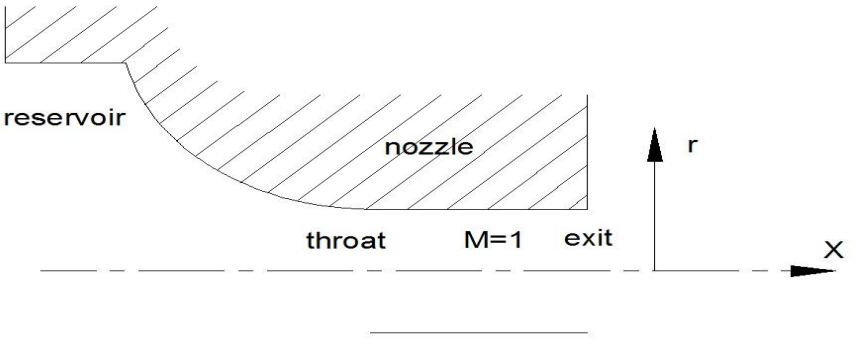


Figure 5. Layout of a circular tapered nozzle used in Ladenburg experiment (Greenshields, 2009)

The experimental result shown on the density Figure 6 where a weak shock is produced due to the expansion of air from the nozzle orifice from the density $\rho = 3.8$ kg/m^3 which extends towards the nozzle axis. Thus it creates a Mach disk feature and a triple point. Interferogram analysis yields the Ladenburg data shown on the density figure 6 that shows the position of triple point i.e. the location and diameter of Mach disk at (13.3 mm, 1.7 mm) approximately provided the jet radius are plotted against the distance from orifice.

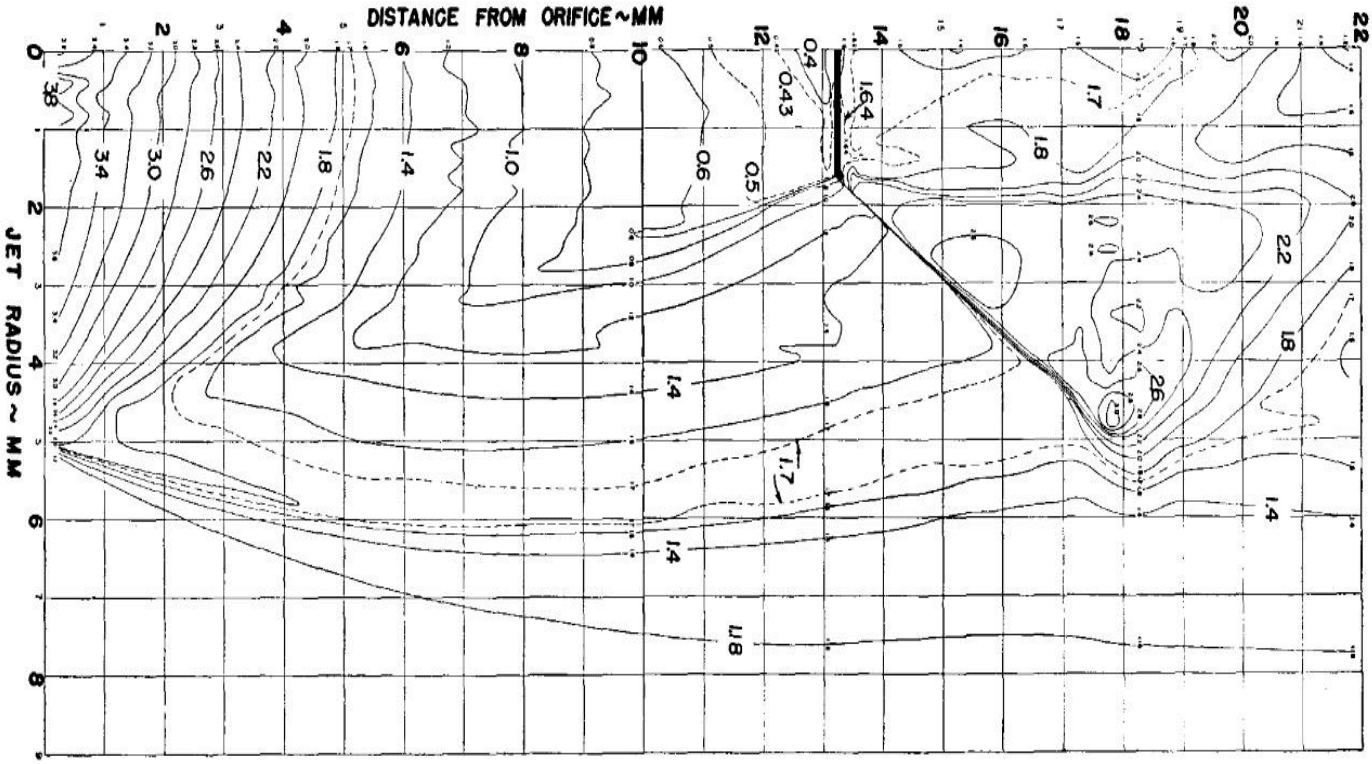


Figure 6. Density distribution and the location and diameter of Mach disk achieved from Ladenburg experiment (Ladenburg, 1949).

2.4 Numerical solution approach

The dissertation of this section is inspired from Greenshields et al. (2009), Marcantoni (2012), Kurganov (2000) and Kurganov et al. (2001) where discretization procedure of governing equations and computational method will be demonstrated.

2.4.1 Governing equations

Considering Eulerian frame of reference, the standard governing flow equations aspired to solve are:

- Mass conservation

$$\frac{\partial \rho}{\partial t} + \boldsymbol{\nabla} . \ [\mathbf{u}\rho] = 0 \qquad (1)$$

- Conservation of momentum assuming no body forces

$$\frac{\partial (\rho \boldsymbol{u})}{\partial t} + \boldsymbol{\nabla} . \ [\mathbf{u}(\rho \mathbf{u})] + \boldsymbol{\nabla} p + \boldsymbol{\nabla} . \mathbf{T} = 0 \qquad (2)$$

- Conservation energy total energy

$$\frac{\partial (\rho E)}{\partial t} + \boldsymbol{\nabla} . \ [\mathbf{u}(\rho \mathrm{E})] + \boldsymbol{\nabla} . [\mathbf{u} p] + \boldsymbol{\nabla} . (\mathbf{T} . \mathbf{u}) + \boldsymbol{\nabla} . \mathbf{j} = 0 \qquad (3)$$

Where

ρ is the mass density, **u** is the fluid velocity field, p is the pressure, E is the total energy density expressed by $E = e + |u|^2/2$ with e as the specific internal energy, **T** is the viscous stress tensor considered positive in compression and **j** is the diffusive heat flux.

In case of non-viscous or inviscid flow **T** and **j** can be set to zero which transforms the above equation into Euler's equations. Considering no bulk viscosity the stress tensor **T** can be represented by the following equation according to newton's law:

$$\boldsymbol{T} = -2\mu dev(\boldsymbol{D}) \qquad (4)$$

Where μ is the dynamic viscosity and **D** is the deformation gradient tensor expressed by the equation $\mathbf{D} = 0.5[\boldsymbol{\nabla}\mathbf{u} + (\boldsymbol{\nabla}\mathbf{u})^{\mathbf{T}}]$ and the deviatoric component dev (**D**) can be expressed as

dev(**D**) = **D** – (1/3) tr(**D**)**I** where **I** = unit tensor

The diffusive heat flux **j** can be defined by the fourier's law as follows:

$$\boldsymbol{j} = -k\nabla T \qquad (5)$$

Where k is the heat conductivity and T is the temperature. Using the viscous and heat conduction equation the equations of Navier-Stokes are derived. Assuming the working gas (air) as a perfect caloric gas the relations are implemented:

$$p = \rho RT \qquad (6)$$

$$e = c_v \mathrm{T} = (\gamma - 1)\mathrm{RT} \qquad (7)$$

Where R is the gas constant, $\gamma = c_p/c_v$ is the specific heat ratio at constant pressure c_p and volume c_v respectively.

The value of temperature T is determined by the following equation:

$$\mathrm{T} = \frac{1}{c_v}\left[\frac{\rho E}{\rho} - \frac{|u^2|}{2}\right] \qquad (8)$$

2.4.2 Computational Methods

Finite Volume Method is used in the meshed of polyhedral cells with arbitrary number of faces each of which contains arbitrary number of vertices. The computational domain is therefore divided into several number of cell or control volumes. The number of cells can vary and there is no alignment among them generally. A cell is connected to another cell either by intersecting two cells only or the face is internal. The cells face can be treated as an external boundary. Figure 7 represent two cells or control volume (CV) of a computational domain. The first cell is connected to the second cell with a face f indicated by a surface area vector S_f. The cell which has this area vector is called owner cell and the sharing cell is called neighbor cell. Both of center of the cell is denoted by P and N respectively. Vector d represents the connecting line of the both of the center and vector d_{fN} connects the center of the faces. The partial differential equation (PDE) are integrated over these CV's and all the volume integrals are converted into surface integrals over face f applying Gauss's theorem.

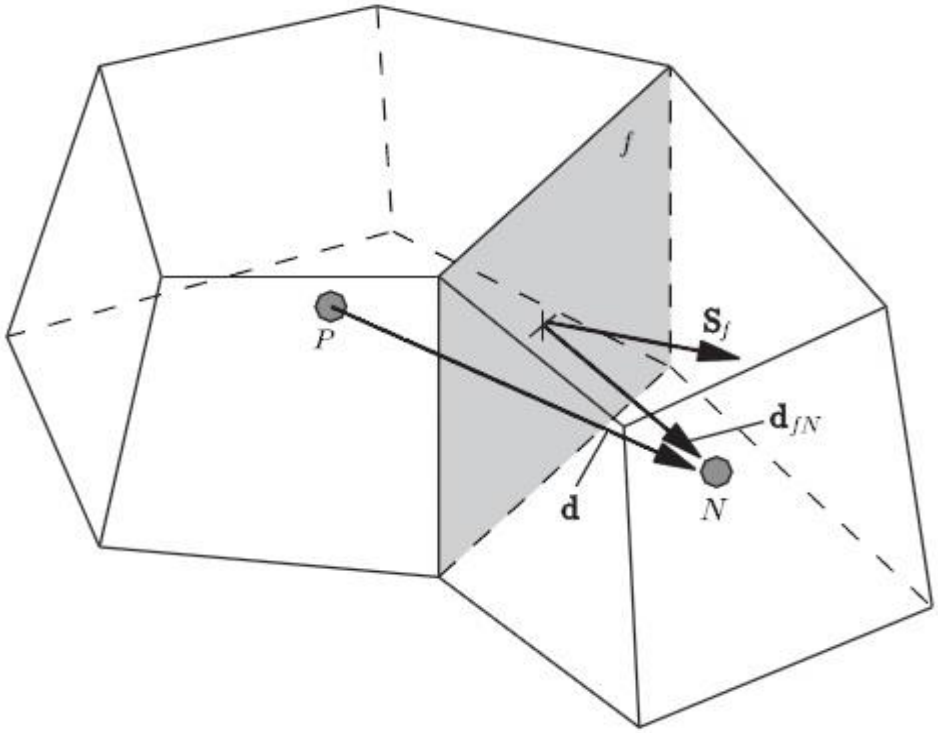


Figure 7. Finite volume discretization (Greenshields et al. 2009)

In the discretization process, all the surface integrals are transformed into a set of linear algebraic equations consist of flux value of the primary variable Ψ_f whereas the flux values on the other faces like are found through interpolation of the flux values at the centers Ψ_P and Ψ_N respectively.

The Kurganov (2000) and Kurganov et al. (2001) method which is the second-order semi-discrete, non-staggered schemes enables the formation of flux interpolations considering its transport in any direction since the properties in the compressible fluids transported by the flow as well as wave propagation.

2.4.3 Discretization of convective terms

The convective terms in the governing equations are $\nabla . [u\rho]$, $\nabla . [u(\rho u)]$, $\nabla[u(\rho E)]$ and $\nabla . [up]$ which are integrated over the CV and linearized as follows.

$$\int_V \nabla . [u\Psi] dV = \int_S dS . [u\Psi] \approx \sum_f S_f . u_f \Psi_f = \sum_f \phi_f \Psi_f \qquad (9)$$

Where $\sum_f$ indicates summation over all faces and $\phi_f = S_f . u_f$ is volume of fluid flow per second through the face known as the volumetric flux.

Now to obtain $\mathbf{u}_f$, central differencing method is used which is the linear interpolation of **u** with respect to the neighboring cells and Ψ_f is obtained through upwind differencing which is performed by splitting the flux in outgoing and incoming directions to the face of the owner cell (Marcantonia, 2012). The linear interpolation of Ψ is performed by the following equation.

$$\Psi_f = \omega_f \Psi_P + (1- \omega_f) \Psi_N \qquad (10)$$

Where ω_f = weighting coefficient = $| S_f . d_{fN} | / | S_f . d |$

Let the outward and inward of the face f and face area vector S_f are denoted by f+ ,+S_f and f- ,- S_f respectively. The discretization scheme used is as follow.

$$\sum_f \phi_f \Psi_f = \sum_f [\alpha \phi_{f+} \Psi_{f+} + (1 - \alpha)\phi_{f-} \Psi_{f-} + \omega_f (\Psi_{f-} - \Psi_{f+}) \qquad (11)$$

Where the first two terms on the right hand side of the equation represents the flux evaluation in the direction in the f+ and f- direction respectively. The third term which is an extra term necessary if only the convection term is part of a substantial or total derivative $\frac{\partial(\rho u)}{\partial t} + \nabla . [\mathbf{u}(\rho \mathbf{u})]$. It uses a volumetric flux ω_f based on the maximum speed of propagation of any discontinuity that may exist at a face between values interpolated in the f+ and f directions (Greenshields et al., 2009). Volumetric fluxes associated with the local speeds of propagation can be calculated as follows:

$$\Psi_{f+} = \max(c_f + |\boldsymbol{S}_f| + \phi_{f+},\ c_f - |\boldsymbol{S}_f| + \phi_{f-}, 0) \qquad (12)$$

$$\Psi_{f-} = max\ (c_f + |\boldsymbol{S}_f| - \phi_{f+},\ c_f - |\boldsymbol{S}_f| - \phi_{f-}, 0) \qquad (13)$$

Where $c_{f\pm}$ are the speeds sound of the gas at the face, outward and inward of the owner cell expressed as

$$c_{f\pm} = \sqrt{\gamma R\, T_{f\pm}} \qquad (14)$$

f+ and f- contributions are equally weighted and weighting coefficient $\boldsymbol{\alpha}$ = 0.5 in KT method. Therefore the scheme is a central scheme. On the other hand, KNP method use the scheme on which the weighting is biased in the upwind direction using the factor $\boldsymbol{\alpha}$ expressed as known as central upwind scheme.

$$\alpha = \frac{\Psi_{f+}}{\Psi_{f+} - \Psi_{f-}} \qquad (15)$$

(Greenshields et al., 2009) reported that the diffusive volumetric flux can be found as:

$$\omega_f = \begin{cases} \alpha \max(\Psi_{f+}, \Psi_{f-}) & \text{Kurganov and Tadmor Method} \\ \alpha\ (1-\alpha)(\Psi_{f+} + \Psi_{f-}) & \text{Kurganov, Noelle and Petrova Method} \end{cases} \qquad (16)$$

For switching between the low and high-order schemes, a flux limiter function β(r) in introduced in the interpolation procedure where r represents the ratio of successive gradients of the interpolated variable that is constrained to r ≥0. Greenshields et al. (2010) defines r on polyhedral mesh for the f+ direction is as follows :

$$r = 2\frac{\mathbf{d}\,.(\nabla \boldsymbol{\Psi})_P}{(\nabla_d \boldsymbol{\Psi})_f} - 1 \qquad (\boldsymbol{\Psi}\ is\ scalar) \qquad (17)$$

Where

$(\boldsymbol{\nabla\Psi})_P$ = Full gradient determined at the owner cell P by linear interpolation.

$(\boldsymbol{\nabla}_d\boldsymbol{\Psi})_f = \boldsymbol{\Psi}_N - \boldsymbol{\Psi}_P$ = Normal gradient component to the face scaled by $|\mathbf{d}|$.

Both Greenshields et al. (2009) and Marcantoni (2012) reported that the solver rhoCentralFoam uses the total variation diminishing (TVD) limiter which are symmetric as well for the interpolation of all flow properties such as pressure, temperature, velocity and density. Some of the limiter and limiter functions are given on the table 2.

Table 2. Limiter definition

Limiter	Limiter functions
Minmod	max (0, min (1, r))
Van Leer	$\frac{r + \lvert r \rvert}{1 + r}$
Van Albada	$\frac{r + \lvert r^2 \rvert}{1 + r^2}$

For example, the f+ interpolation of $\boldsymbol{\Psi}$ is simply determined by the simple equation below:

$$\boldsymbol{\Psi}_{f+} = (1 - g_{f+})\boldsymbol{\Psi}_P + g_{f+}\boldsymbol{\Psi}_N) \qquad (18)$$

Where

P and N are the indicator subscript of the owner and the neighbor cell.

$g_{f+} = \beta(1 - \boldsymbol{\omega}_f)$; $\boldsymbol{\omega}_f$ = the weighting factor = $| S_f \,.d_{fN} |/| S_f \,.d_{PN} |$ indicates that the vector d is a connector of the owner cell center P to the center of neighbor cell N and vector d_{fN} is a connector of the face center and the center of the neighbor cell N.

2.4.4 Discretization of gradient terms

All the gradient terms of the governing equation which also including ∇p are integrated over the CV and discretized as follows:

$$\int_V \nabla \Psi dV = \int_S dS\, \Psi \approx \sum_f S_f \Psi_f \qquad (19)$$

The Kurganov (2000) and Kurganov et al. (2001) schemes separate the interpolation procedure of Ψ_f toward the f+ and f- directions as per following equation:

$$\sum_f S_f \Psi_f = \sum_f [\alpha S_f \Psi_{f+} + (1-\alpha) S_f \Psi_{f-}] \qquad (20)$$

There are also usage of same limiter as mentioned earlier in table x for the interpolation of f+ and f-.

2.4.5 Discretization of Laplacian terms

Laplacian terms are discretized initially with the diffusion coefficient Γ for the polyhedral meshes as follows:

$$\int_V \nabla . (\Gamma \nabla \Psi) dV = \int_S dS\, (\Gamma \nabla \Psi) \approx \sum_f \Gamma_f S_f . (\nabla \Psi)_f \qquad (21)$$

Diffusion coefficient Γ_f in the face f is determined by linear interpolation from the value of the cell centroid. When vector $\mathbf{S}_f$ and vector $\mathbf{d}$ are not parallel to each other the face f is called an orthogonal face and in case of a non-orthogonal face the term $\mathbf{S}_f . (\nabla \Psi)_f$ is resolved into an orthogonal component represented in terms of Ψ at cell centroids (N and P) and a non-orthogonal component represented by a full gradient of Ψ at the face. The equation is as follows:

$$\mathbf{S}_f . (\nabla \Psi)_f = A\, (\Psi_N - \Psi_P) + \mathbf{a} . (\nabla \Psi)_f \qquad (22)$$

Where $A = |\mathbf{S}_f|^2 / (\mathbf{S}_f . \mathbf{d})$ and $\mathbf{a} = \mathbf{S}_f - A\mathbf{d}$

2.4.6 Discretization of temporal term

Temporal terms can be discretized by Euler explicit scheme as following:

$$\int_V \frac{\partial \Psi}{\partial t} = \frac{(\Psi^{n+1} - \Psi^n)\, dV}{\delta t} \qquad (23)$$

Where δt represents the time step.

2.4.7 Boundary conditions

A constant value $\boldsymbol{\Psi}_b$ is described at the boundary for a Dirichlet condition (fixed value on the boundary). The convective terms discretization requires the value of $\boldsymbol{\Psi}$ at all faces where $\boldsymbol{\Psi}_b$ can be replaced at boundary face. Laplacian terms discretization requires the normal gradient of $\boldsymbol{\Psi}$ at all faces determine by differencing $\boldsymbol{\Psi}_b$ and $\boldsymbol{\Psi}_i$ at the boundary face where i represents the cell next to boundary face.

A constant normal gradient $(n \,.\boldsymbol{\nabla}\Psi)_b$ is described at the boundary for a Neumann condition (a flux on the boundary) that can be replaced at boundary face for the Laplacian term discretization. The boundary face value for the convection term is determined by the extrapolation from $\boldsymbol{\Psi}_i$ and the normal gradient.

2.5 Simplified jet model

Any type of analytical model does not exist or available that fully envisage the shock structure. Therefore, simplifications are performed to the present analytical calculation method and simple mathematical model is formed which yields approximately accurate results as compare to numerical methods. The authentication of the simplified jet model are performed through experiments. The simplified model adopted by FLACS (gexcon.no, 2015) and KFX (computit.no, 2015) are two such embedded jet utility program used to determine the leakage parameter in case of under expanded jet. Both the program adopt the isentropic conditions and the procedure are based on simplified model proposed by Birch et al. (1984) and Birch et al. (1987) where the equation of mass, momentum and energy are employed. The dissertation of this section is inspired from Birch et al. (1984), Birch et al. (1987).

Gas leakage in the form of underexpanded jet rapidly achieve self-similarity i.e. the shapes of the normalized transverse profiles of mean quantities are preserved with distance downstream. The flow demonstrates a little memory of its initial configuration in this similar region. Therefore, we can expect that a supercritical jet (fluid at a pressure and temperature above critical point) would behave in a way that has resemblance to a classical free jet although with modified velocity and length scales. The simplified method defines a "pseudo-diameter" that replicates the concentration field of the fluid in the self-preserving region of a supercritical gas jet when supplanted in the equations that defines a subsonic round free jet. The "pseudo-diameter" consideration involves taking account of the area occupied by the same mass flow rate at ambient pressure and temperature with a uniform sonic velocity. Also conservation of mass and momentum are employed in the expansion region assuming the pressure is reduced to the ambient level. Figure 8 demonstrates the underexpanded release of a gas through an orifice diameter d. There are three level of conditions in terms of pressure, temperature and density existed here which are provided below:

Level 1- Reservoir conditions (p_1, T_1, ρ_1)

Level 2- The Orifice conditions (p_2, T_2, ρ_2)

Level 3- After expansion conditions (p_3, T_3, ρ_3)

There is no physical existence of level 3 that represent the "pseudo-diameter". Assuming no ambient fluid entrainment and negligible viscous force over the expansion surface, the equation for the conservation of mass can be written as:

$$\rho_2 u_2 A_2 C_D = \rho_3 u_3 A_3 \qquad (24)$$

Where u_2 and u_3 are the flow local sonic velocity and ambient sonic velocity respectively at level 2 and level 3 and C_d is the discharge coefficient considering the velocity profile uniformity. It is uniform i.e. equals to one in the "pseudo-diameter". The orifice diameter d for the cross sectional area A_2 and pseudo-diameter for the cross sectional area A_3 which converts equation (24) as:

$$\left(\frac{d_e}{\mathrm{d}}\right)^2 = C_D * \frac{u_2}{u_3} * \frac{\rho_2}{\rho_3} \qquad (25)$$

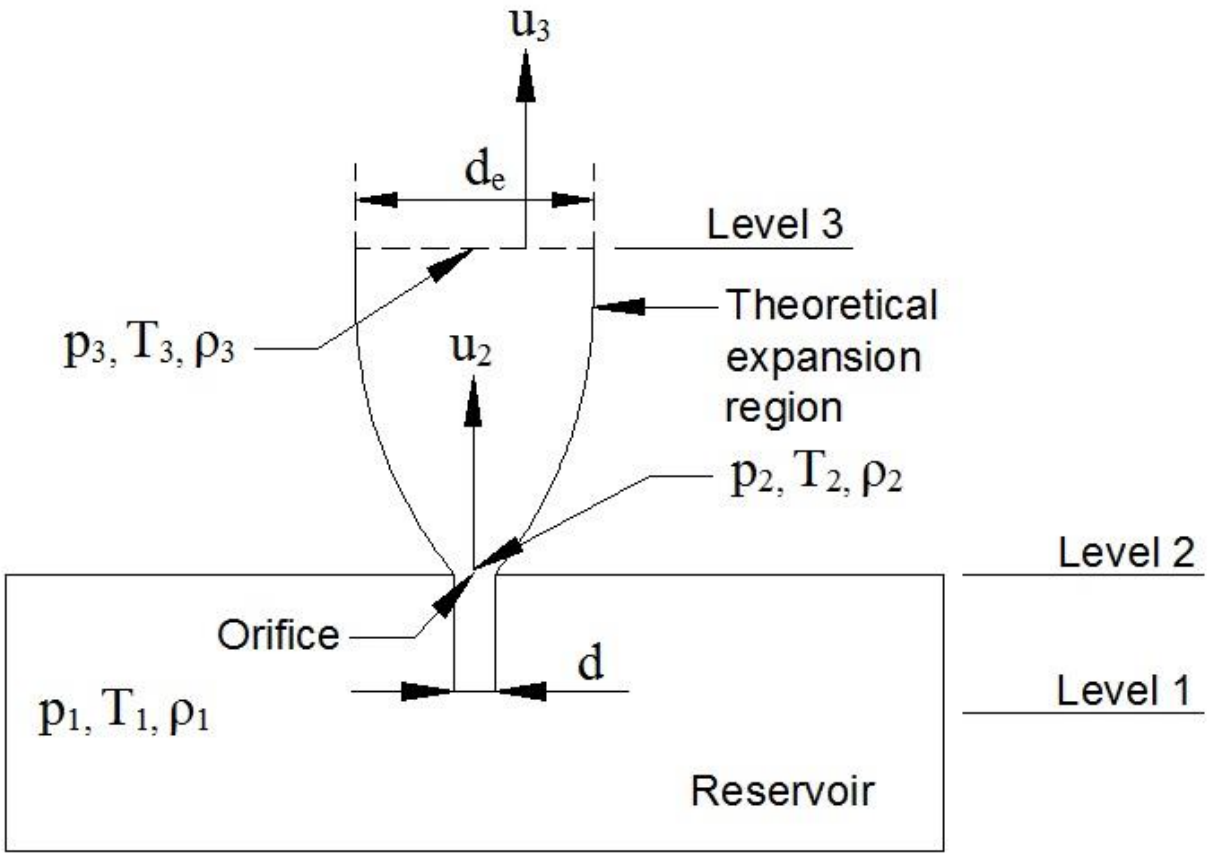


Figure 8. Simplified diagram for an underexpanded gas jet release (Birch et al., 1987)

The equation for the conservation of momentum can be defined as:

$$\rho_3 {u_3}^2 A_3 = \rho_2 {u_2}^2 A_2 {C_D}^2 + A_2(p_2 - p_3) \qquad (26)$$

Where the approximated momentum discharge coefficient is C_D^2 since there is no discharge coefficient due the present of a normal shock at level 3. Assuming p_3 as ambient pressure p_a, solving equation (24) and (26) for u_3 and A_3 yields:

$$u_3 = u_2 C_D + \frac{(p_2 - p_a)}{\rho_2 u_2 C_D} \tag{27}$$

And

$$\frac{A_3}{A_2} = \frac{\rho_2{}^2 u_2{}^2 C_D{}^2}{\rho_3(\rho_2{}^2 u_2{}^2 C_D{}^2 + p_2 - p_a)} \tag{28}$$

Considering the isentropic expansion and choked flow conditions ($p_1 \geq 2p_2$) at level 2, the reservoir parameters can be related to the conditions at level 2 by the following equations

$$p_2 = p_1 \left(\frac{2}{\gamma + 1}\right)^{\frac{\gamma}{\gamma - 1}} \tag{29}$$

$$T_2 = T_1 \left(\frac{2}{\gamma + 1}\right) \tag{30}$$

Where γ is the specific heat ratio. Now using the gas law ρ_2 can be defined as:

$$\rho_2 = p_1 \left(\frac{2}{\gamma + 1}\right)^{\frac{\gamma}{\gamma - 1}} * \frac{M}{R_{gas} T_1} \tag{31}$$

Where M is the molecular weight and R_{gas} is the universal gas constant. It is shown from the measurement that the temperature elevated just after the post shock region such that the original stagnation temperature of the fluid is restored rapidly i.e. $T_3 \approx T_1$. Thus we get the expression for ρ_3 as:

$$\rho_3 = \frac{p_3 M}{R_{gas} T_3} = \frac{p_a M}{R_{gas} T_1} \tag{32}$$

From equation (31) and (32) we get

$$\frac{\rho_2}{\rho_3} = \frac{p_1}{p_a} * \left(\frac{2}{\gamma + 1}\right)^{\frac{1}{\gamma - 1}} \tag{33}$$

Also the sonic velocity is proportional to the square root of the local temperature such that

$$u_2 = \sqrt{\frac{2\gamma}{\gamma+1} * \frac{R_{gas}T_1}{M}} \quad (34)$$

Using equation (29) and (31) the equation (27) can be derived as:

$$u_3 = u_2 C_D + \frac{(p_2 - p_a)}{\rho_2 u_2 C_D} = u_2[C_D + \frac{1 - \frac{p_a(\frac{2}{\gamma+1})^{-\frac{\gamma}{\gamma-1}}}{p_1}}{\gamma C_D}] \quad (35)$$

Thus equation (25) can be written as:

$$\frac{d_e}{d} = \sqrt{[C_D \frac{p_1}{p_a} * (\frac{2}{\gamma+1})^{\frac{1}{\gamma-1}} * \frac{u_2}{u_3}]} \quad (36)$$

Where d_e is the effective or pseudo-diameter. At relatively high pressure i.e. when

$$\frac{p_1}{p_a} \gg (\frac{2}{\gamma+1})^{-\frac{\gamma}{\gamma-1}} \quad (37)$$

Then equation (36) simplifies to

$$\frac{d_e}{d} = C_D \sqrt{[\frac{p_1}{p_a} * (\frac{2}{\gamma+1})^{\frac{1}{\gamma-1}} * \frac{1}{(\gamma {C_D}^2 + 1)}]} \quad (38)$$

Thus the effective or pseudo-diameter is proportional to the square root of the upstream pressure p_1.

3. OpenFOAM

The dissertation of this section is retrieved from Samel (2011), Winter (2013) and OpenFOAM user guide (2015). The Open Source Field Operation and Manipulation (OpenFOAM) is a open source CFD software package. All the codes in OpenFOAM are written in C++ and it has object oriented programming interface. It provides variety of solvers both pre-processing and post-processing solvers with several finite volume solvers with structured and non-structure grids. These solvers are capable of solving the steady, unsteady, compressible, incompressible, laminar and turbulent flow using FV numeric that solve a system of partial differential equations (PDE) within three dimension but can be used for one or two dimension case as well. The computational method described in this chapter is used to develop an OpenFOAM solver called rhoCentralFoam. OpenFOAM does not possess any graphical interface and all the input parameters are set up on some text files called dictionary files. The post-processing and results viewing are performed by another interface called ParaView. Any OpenFOAM case structure contains three major folder called "0", "constant" and "system" respectively. The "0" folder contains all the initial field definition like pressure, temperature, velocity, turbulent energy, dissipation rate etc. The "constant" folder contains full information about the geometry and boundary conditions. The "system" folder contains information about the solver control. Figure 9 depicts the case structure of OpenFOAM. OpenFOAM enables the solver structure to yield a system of equations. As mentioned in OpenFOAM user guide (2015) the equation $\partial \rho U / \partial t + \nabla \bullet \varphi U - \nabla \bullet \mu \nabla U = -\nabla p$ is coded for example as:

```
solve ( fvm::ddt(rho, U)
 + fvm::div(phi, U)
 - fvm::laplacian(mu, U)
 = =
 - fvc::grad(p)
) ;
```

The command above converts the PDE continuity equation to a sets of equation of the matrix form [P][x]=[q] where [P] consist of the coefficients extracted from the discretization process of different terms like gradient, convective, Laplacian and [x] represent the variable matrix. The matrix [q] consist of source terms.

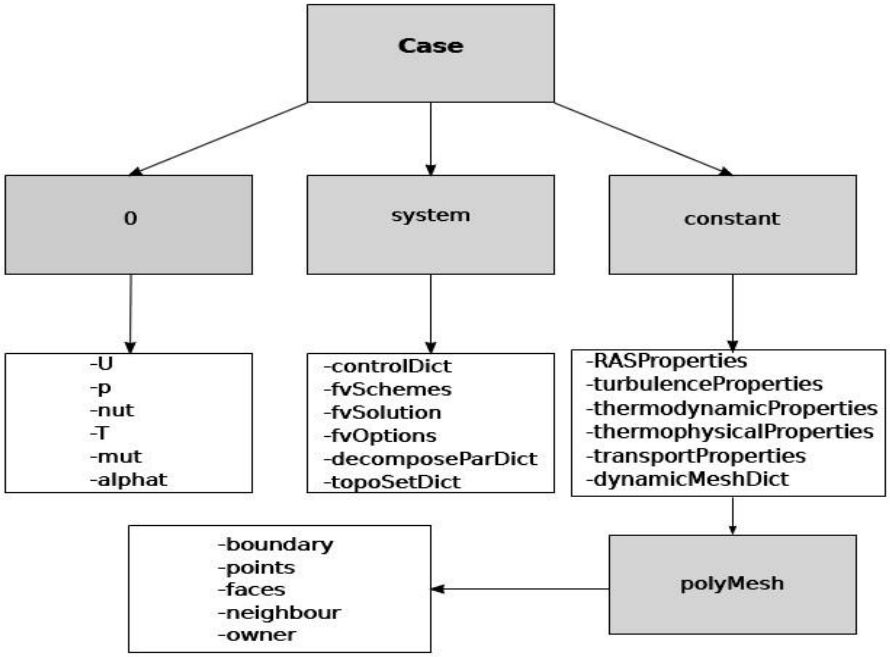


Figure 9: Case structure in OpenFOAM (Winter, 2013)

3.1 rhoCentralFoam

This section is inspired from Samel (2011). Based on the computational methodology described in chapter 2 the density based OpenFOAM solver rhoCentralFoam is developed. Inviscid equations are solved explicitly in rhoCentralFoam for a predicted a variable value whereas the viscous equations are solved by time splitting method. The actual inviscid equations is then corrected implicitly by associating diffusion terms. First of all the ρ, T and u are calculated at the cell face in both f+ and f- directions which are interpolated from the cell centroids and replaced in the convective terms $\nabla.(\mathbf{u}\rho)$ calculation. Thus the continuity equations is solved for density ρ. The predicted velocity value u is determined explicitly from the momentum equation like below:

$$\frac{(\rho\dot{u}) - (\rho u)^{n+1}}{\partial t} + \nabla . [u(\rho u)] + \nabla p = 0 \qquad (39)$$

$$\dot{u} = \frac{(\rho\dot{u})}{\rho} \qquad (40)$$

To determine the corrected velocity value implicitly at the next time step (n+1), this predicted value $\dot{u}$ is used by the viscous momentum equation below:

$$\frac{(\rho u)^{n+1} - (\rho\dot{u})}{\partial t} - \boldsymbol{\nabla} . (\mu\nabla\mathbf{u}) = 0 \qquad (41)$$

The solution of the energy equation follows the similar procedure. $(\rho\dot{E})$, the energy predictive value is calculated from the inviscid energy equation:

$$\frac{\partial(\rho E)}{\partial t} + \boldsymbol{\nabla} . [\mathbf{u}(\rho E)] + \boldsymbol{\nabla} . [\mathbf{u}p] + \boldsymbol{\nabla} . (\mathbf{T} . \mathbf{u}) = 0 \qquad (42)$$

The temperature T is determined using the equation mentioned in (8) considering the parameter ρ,u and E which is used to corrected energy equation:

$$\frac{\partial(\rho c_v T)}{\partial t} + \boldsymbol{\nabla} . (k\boldsymbol{\nabla} T) = 0 \qquad (43)$$

Then the pressure is updated by the equation of state for the ideal gas. To determine the viscosity Sutherland's law of viscosity is used as mention on chapter 2.

3.2 OpenFOAM simulation

In this section three different simulation case will be studied in OpenFOAM to test the solver capability.

3.2.1 Shock tube case set-up

Based on the shock tube case information and boundary condition mentioned earlier in chapter 2, we will simulate a shock tube case of similar set up here and extract the results for four different grid resolutions. Note that this section only contains a brief and concise information about the case set up information. Detail of shock tube case set up including all CFD codes in OpenFOAM are attached in **Appendix A**.

3.2.1.1 Case Description

To solve all fluid properties such as pressure, temperature, density, velocity, a fully structured mesh of the geometry will be create first to solve all fluid properties such as pressure, temperature, density, velocity by using the rhoCentralFoam. The geometry is three dimensional with one dimensional solution domain shown in Figure 10.

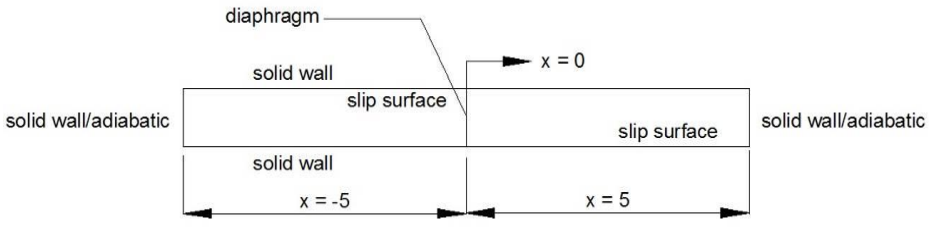


Figure 10. Geometry and Boundary conditions (Drawn by Talukdar)

3.2.1.2 Mesh Generation with `blockMesh`

The computational mesh required is created by using the OpenFOAM utility `blockMesh` after performing necessary edition to the file `blcokMeshDict`. Thus, the geometry is formed based on corner points in a quadrilateral block which is meshed with hexahedral elements. The resulting structure is presented on Figure 11.

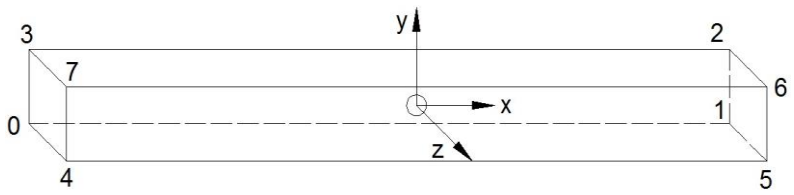


Figure 11. Block structure and numbering of vertices (Drawn by Talukdar)

Blocks (in this case only one) are instructed in the section `blocks`. Each block comprise of eight defined vertices oriented in a correct order. Also the number of cells in x-, y- and z- direction are defined in the entries. In this case, the number of cells in x- direction is set 100 and 1 in both y- and z- directions. The boundary faces are defined under section `patches`. Names and types of patches are also defined. In this case active patches are the side patches that is patch 1265 and 0473. Empty patches are 0154, 5674, 3762, 0321 which actually leads this case to a one dimensional case.

3.2.1.3 Initial and Boundary Conditions

Initial and boundary conditions are set in the field files (in this case U, p &T) in 0 directory. For the setup of initial conditions for p and T `internalField` is set to `nonuniform`. For the setup of the initial conditions of U `internalField` is set to `uniform`.

For p & T:

Patch	1265	0473	0154	5674	3762	0321
Type	zeroGradient	zeroGradient	empty	empty	empty	empty

Also for U:

Patch	1265	0473	0154	5674	3762	0321
Type	zeroGradient	zeroGradient	empty	empty	empty	empty

3.2.1.4 Solver Controls

In the file `system/controlDict` the `startTime` is set to 0 and `endTime` is set to 0.007 since the solution is a transient one. Also `writeInterval` is set to 0.001. The OpenFOAM command `rhoCentralFoam` is used to execute the case.

3.2.1.5 Post-processing

The results are post processed in the ParaView after executing the OpenFOAM command `paraFoam`. The results can be extracted by using the OpenFOAMs command `sample` as an excel files. The simulations and results of different variables output are extracted for different grid resolutions like 25, 50, 100, 500. Figure 12 depicts the typical ParaView post processing results for 500 grids for example.

3.2.1.6 Results

Output shock tube simulation data has been extracted from OpenFOAM for four different grid resolution and plotted against the shock tube axis to investigate the behavior of the solver. Figure 13, 14, 15 and 16 represent the output density, pressure, temperature and velocity field for the one dimensional grid resolution 500, 100, 50, 25 respectively.

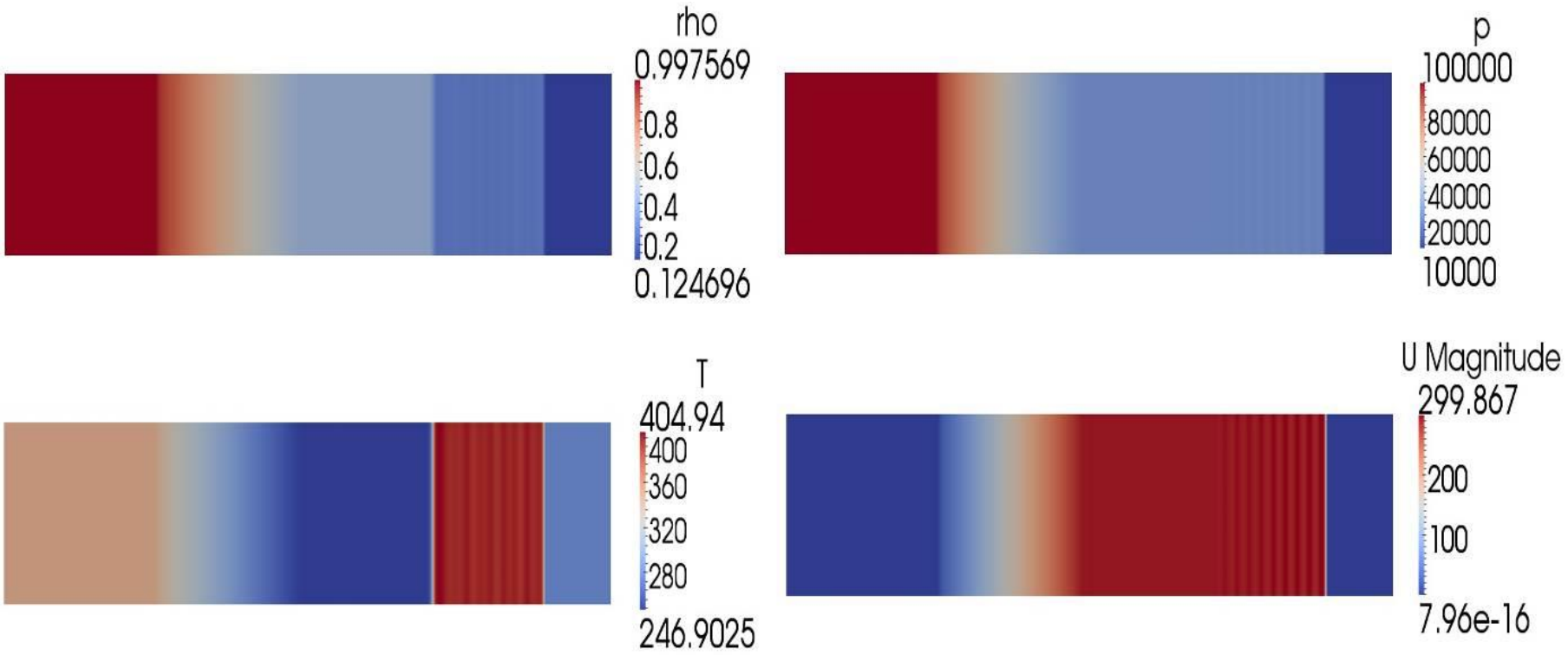


Figure 12. Density, pressure, temperature and velocity field for shock tube case in OpenFOAM for 500 grids (Prepared by Talukdar)

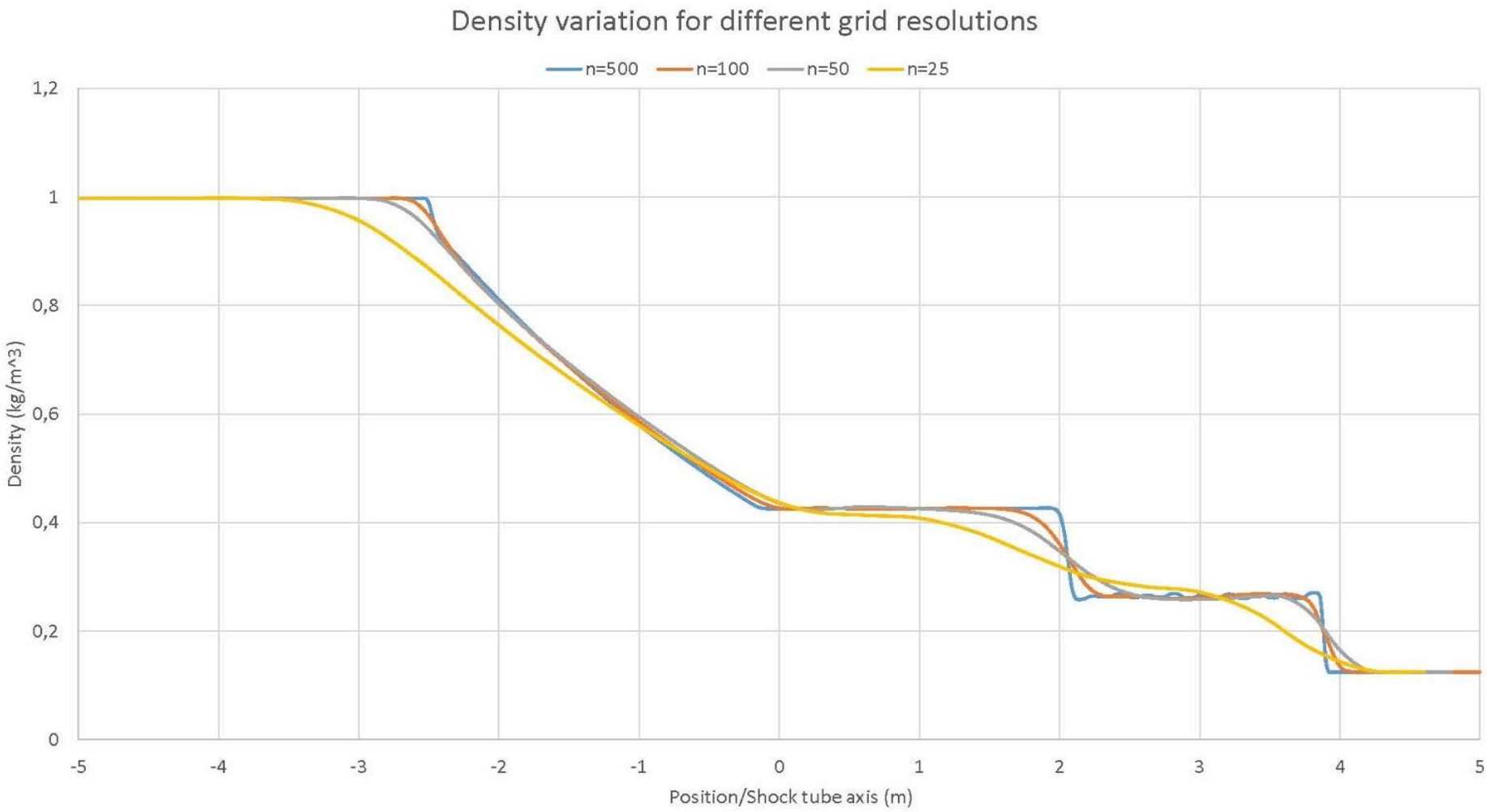


Figure 13. OpenFOAM results of the shock tube case set up for the density and it's variation for different number of grids.

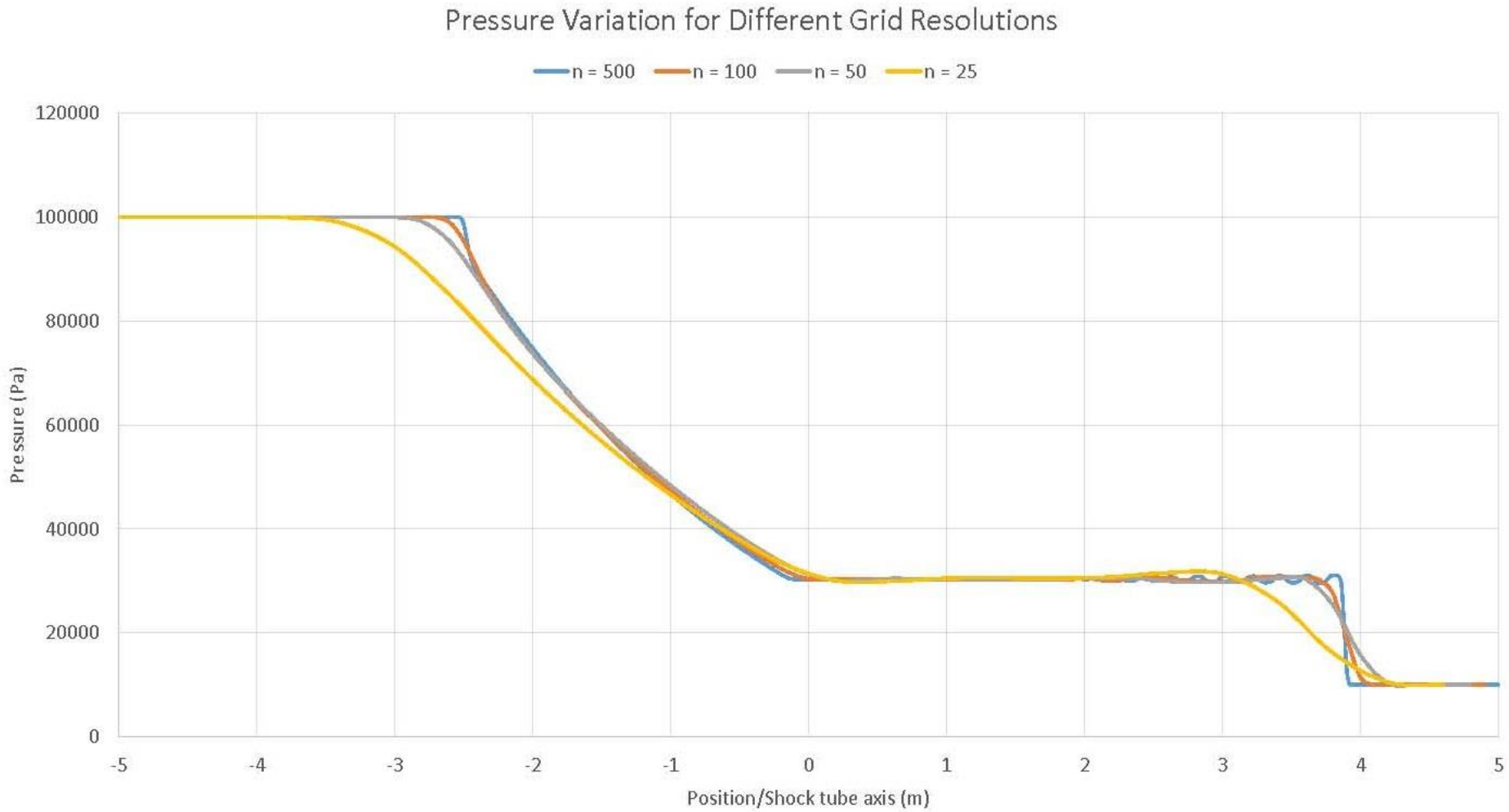


Figure 14. OpenFOAM results of the shock tube case set up for the pressure and it's variation for different number of grids.

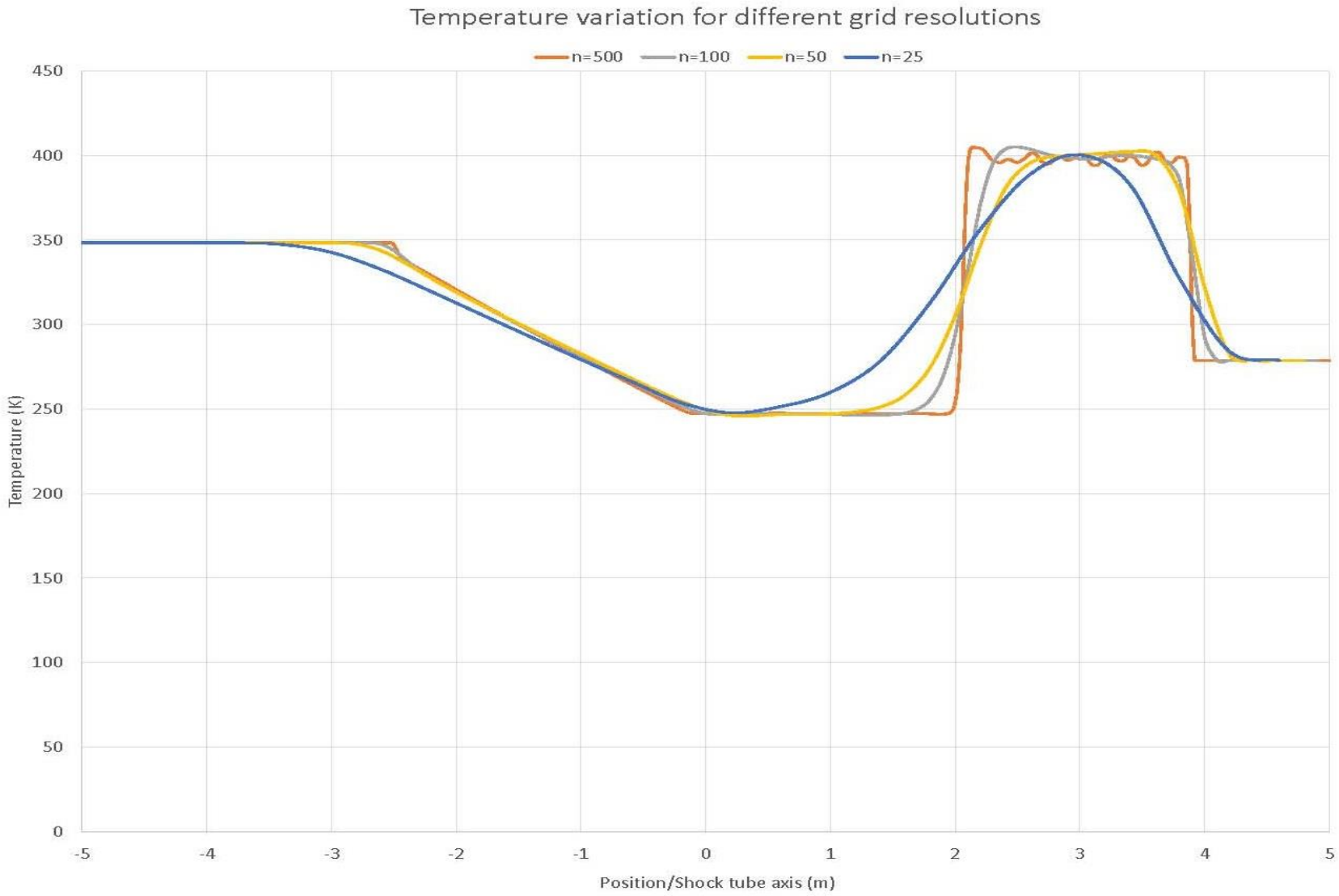


Figure 15. OpenFOAM results of the shock tube case set up for the temperature and it's variation for different number of grids.

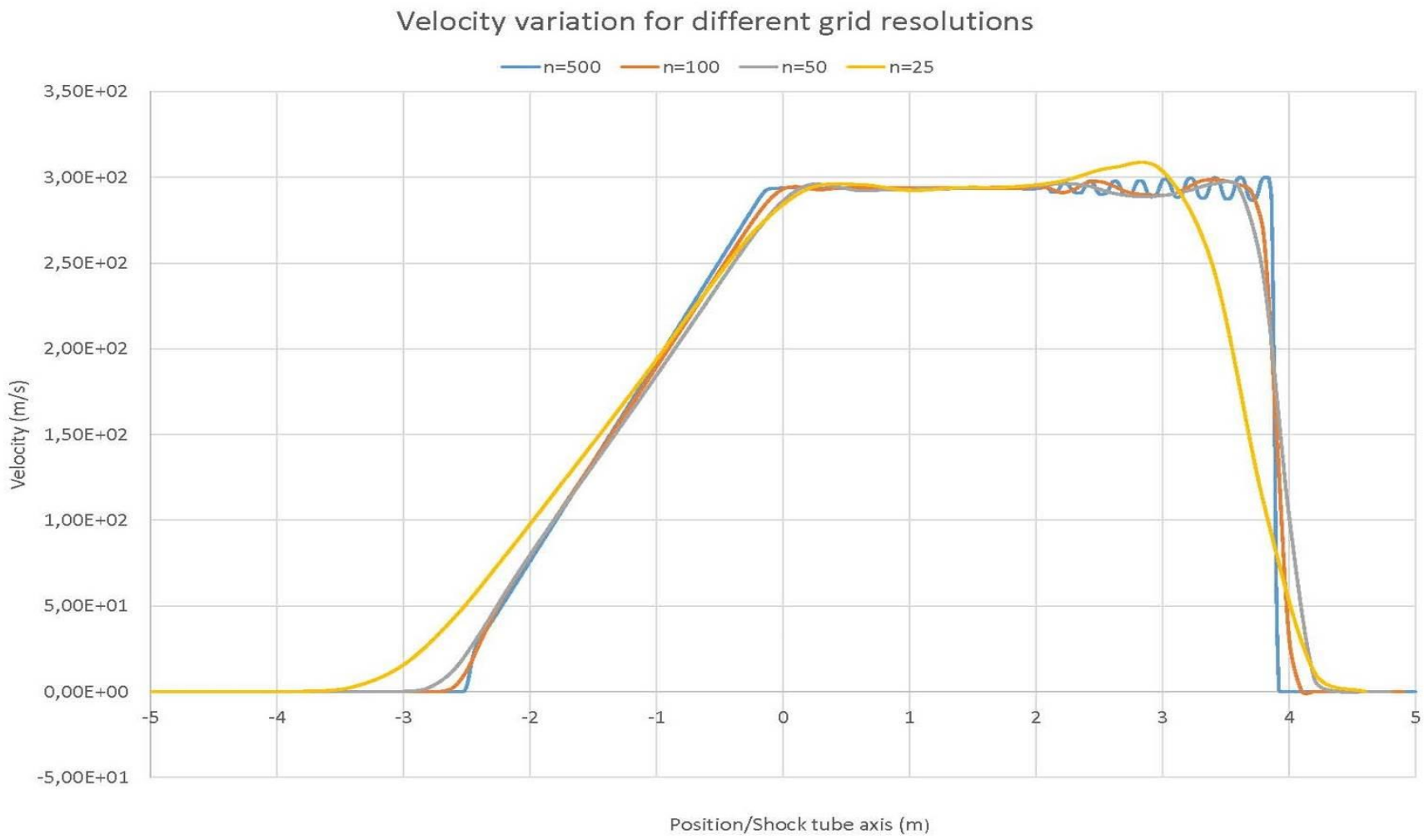


Figure 16. OpenFOAM results of the shock tube case set up for the velocity and it's variation for different number of grids.

3.2.2 Ladenburg case set-up

The case above is simulated as an axisymmetric problem with the domain configuration of 30 mm length and 10 mm height which is equal to the orifice diameter. The domain is meshed both along the length and in the radial direction for three different grid resolution such as 60x20, 90x30 and 120x40. Figure 17 demonstrates the block geometry and boundary conditions for the Ladenburg case set up. The detail in formation on this case setup is provided on **Appendix B**. In both of the three cases the solver should produce a solution near the Mach disk location. The CFL number for the solver is set to 0.5 and the end time is set to 2ms approximately i.e. it takes approximately twenty times of the time that a particle would take to travel along the length of the domain with the discharge velocity.

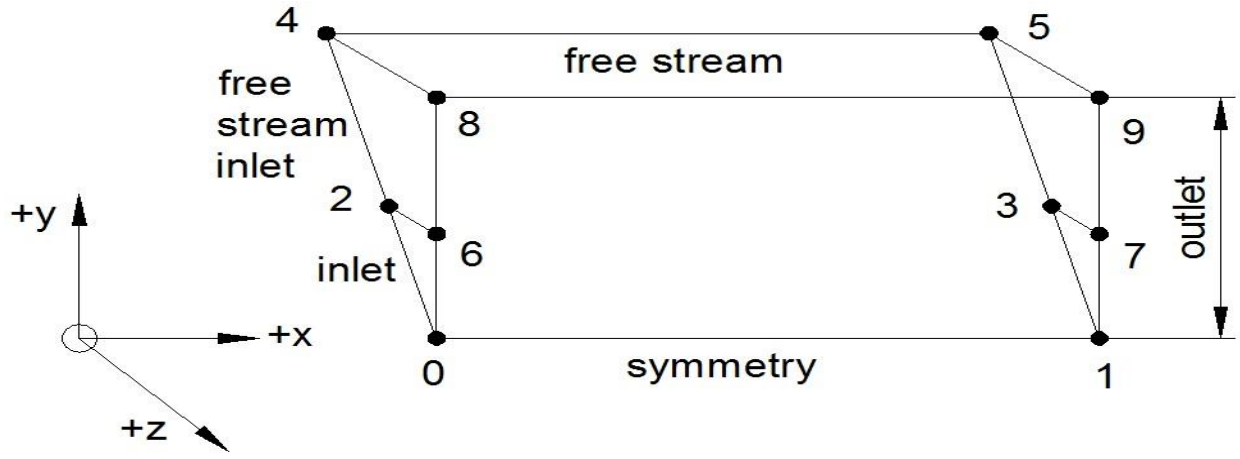


Figure 17. Block geometry and boundary conditions with vertices for Ladenburg case by OpenFOAM.

The simulation results are given below for different grid resolution as compared to the experimental results in figure 18, 19 and 20. The location and diameter of the Mach disk also can be determined from the pressure graph for different grid resolutions. The location of the pressure drop point serves as a significant indicator of the position of Mach disk. Figure 21 below shows the pressure output of OpenFOAM for different grid resolutions.

By analyzing the figure and experimental data and the numerical output from OpenFOAM, it can be derived that the position and diameter of Mach disk are dependent on grid resolution which are given on the table 3.

Table 3. Comparison of shock structure parameter for different grid resolution

Grid Resolution	Location of Mach disk x_m	Height of triple point r_m	Diameter of the Mach disk d_m
60x20	15.6	0.3	0.6
90x15	15.3	0.9	0.18
120x40	13.8	1.8	3.6

From the table above, it can be derived that the position of triple point (13.8, 1.8) for highest grid resolution is very adjacent to the experimental results for triple point location (13.7, 1.7).

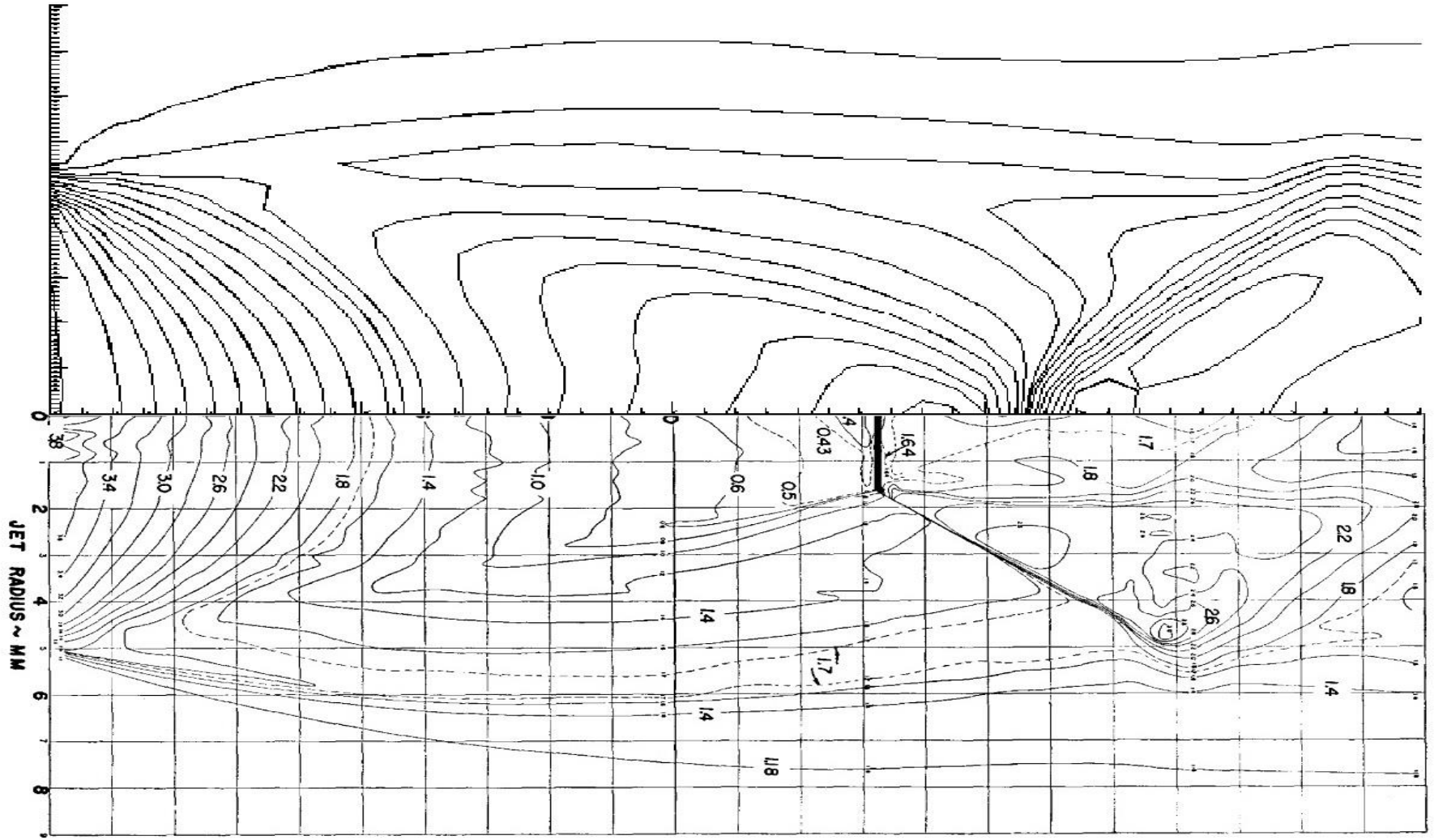


Figure 18. Comparison of the location of mach disk with density contour in Ladenburg jet; OpenFOAM simulation for 60x20 grid resolution on top and Ladenburg experimental data on bottom.

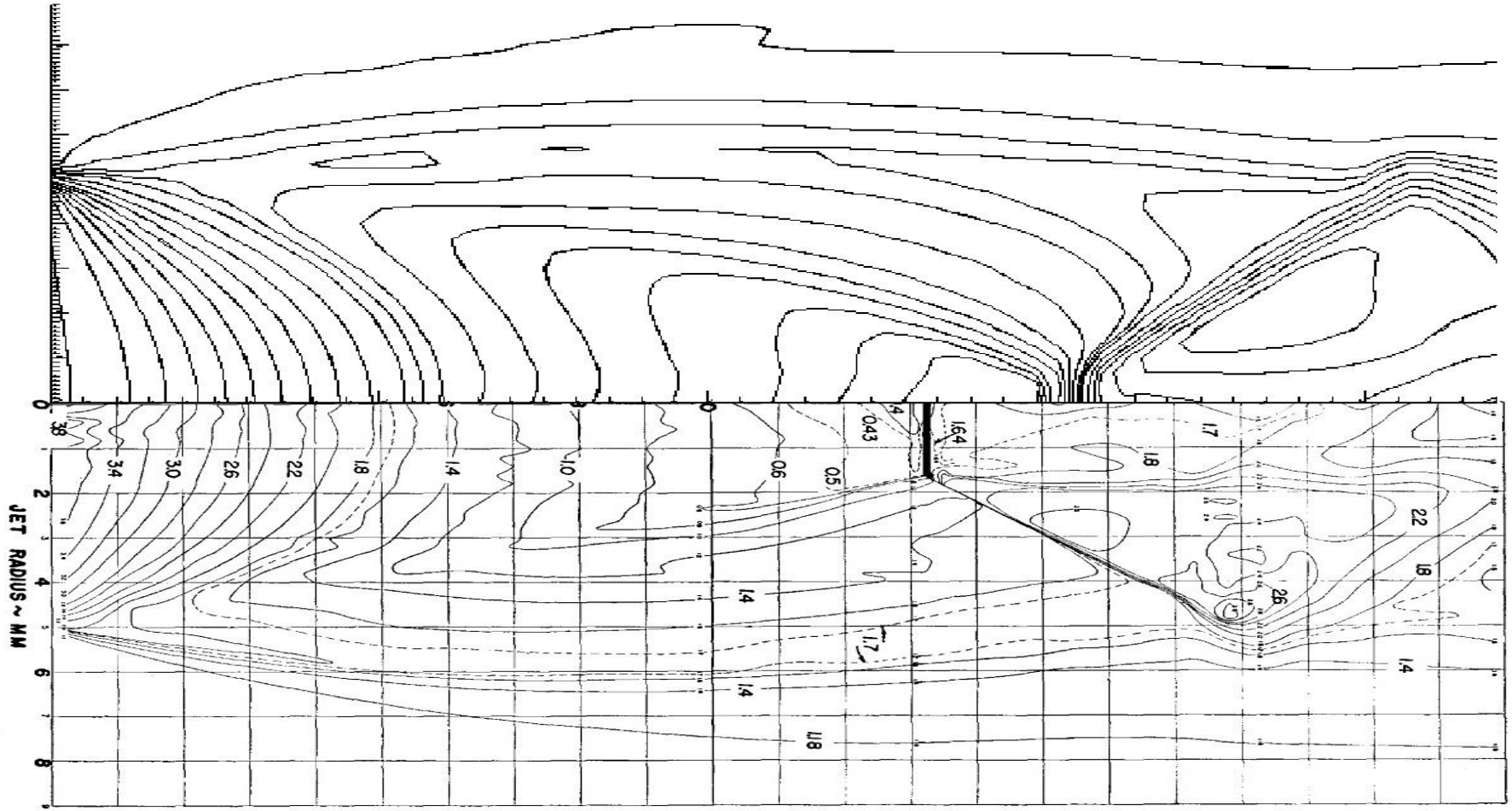


Figure 19. Comparison of the location of mach disk with density contour in Ladenburg jet; OpenFOAM simulation for 90x30 grid resolution on top and Ladenburg experimental data on bottom.

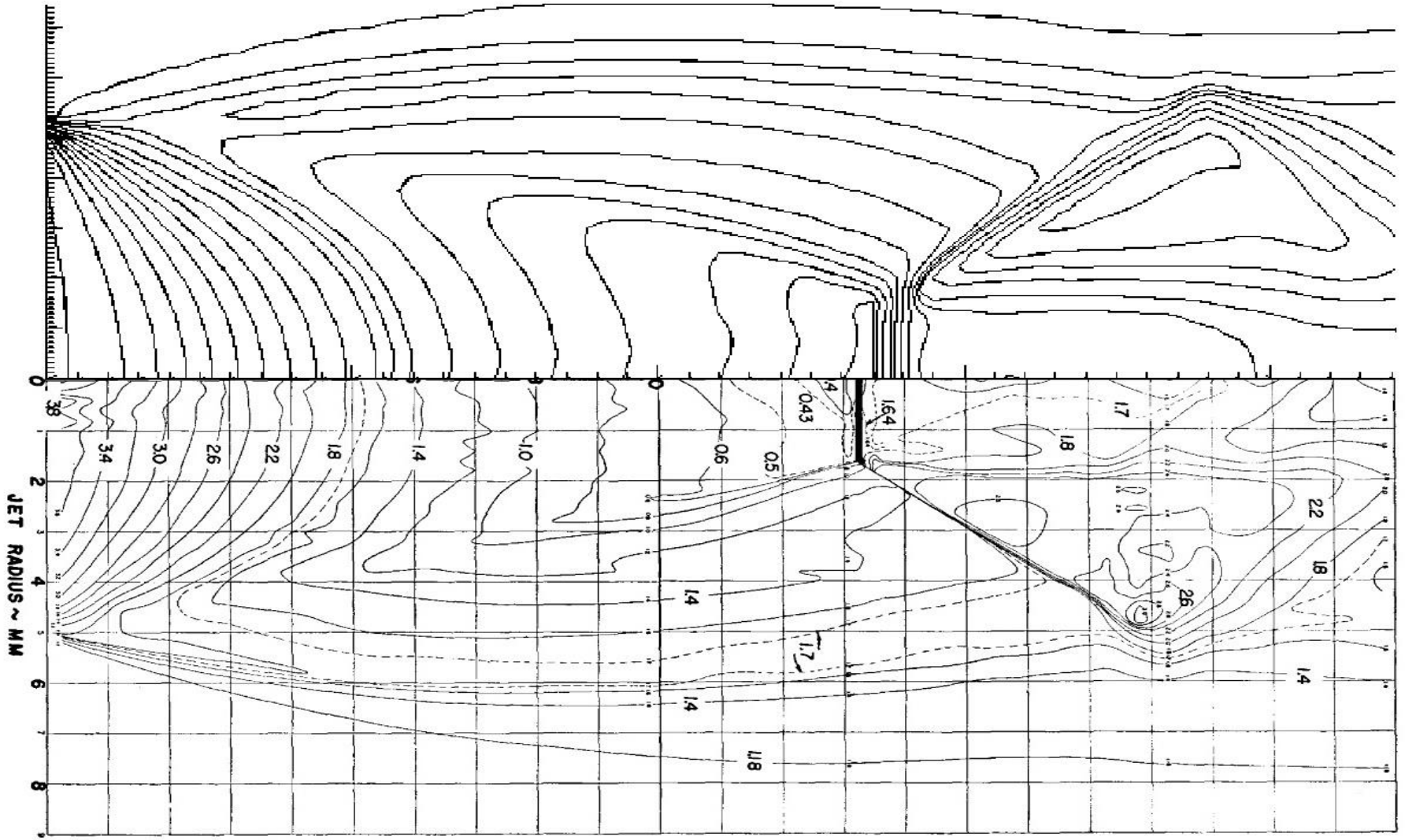


Figure 20. Comparison of the location of mach disk with density contour in Ladenburg jet; OpenFOAM simulation for 120x40 grid resolution on top and Ladenburg experimental data on bottom.

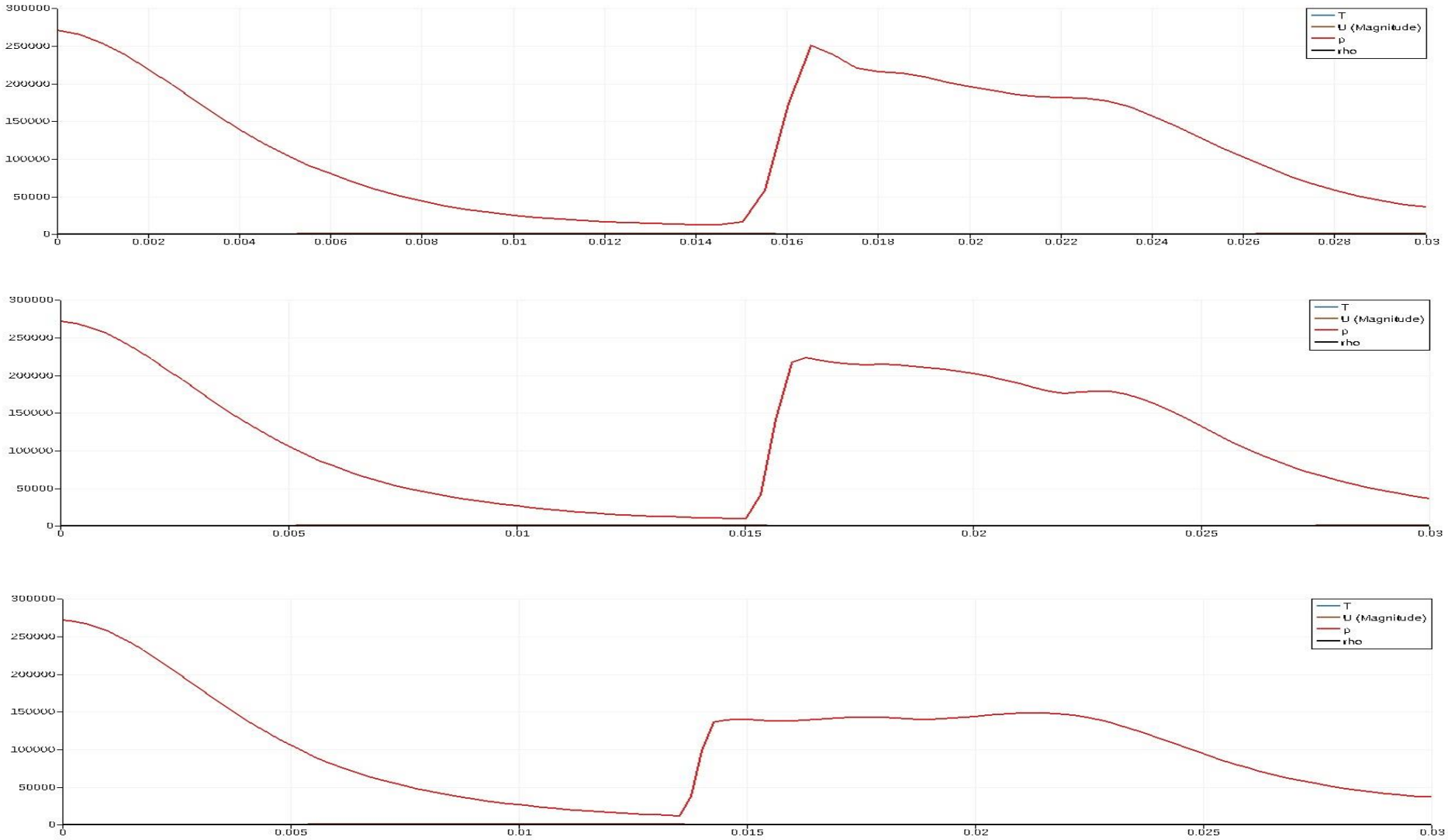


Figure 21. OpenFOAM pressure output for 60x20, 90x30 and 120x40 grid resolution respectively starting from top. The location of the lowest pressure point could also be an indicator for the location of mach disk.

3.2.3 Jet simulation case set-up by OpenFOAM

The OpenFOAM simulation for the leakage of reservoir at pressure 20 bar is performed in this section up to the first cell that i.e. the position of first Mach disk within a region considered near the leak is about the dimension 600 mm x 300 mm x 26 mm. The region consist of two blocks of which one is for the inlet and another is for region above the inlet. The number of cells are increased and enhanced near the inlet region by stretching the grids towards the inlet (see **Appendix C** for detail). The block structure and boundary conditions for the 20 bar leakage case is similar to that of Ladenburg case shown in figure 17.

3.2.3.1 Initial and boundary conditions

The initial and boundary conditions for the pressure, temperature and velocity are set according to the following tables (see **Appendix C** for detail).

For p:

	inlet	outlet	freestream	freestreamInlet
Type	fixedValue	waveTransmissive	waveTransmissive	zeroGradient
Value/Gradient	uniform 1.0566×10^6	uniform 101325	uniform 101325	-

For T:

	inlet	outlet	freestream	freestreamInlet
Type	fixedValue	zeroGradient	Inlet0utlet	fixedValue
Value/Gradient	uniform 233.33	-	uniform 283	uniform 283

For U:

	inlet	outlet	freestream	freestreamInlet
Type	fixedValue	Inlet0utlet	Inlet0utlet	fixedValue
Value/Gradient	uniform (306.03,0,0)	uniform (0,0,0)	uniform (0,0,0)	uniform (0,0,0)

3.2.3.2 Solver controls

In the file `system/controlDict` the `startTime` is set to 0 and `endTime` is set to 0.003s since the solution is a transient one. Also `writeInterval` is set to 0.00002. The OpenFOAM command `rhoCentralFoam` is used to execute the case.

3.2.3.3 Post-processing output

As the flow after the first Mach disk is turbulent in nature, the mean of the the variables are taken for the results. Figure 22 below shows the transition variable pMean as an example of the simulation result. Detail result will be demonstrated and explained on discussion segment.

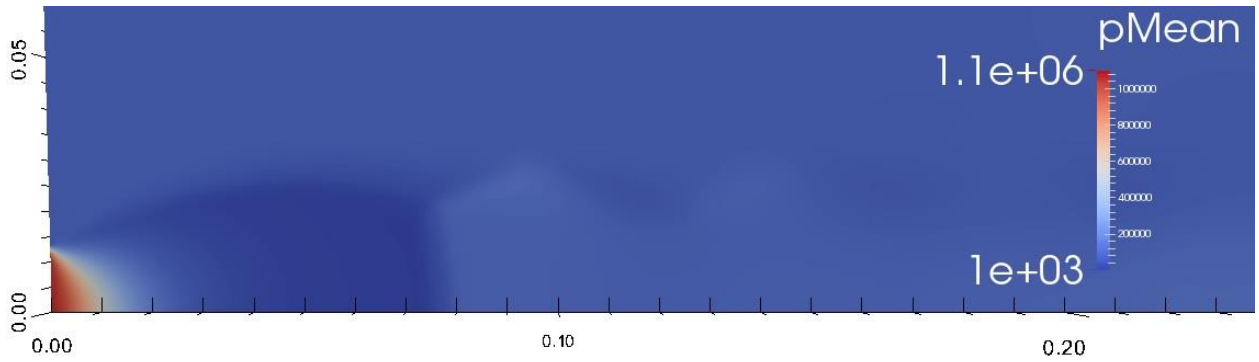


Figure 22. OpenFOAM simulation output for mean pressure (pMean).

3.2.3.4 Calculation & Results of simplified methods (KFX & FLACS)

In this section we will calculate the jet flow properties considering the formulas of the jet model mentioned on simplified method which are the theoretical foundation for the KFX and FLACS software and compare them with the OpenFOAM simulation results for the same case to find out which software yields these properties that are adjacent to the OpenFOAM simulation results. As mentioned earlier in section 2.5, both KFX and FLACS adopt different variants of simplified model. Let us consider the following input parameters for the simplified model 1 (KFX) and simplified model 2 (FLACS) that will be used for the jet modeling and extraction of output:

Diameter of the leak D = 0.025 m

Ration of the specific heats for the gas $\gamma = c_p/c_v$

Molar mass of the gas M = 29 kg/kmol

Gas constant R = 8.314 kg m^2/K kmol s^2

Reservoir or vessel pressure p_0 = 20 bar = 20x10^5 Pa

Reservoir or vessel temperature T_0 = 280 K

Atmospheric pressure = 1 bar = 1x10^5 Pa

Discharge coefficient C_d = 0.85 (considered in practical implementation)

The simplified models adopted by KFX (computit.no, 2015) and FLACS (gexcon.no, 2015) yields the leak output at the sonic conditions and the results for temperature, velocity, density and diameter are demonstrated in Table 4 and Table 5 respectively.

Table 4. Leak output at the sonic conditions used as a boundary for OpenFOAM simulation.

Conditions at the leak output or OpenFOAM BC	Pressure (Pa)	Temperature (K)	Velocity (ms^{-1})
Values	1.0566 x10^6	233.33	306.03

Table 5. Results of KFX and FLACS based on simplified methods.

Properties	Density (kgm^{-3})	Temperature (K)	Velocity (ms^{-1})	Diameter (m)
KFX (SM1)	2.2	159	493	0.048
FLACS (SM2)	1.3	263	183	0.103

The results obtained here will be compared to the OpenFOAM simulation result on the discussion part.

4. Discussions

4.1 Shock tube simulation results

- From the shock tube simulation results on chapter 2, it is obvious that with the elevation of the number of mesh or grid resolution the solution fields for different variable pressure, temperature, density and velocity tends to converge to the exact analytical solution. That is, as the mesh size tends to zero the error also tends to zero. For example, in figure 14, between the shock tube axis point x = -4 to x = -3.5 the bent curve tends to lose its curvature and converges towards a straight line almost as the number of grid elevated. Also in the same figure the pressure curve between p = 30000 to p = 10000 converges to a straight line almost which looks pretty similar to the analytical solution.

- At lower grid resolution, numerical diffusion is relatively higher which is dissipated by the differencing scheme whereas the diffusion is too low at high grid resolution to smear out the oscillations or wiggle produced by the differencing scheme of the rhoCentralFoam solver. Euler simulation considers the time and space segregated into discrete and non-overlapping grid with the discretization of partial differential equations of motions which are the Navier-Stokes equations into a set of equations of finite different approximation of derivatives. Original partial differential equation have very less diffusion as compare to these discrete equations (Wikipedia, May 2015). This can be

the reason of the slightly different behavior of simulated system as compare to the original physical system. The characteristics of the system and the type of discretization scheme used in the solver defines this different behavior. Most of the CFD solver are conceived to lessen the numerical diffusion as much as possible to maintain the quality of the simulation and achieve high efficiency. But in some cases, the discretization scheme intentionally adopt or add diffusion into the system to get rid of mathematical singularities which is the undefined condition of a mathematical function at a given point (Wikipedia, May 2015) that can be the possible reason of oscillating behavior of the solution at higher gird for example the shock tube solution at higher grid points.

- As the size of the mesh tends to zero the exact partial differential equation (PDE) should be equivalent to its discretized equation to maintain the solution it's consistency. As implied on pg. 294 in (Versteeg, 2007), discretization error tends to zero as the mesh size and time step goes to zero. From the results it is quite obvious that the numerical scheme of the rhoCentralFoam solver maintains its consistency.
- As implied on pg. 287 in (Versteeg, 2007) that for maintaining the convergence of the numerical solution, the discretized equations has to be equivalent to the exact solution of the flow differential equation. Convergence can be investigated numerically by comparing the results for different grid resolutions. Truncation error of the discretization scheme governs the rate of convergence. Convergence are obvious from our result of different grid resolutions.
- As mentioned by Versteeg (2007) on pg.141 that the conservativeness is the fundamental properties of the discretization scheme i.e. conservation law or physical principle should be maintained at all discrete level. In our case, the central differencing scheme of the solver implies physically consistent expressions for the evaluation of the convective and diffusive fluxes at the control volume faces which are obvious from the results.
- The solver used here is able to predict the contact discontinuity correctly. It is explicit and unconditionally stable. All the temperature, pressure, velocity and density profile before and behind the discontinuity is not flat in the simulation as in the exact solution.

Furthermore, the solution produces bulge in higher grid resolution. The discretization scheme and the simulation time step are also the deciding factor for the magnitude and the frequency of these oscillations.

- It can also be noted from the plot that the exact solution of the shock tube case is about two mesh point ahead than that of numerical one. As explained by Huynh (1995), the solution formed by the numerical scheme produce the expansion fan first then the contact discontinuity. So, the shock is two cells behind by the time it is produced. Also it is noticeable that if there is not any correction in the flux-difference splitting, the solution is independent of time because of the balance of fluxes. As mentioned by the Huynh (1995), the simple-wave upwind flux or the flux-difference splitting with admissibility condition are employed near a discontinuity.

4.2 Ladenburg case simulation results

- From the simulation results, it is obvious the simulation results tend to the actual experimental results with the increase of grid resolutions. The difference among the results are obvious this time as the solver controls the solution in two direction this time.

- According to Velikorodny (2012), the vital parameter that determines the under expanded jet structure is the pressure ration P_0/P_a where P_0 is the stagnation pressure in the reservoir and P_a is the ambient pressure. This shock structure also depends on the shape or geometry of the nozzle and the characteristics of the expanded gas. Velikorodny (2012) also reported that a complex shock structure is produced for $P_0/P_a > 15$. In our case as mentioned earlier, due to the complex shock structure the reflection of the incident shock becomes irregular and a Mach disk pattern is formed. The diameter of the Mach disk is inversely proportional to specific heat ratio γ (Velikorodny A.2012). After the Mach disk Mach number $M<1$ (velocity relative to sound the velocity) and the flow is subsonic and before Mach disk $M>1$ and the flow is supersonic. All the discontinuities coincides at the triple point and it forms the new origin of the slip line. The size of each shock cell is a function of the pressure ratio and the exit Mach number. The location of the Mach disk is a function of γ, stagnation temperature, condensation nozzle

configuration and absolute pressure level. Also the size of the Mach disk diameter depends on molecular structure and condensation. The nozzle exit diameter is become less as compare to the size of the Mach disk. Therefore, the nozzle exit can be a boundary for maximum jet velocity and the mass and momentum fluxes at this point. The relation of the Mach disk distance and diameter with the nozzle exit diameter can be given by (67) and (68) that indicates their dependence on γ and pressure ratio.

$$\frac{x_m}{d_e} = \frac{1}{2}\sqrt{\gamma} * \sqrt{\frac{P_e}{P_a}} * \sqrt{\frac{\gamma+1}{\gamma-1}} \tag{44}$$

$$\frac{d_m}{d_e} = kx_m * \frac{1}{2}\sqrt{\gamma} * \sqrt{1-\frac{\gamma+1}{\gamma}} * \sqrt[4]{\frac{\gamma-1}{\gamma+1}} \tag{45}$$

Where

x_m = location of the Mach disk

d_m = diameter of the Mach disk

k = empirical constant accounted for the growth of mixing layer

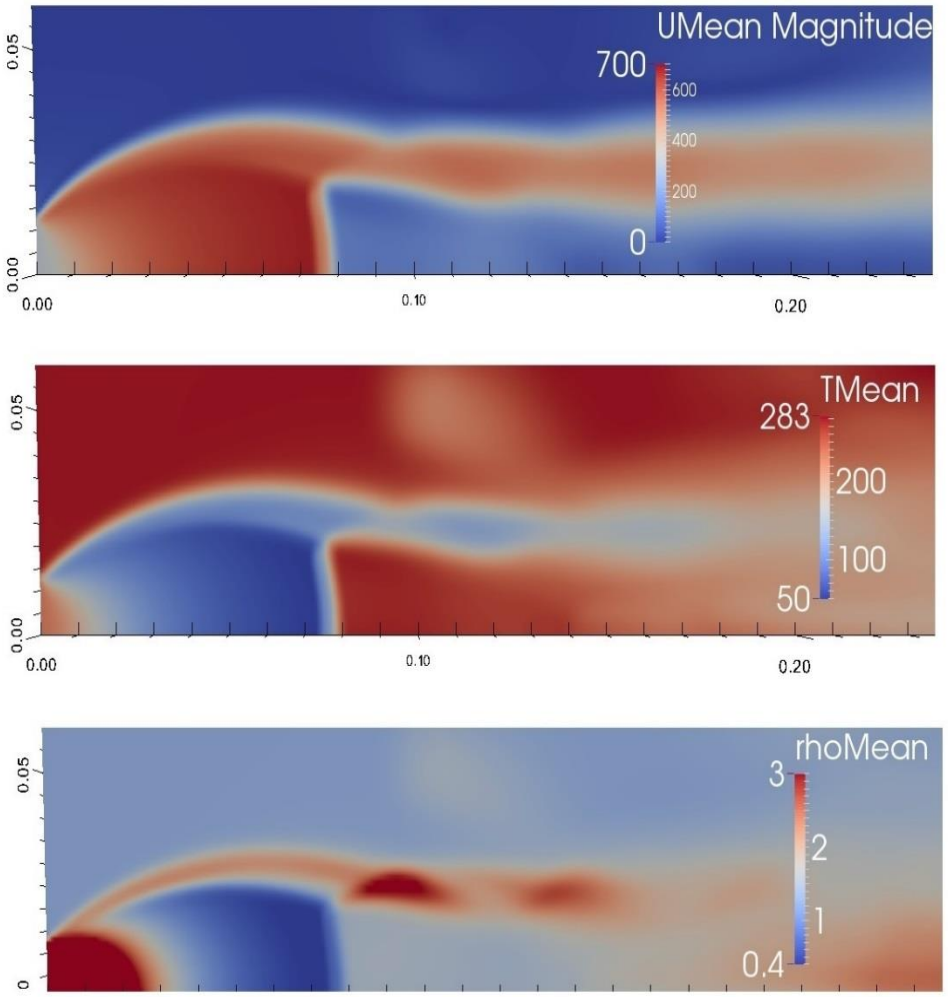
P_e = exit static pressure

P_a= ambient pressure

- The rhoCentralFoam solver has demonstrated competence in capturing two dimensional shock structure considering all the physical phenomena involved and parameters mentioned above. The total variation diminishing (TVD) properties of this solver enables it to handle any kind of spurious numerical oscillations.

4.3 Simplified method and jet simulation

The comparison of the flow properties of OpenFOAM jet simulation with KFX and FLACS are demonstrated on figure 23. The KFX yields a higher velocity with a smaller diameter hole and FLACS yields a lower velocity with relatively bigger diameter hole. In both cases mass flow rate are equal. The thin velocity layer of the downstream on OpenFOAM simulation seems adjacent to KFX results in terms of velocity. But still momentum is a key factor to be investigated in addition to other variables. The temperature at the downstream where the flow starts smearing out seems adjacent to the FLACS results. The density at the downstream seems adjacent to the KFX result. Also, momentum plays a vital role in for the density results. So, overall it is difficult to comment about the reliability of the simplified models adopted by KFX and FLACS based on one simulation test at a certain leakage pressure. But this approach demonstrates prominence for further testing, analysis and assessment.



Jet Model Output		
Parameter	**KFX**	**FLACS**
Density (kgm^{-3})	**2.2**	**1.3**
Temperature (K)	**159**	**263**
Velocity (ms^{-1})	**493**	**183**
Diameter (m)	**0.048**	**0.103**

Figure 23. Comparison of OpenFOAM jet simulation with jet model output for KFX and FLACS

5. Conclusions

- Although the solution field of one dimensional shock tube still adjacent to the analytical solution, it produces numerical oscillation at high grid resolution. So the discretization strategy of the rhoCentralFoam solver can be enhanced further to adapt this oscillations.
- It can be stated the rhoCentralFoam shows proficiency to capture shock structure produced by supersonic air jet release. Even for a coarse grid the simulated shock structure demonstrates a clear resemblance with the basic shock structure.
- The position and diameter of the Mach disk are in good agreement with the experimental results. Also the length of the first cell of jet seems equivalent to that from OpenFOAM simulation.
- The result achieved downstream of the Mach disk can be used as a boundary condition for the future far field simulation.
- It is almost impossible to express exactly the value of pressure, temperature, velocity and density at the downstream of the air jet simulation. But these properties can be predicted which are close to the exact value.
- The solver rhoCentralFoam can be employed preliminary for the conceptualization of the complex jet structure before simulating it with KFX or FLACS. Also, the result of any tool does not certify the real picture of jet in practice because of uncertainty related to the simulator such as mesh size and initial boundary conditions.

6. Future work

Although several task has been accomplished in this thesis, there are lot of simulation case results yet to achieve which could be performed in following ways:

- The solver rhoCentralFoam could be enhanced further by adopting non-oscillatory scheme and TVD properties should be improved to lessen the numerical oscillations at higher grid resolutions.
- More versatile cases with different set of parameters could be run.
- The solver rhoCentralFoam could be enhanced further to adopt complex geometries and leak sequences.
- The jet fluid considered for this thesis work is air. The solver rhoCentralFoam could be enhanced further to provide support for multicomponent gases as well.
- The solver rhoCentralFoam does not consider the ambient fluid entrainment on the jet simulation. So, further approaches could be developed to check ambient fluid entrainment.

7. References

Addy A. L. (1981). AIAA Journal 19 1.

Bayeh A. C. (2009). Analysis of Mach Disks from an Underexpanded Nozzle Using Experimental and Computational Methods, 47th AIAA Aerospace Sciences Meeting Including The New Horizons Forum and Aerospace Exposition, 5 - 8 January 2009, Orlando, Florida.

Birkby P., Dent J.C. and Page G.J. (1996). CFD prediction of Turbulent Sonic Underexpanded Jets, Preceedings of the 1996 ASME Fluids Engineering Summer Meeting. Part 2 (of3), pp 465-470.

Computit.no (2015), Kameleon FireEx (KFX), www.**computit**.no (Date checked: June 07, 2015)

Cumber P.S., Fairweather M., Falle S.A.E.G and Giddings J.R. (1995). Predictions of the Structure of Turbulen, Highly Underexpanded Jets" Journal of Fluids Engineering, vol 117, pp 599-604.
Hsu A. T. and Liou M.S. (1991). Computational Analysis of Underexpanded Jets in the Hypersonic Regime, Journal of Propulsion and Power , vol 7, no. 2, pp 297-299.

Dash S. M., Wolf D. E, and Siener J. M. (1985). AIAA Journal, 23 4.

Gary A. Sod (1978). A Survey of Several Finite Difference Methods for Systems of Nonlinear Hyperbolic Conservation Laws, Journal of Computational Physics 27, 1-31.

Gexcon.no (2015), Flame Accelerator Simulator (FLACS), www.gexcon.com (Date checked: June 07, 2015)

Gribben B. J., Badcock K. J., and Richards B. E. (2000). AIAA Journal 38 2.

Huynh H. T. (1995). Accurate Upwind Methods for the Euler Equations Society for Industrial and Applied Mathematics.

Katanoda H., Miyazato Y., Masuda M., and Matsuo K.(2000). Shockwaves 10 pp 95-101.

Ksibi Hatem and Moussa Ali Ben (2008). Numerical simulation of a one-dimensional shock tube problem at supercritical fluid conditions, International Journal of Physical Sciences Vol. 3 (12), pp. 314-320.

Kurganov Alexander and Tadmor Eitan (2000). New High-Resolution Central Schemes for Nonlinear Conservation Laws and Convection–Diffusion Equations, Journal of Computational Physics 160, 241–282.

Kurganov Alexander, Noelle Sebastian and Petrova Guergana (2001). Semidiscrete central-upwind schemes for hyperbolic conservation laws and hamilton–jacobi equations, Society for Industrial and Applied Mathematics.

Ladenburg R, van Voorhis CC, Winkler J. (1949). Interferometric studies of faster than sound phenomena. Part II, Analysis of supersonic air jets; 76:662–677.

Love E. S. et al. Woodling M. J., and Lee L. P.(1956). Boundaries of supersonic axisymmetric free jets NACA RM L56G18.

Love E. S. and Lee L. P (1958). Shape of the initial portion of boundary of supersonic axisymmetric free jets at large pressure rations NACA TN 4185.

Love E. S., Grigsby C. E.; Lee L. P., Woodling, M. J. (1959). Experimental and theoretical studies of axisymmetric freejets: NASA Technical Report, R-6, pp. 1-292.

Luis F. Gutiérrez Marcantoni, José P. Tamagnoa,d and Sergio A. Elaskar (2012), High Speed Flow Simulation Using OPENFOAM, Departamento de Aeronáutica, FCEFyN, Universidad Nacional de Córdoba, Av. Vélez Sarsfield 1601(5000), Córdoba, Argentina.

McDaniel J., Glass C., Staack D., and Miller C. (2002). In Processing of the 40 th AIAA Aerospace Sciences Meeting and Exhibit.
Menon N. and Skews. B. W. (2009). Effect of nozzle inlet geometry on underexpanded supersonic jet characterstics, School of Mechanical Engineering, University of the Witwatersrand.

Pack, D. C. (1948). On the formation of shock waves in supersonic gas jets, Q. J. Mech. Appl. Math. 1, pt 1, pp. 1-17.

Prudhomme S. M. and Haj-Hariri H. (1994). Investigation of Supersonic Underexpanded Jets using Adaptive Unstructured Finite Elements" Finite Elements in Analysis and Design, vol 17, pp 21-40.

Samel M. A. (2011). Numerical Investigation of Gas-Particle Supersonic Flow, Department of Mechanical and Industrial Engineering, University of Massachusetts Amherst.

Tsangtis S., Pappou Th. (2000). Analytical Solutions for the Unsteady Compressible Flow Equations Serving as Test Cases for the Verification of Numerical Schemes, Laboratory of Aerodynamics, National Technical University of Athens.

User Guide OpenFOAM, , OpenFOAM Foundation Ltd (2015). Last downloaded: (June 07, 2015).
URL http://foam.sourceforge.net/docs/Guides-a4/UserGuide.pdf

Vembe B. E., Rian K.E., Holen J.K., Lilleheie N.I., Grimsmo B. and Myhrvold T. (2001). Kameleon FireEx 2000 Theory Manual.

Versteeg H. K. and Malalasekera W. (2007). An Introduction to Computational Fluid Dynamics, 2nd Edition, Pearson Education Limited.

Wikipedia (2015). Numerical Diffusion (Last browsed: June 03, 2015).

Wikipedia (2015). Mathematical Singularities (Last browsed: June 03 2015).

Winter Magnus (2013). Benchmark and validation of Open Source CFD codes, with focus on compressible and rotating capabilities, for integration on the SimScale platform, Department of Applied Mechanics Division of Fluid Dynamics, Chalmers University of Technology.

Zhang Guang and Kim Heuy Dong (2014), Numerical simulation of shock wave and contact surface propagation in micro shock tubes, Department of Mechanical Engineering, Andong NationalUniversity.

Appendices

Appendix A: Shock tube case

Appendix A.1: `setFieldsDict`

```
/*--------------------------------*- C++ -*----------------------------
-------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  1.7.1
|
|   \\  /    A nd           | Web:      www.OpenFOAM.com
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------
-------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      setFieldsDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * //

defaultFieldValues ( volVectorFieldValue U ( 0 0 0 )
volScalarFieldValue T 348.432 volScalarFieldValue p 100000 );

regions         ( boxToCell { box ( 0 -1 -1 ) ( 5 1 1 ) ; fieldValues
( volScalarFieldValue T 278.746 volScalarFieldValue p 10000 ) ; } );


//
*********************************************************************
*** //
```

Appendix A.2: `blockMeshDict`

```
/*--------------------------------*- C++ -*----------------------------
-------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  1.7.1
|
|   \\  /    A nd           | Web:      www.OpenFOAM.com
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------
-------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * //

convertToMeters 1;

vertices
(
    (-5 -1 -1)
    (5 -1 -1)
    (5 1 -1)
    (-5 1 -1)
    (-5 -1 1)
    (5 -1 1)
    (5 1 1)
    (-5 1 1)
);

blocks
(
    hex (0 1 2 3 4 5 6 7) (500 1 1) simpleGrading (1 1 1)
);

edges
(
);

patches
```

```
(
    patch sides
    (
        (1 2 6 5)
        (0 4 7 3)
    )
    empty empty
    (
        (0 1 5 4)
        (5 6 7 4)
        (3 7 6 2)
        (0 3 2 1)
    )
);

mergePatchPairs
(
);

//
*********************************************************************
*** //
```

Appendix A.3: p field

```
/*--------------------------------*- C++ -*----------------------------
-------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  1.7.x
|
|   \\  /    A nd           | Web:      www.OpenFOAM.com
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------
-------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * //
```

```
dimensions      [1 -1 -2 0 0 0 0];

internalField   nonuniform List<scalar>
500
(
100000
100000
.
.
.
.
.
.
.
.
.
.
.
100000
10000
10000
.
.
.
.
.
.
.
.
.
.
10000
10000
)
;

boundaryField
{
    sides
    {
        type            zeroGradient;
    }
    empty
    {
        type            empty;
    }
}


//
*********************************************************************
*** //
```

Appendix A.4: T field

```
/*--------------------------------*- C++ -*----------------------------
-------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  1.7.x
|
|   \\  /    A nd           | Web:      www.OpenFOAM.com
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------
-------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      T;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * //

dimensions      [0 0 0 1 0 0 0];

internalField   nonuniform List<scalar>
500
(
348.432
348.432
.
.
.
.
.
348.432
348.432
278.746
278.746
.
.
.
.
278.746
)
;
```

```
boundaryField
{
    sides
    {
        type            zeroGradient;
    }
    empty
    {
        type            empty;
    }
}


//
*********************************************************************
*** //
```

Appendix A.5: U field

```
/*--------------------------------*- C++ -*----------------------------
-------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  1.7.x
|
|   \\  /    A nd           | Web:      www.OpenFOAM.com
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------
-------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    location    "0";
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * //

dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (0 0 0);

boundaryField
```

```
{
    sides
    {
        type            zeroGradient;
    }
    empty
    {
        type            empty;
    }
}


//
*********************************************************************
*** //
```

Appendix A.6: thermophysicalProperties

```
/*---------------------------------*- C++ -*--------------------------
-------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  1.7.1
|
|   \\  /    A nd           | Web:      www.OpenFOAM.com
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------
-------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      thermophysicalProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * //

thermoType
ePsiThermo<pureMixture<constTransport<specieThermo<eConstThermo<perfec
tGas>>>>>;

mixture         air 1 28.9 717.5 0 0 0.7;
```

```
//
*********************************************************************
*** //
```

Appendix A.7: `controlDict`

```
/*--------------------------------*- C++ -*----------------------------
-------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  1.7.1
|
|   \\  /    A nd           | Web:      www.OpenFOAM.com
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------
-------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * //

application     rhoCentralFoam;

startFrom       startTime;

startTime       0;

stopAt          endTime;

endTime         0.007;

deltaT          1e-06;

writeControl    adjustableRunTime;

writeInterval   0.001;

cycleWrite      0;

writeFormat     ascii;
```

```
writePrecision  6;

writeCompression uncompressed;

timeFormat      general;

timePrecision   6;

runTimeModifiable yes;

adjustTimeStep  yes;

maxCo           0.2;

maxDeltaT       1;


//
*********************************************************************
*** //
```

Appendix A.8: fvSchemes

```
/*--------------------------------*- C++ -*----------------------------
-------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  1.7.1
|
|   \\  /    A nd           | Web:      www.OpenFOAM.com
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------
-------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * //

fluxScheme      Kurganov;
```

```
ddtSchemes
{
    default         Euler;
}

gradSchemes
{
    default         Gauss linear;
}

divSchemes
{
    default         none;
    div(tauMC)      Gauss linear;
}

laplacianSchemes
{
    default         Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
    reconstruct(rho) vanLeer;
    reconstruct(U)  vanLeerV;
    reconstruct(T)  vanLeer;
}

snGradSchemes
{
    default         corrected;
}


//
************************************************************************
*** //
```

Appendix A.9: `fvSolution`

```
/*--------------------------------*- C++ -*----------------------------
-------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  1.7.1
|
|   \\  /    A nd           | Web:      www.OpenFOAM.com
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------
-------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * //

solvers
{
    rho
    {
        solver          diagonal;
    }

    rhoU
    {
        solver          diagonal;
    }

    rhoE
    {
        solver          diagonal;
    }

    U
    {
        solver          smoothSolver;
        smoother        GaussSeidel;
        nSweeps         2;
        tolerance       1e-09;
```

```
        relTol          0.01;
    }

    h
    {
        solver          smoothSolver;
        smoother        GaussSeidel;
        nSweeps         2;
        tolerance       1e-10 relTol 0;
    }
}


//
*********************************************************************
*** //
```

Appendix B: Ladenburg case

Appendix B.1 `blockMeshDict`

```
/*--------------------------------*- C++ -*----------------------------
-------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  1.7.1
|
|   \\  /    A nd           | Web:      www.OpenFOAM.com
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------
-------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * //

convertToMeters 0.001;
```

```
vertices
(
    ( 0 0 0)
    (30 0 0)
    ( 0  5 -0.008726655121)
    (30  5 -0.008726655121)
    ( 0 10 -0.017453310242)
    (30 10 -0.017453310242)
    ( 0  5  0.008726655121)
    (30  5  0.008726655121)
    ( 0 10  0.017453310242)
    (30 10  0.017453310242)
);

blocks
(
    hex (0 1 3 2 0 1 7 6) (120 20  1) simpleGrading (1 1 1)
    hex (2 3 5 4 6 7 9 8) (120 20  1) simpleGrading (1 1 1)
);

edges
(
);

patches
(
    patch inlet
    (
        (0 2 6 0)
    )

    patch outlet
    (
      (1 3 7 1)
        (3 5 9 7)
    )

    patch freestreamInlet
    (
        (2 4 8 6)
    )

    patch freestream
    (
        (4 8 9 5)
    )

    wedge wedge1
    (
        (0 2 3 1)
        (2 4 5 3)
```

```
    )

    wedge wedge2
    (
        (0 1 7 6)
        (6 7 9 8)
    )
);

mergePatchPairs
(
);

//
*********************************************************************
*** //
```

Appendix B.2 p field

```
/*--------------------------------*- C++ -*----------------------------
-------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  1.7.1
|
|   \\  /    A nd           | Web:      www.OpenFOAM.com
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------
-------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * //

dimensions      [1 -1 -2 0 0 0 0];

internalField   uniform 101325;

boundaryField
{
    inlet
```

```
    {
        type            fixedValue;
        value           uniform 271724;
    }

    outlet
    {
        type            waveTransmissive;
        field           p;
        phi             phi;
        rho             rho;
        psi             psi;
        fieldInf        101325;
        gamma           1.4;
        lInf            0.025;
        value           uniform 101325;
    }

    freestream
    {
        type            totalPressure;
        value           uniform 101325;
        p0              uniform 101325;
           U               U;
        phi             phi;
        rho             none;
        psi             psi;
        gamma           1.4;
    }

    freestreamInlet
    {
        type            zeroGradient;
    }

    wedge1 {type wedge;}
    wedge2 {type wedge;}
}

//
*********************************************************************
*** //
```

Appendix B.3 T field

```
/*--------------------------------*- C++ -*----------------------------
-------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  1.7.1
|
|   \\  /    A nd           | Web:      www.OpenFOAM.com
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------
-------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      T;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * //

dimensions      [0 0 0 1 0 0 0];

internalField   uniform 298.0;

boundaryField
{
    inlet
    {
        type            fixedValue;
        value           uniform 247.1;
    }

    outlet
    {
        type            zeroGradient;
    }

    freestream
    {
        type            totalTemperature;
        value           uniform 297;
        T0              uniform 297;
           U               U;
        phi             phi;
```

```
        rho             none;
        psi             psi;
        gamma           1.4;
    }

    freestreamInlet
    {
        type            fixedValue;
        value           uniform 297.0;
    }

    wedge1 {type wedge;}
    wedge2 {type wedge;}
}

//
*********************************************************************
*** //
```

Appendix B.4 U field

```
/*--------------------------------*- C++ -*----------------------------
-------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  1.7.1
|
|   \\  /    A nd           | Web:      www.OpenFOAM.com
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------
-------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * //

dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (0 0 0);

boundaryField
```

```
{
    inlet
    {
        type            fixedValue;
        value           uniform (315.6 0 0);
    }

    outlet
    {
        type            inletOutlet;
        inletValue      uniform (0 0 0);
        value           uniform (0 0 0);
    }

    freestream
    {
        type            zeroGradient;
    }

    freestreamInlet
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }

    wedge1 {type wedge;}
    wedge2 {type wedge;}
}

//
*********************************************************************
*** //
```

Appendix B.5 thermophysicalProperties

```
/*---------------------------------*- C++ -*----------------------------
-------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  1.7.1
|
|   \\  /    A nd           | Web:      www.OpenFOAM.com
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------
-------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      thermophysicalProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * //

thermoType
ePsiThermo<pureMixture<sutherlandTransport<specieThermo<hConstThermo<p
erfectGas>>>>>;

mixture         air 1 28.96 1004.5 0 1.458e-06 110.4;


//
************************************************************************
*** //
```

Appendix B.6 `controlDict`

```
/*--------------------------------*- C++ -*----------------------------
-------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  1.7.1
|
|   \\  /    A nd           | Web:      www.OpenFOAM.com
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------
-------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * //

application     rhoCentralFoam;

startFrom       latestTime;

startTime       0;

stopAt          endTime;

endTime         2e-03;

deltaT          1e-10;

writeControl    adjustableRunTime;

writeInterval   2e-05;

cycleWrite      0;

writeFormat     ascii;

writePrecision  15;

writeCompression uncompressed;
```

```
timeFormat      general;

timePrecision   6;

adjustTimeStep  yes;

maxCo           0.5;

maxDeltaT       1;


//
*********************************************************************
*** //
```

Appendix B.7 fvSchemes

```
/*--------------------------------*- C++ -*----------------------------
-------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  1.7.1
|
|   \\  /    A nd           | Web:      www.OpenFOAM.com
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------
-------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * //

fluxScheme      Kurganov;

ddtSchemes
{
    default         Euler;
}
```

```
gradSchemes
{
    default         Gauss linear;
}

divSchemes
{
    default         none;
    div(tauMC)      Gauss linear;
}

laplacianSchemes
{
    default         Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
    reconstruct(rho) vanLeer;
    reconstruct(U)  vanLeerV;
    reconstruct(T)  vanLeer;
}

snGradSchemes
{
    default         corrected;
}


//
********************************************************************
*** //
```

Appendix B.8 `fvSolution`

```
/*---------------------------------*- C++ -*----------------------------
-------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  1.7.1
|
|   \\  /    A nd           | Web:      www.OpenFOAM.com
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------
-------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * //

solvers
{
    rho
    {
        solver          diagonal;
    }

    rhoU
    {
        solver          diagonal;
    }

    rhoE
    {
        solver          diagonal;
    }

    U
    {
        solver          smoothSolver;
        smoother        GaussSeidel;
        nSweeps         2;
        tolerance       1e-10;
```

```
        relTol          0;
    }

    e
    {
        solver          smoothSolver;
        smoother        GaussSeidel;
        nSweeps         2;
        tolerance       1e-10 relTol 0;
    }
}


//
*********************************************************************
*** //
```

Appendix C: Jet case

Appendix C.1 `blockMeshDict`

```
/*--------------------------------*- C++ -*----------------------------
-------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  2.3.0
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------
-------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * //

convertToMeters 0.001;
```

```
vertices
(
    (  0 0 0)
    (600 0 0)
    (  0 12.5 -0.546)
    (600 12.5 -0.546)
    (  0 300 -13.098)
    (600 300 -13.098)
    (  0 12.5  0.546)
    (600 12.5  0.546)
    (  0 300  13.098)
    (600 300  13.098)
);

blocks
(
    /*hex (0 1 3 2 0 1 7 6) (220 30  1) simpleGrading (2 1 1)*/
    /*hex (2 3 5 4 6 7 9 8) (220 90  1) simpleGrading (2 6 1)*/
    hex (0 1 3 2 0 1 7 6) (400 15  1) simpleGrading (4 1 1)
    hex (2 3 5 4 6 7 9 8) (400 100  1) simpleGrading (4 7 1)
);

edges
(
);

boundary
(
    inlet
    {
        type patch;
        faces
        (
            (0 2 6 0)
        );
    }

    outlet
    {
        type patch;
        faces
        (
            (1 3 7 1)
            (3 5 9 7)
        );
    }

    freestreamInlet
    {
        type wall;
        faces
```

```
        (
            (2 4 8 6)
        );
    }

    freestream
    {
        type patch;
        faces
        (
            (4 8 9 5)
        );
    }

    wedge1
    {
        type wedge;
        faces
        (
            (0 2 3 1)
            (2 4 5 3)
        );
    }

    wedge2
    {
        type wedge;
        faces
        (
            (0 1 7 6)
            (6 7 9 8)
        );
    }
);

mergePatchPairs
(
);

//
*********************************************************************
*** //
```

Appendix C.2 p field

```
/*--------------------------------*- C++ -*----------------------------
-------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  2.3.0
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------
-------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * //

dimensions      [1 -1 -2 0 0 0 0];

internalField   uniform 101325;

boundaryField
{
    inlet
    {
        type            fixedValue;
        value           uniform 1.0566e6;
    }

    outlet
    {
        type            waveTransmissive;
        field           p;
        phi             phi;
        rho             rho;
        psi             thermo:psi;
        fieldInf        101325;
        gamma           1.4;
        value           uniform 101325;
    }
```

```
    freestream
    {
        type            waveTransmissive;
        field           p;
        phi             phi;
        rho             rho;
        psi             thermo:psi;
        fieldInf        101325;
        gamma           1.4;
        value           uniform 101325;

        /*type            totalPressure;*/
        /*value           uniform 101325;*/
        /*p0              uniform 101325;*/
        /*U               U;*/
        /*phi             phi;*/
        /*rho             none;*/
        /*psi             thermo:psi;*/
        /*gamma           1.4;*/
    }

    freestreamInlet
    {
        type            zeroGradient;
    }

    wedge1 {type wedge;}
    wedge2 {type wedge;}
}

//
*********************************************************************
*** //
```

Appendix C.3 T field

```
/*--------------------------------*- C++ -*----------------------------
-------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  2.3.0
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------
-------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      T;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * //

dimensions      [0 0 0 1 0 0 0];

internalField   uniform 283.0;

boundaryField
{
    inlet
    {
        type            fixedValue;
        value           uniform 233.33;
    }

    outlet
    {
        type            zeroGradient;
    }

    freestream
    {
        type            inletOutlet;
        inletValue      uniform 283;
        value           uniform 283;
        /*type            totalTemperature;*/
        /*value           uniform 283;*/
```

```
        /*T0              uniform 283;*/
        /*U               U;*/
        /*phi             phi;*/
        /*rho             none;*/
        /*psi             thermo:psi;*/
        /*gamma           1.4;*/
    }

    freestreamInlet
    {
        type            fixedValue;
        value           uniform 283;
    }

    wedge1 {type wedge;}
    wedge2 {type wedge;}
}

//
*********************************************************************
*** //
```

Appendix C.4 U field

```
/*--------------------------------*- C++ -*----------------------------
-------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  2.3.0
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------
-------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * //

dimensions      [0 1 -1 0 0 0 0];
```

```
internalField   uniform (0 0 0);

boundaryField
{
    inlet
    {
        /*type            pressureDirectedInletOutletVelocity;*/
        /*inletDirection   uniform    (1 0 0);*/
        /*value uniform (300.0 0 0);*/
        type fixedValue;
        value uniform (306.03 0 0);
    }

    outlet
    {
        type            inletOutlet;
        inletValue      uniform (0 0 0);
        value           uniform (0 0 0);
    }

    freestream
    {
        type            inletOutlet;
        inletValue      uniform (0 0 0);
        value           uniform (0 0 0);
    }

    freestreamInlet
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }

    wedge1 {type wedge;}
    wedge2 {type wedge;}
}

//
*********************************************************************
*** //
```

Appendix C.5 thermophysicalProperties

```
/*--------------------------------*- C++ -*----------------------------
-------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  2.3.0
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------
-------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      thermophysicalProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * //

thermoType
{
    type            hePsiThermo;
    mixture         pureMixture;
    transport       sutherland;
    thermo          hConst;
    equationOfState perfectGas;
    specie          specie;
    energy          sensibleInternalEnergy;
}

mixture
{
    specie
    {
        nMoles          1;
        molWeight       28.96;
    }
    thermodynamics
    {
        Cp              1004.5;
        Hf              0;
    }
```

```
    transport
    {
        As              1.458e-06;
        Ts              110.4;
        Pr              1;
    }
}


//
*********************************************************************
*** //
```

Appendix C.6 controlDict

```
/*--------------------------------*- C++ -*----------------------------
-------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  2.3.0
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------
-------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * //

application     rhoCentralFoam;

startFrom       latestTime;

startTime       0;

stopAt          endTime;

endTime         3e-03;
```

```
deltaT          1e-10;

writeControl    adjustableRunTime;

writeInterval   2e-05;

cycleWrite      0;

writeFormat     ascii;

writePrecision  15;

writeCompression off;

timeFormat      general;

timePrecision   6;

adjustTimeStep  yes;

maxCo           0.5;

maxDeltaT       1;


functions
{
    fieldAverage1
    {
        type            fieldAverage;
        functionObjectLibs ( "libfieldFunctionObjects.so" );
        outputControl   outputTime;
        timeStart       0.002;
        fields
        (
            p
            {
                mean        on;
                prime2Mean  off;
                base        time;
            }

            rho
            {
                mean        on;
                prime2Mean  off;
                base        time;
            }

            U
            {
```

```
                mean        on;
                prime2Mean  off;
                base        time;
            }

            T
            {
                mean        on;
                prime2Mean  off;
                base        time;
            }
        );
    }
}
//
*********************************************************************
*** //
```

Appendix C.7 fvSchemes

```
/*--------------------------------*- C++ -*----------------------------
-------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  2.3.0
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------
-------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * //

fluxScheme      Kurganov;

ddtSchemes
{
    default         Euler;
```

```
}

gradSchemes
{
    default         Gauss linear;
}

divSchemes
{
    default         none;
    div(tauMC)      Gauss linear;
    div(phi,epsilon) Gauss limitedLinear 1;
    div(phi,k) Gauss limitedLinear 1;
}

laplacianSchemes
{
    default         Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
    reconstruct(rho) vanLeer;
    reconstruct(U)  vanLeerV;
    reconstruct(T)  vanLeer;
}

snGradSchemes
{
    default         corrected;
}


//
************************************************************************
*** //
```

Appendix C.8 `fvSolutions`

```
/*--------------------------------*- C++ -*----------------------------
-------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  2.3.0
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------
-------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * //

solvers
{
    "(rho|rhoU|rhoE)"
    {
        solver          diagonal;
    }

    "(U|k|epsilon)"
    {
        solver          smoothSolver;
        smoother        GaussSeidel;
        nSweeps         2;
        tolerance       1e-10;
        relTol          0;
    }

    e
    {
        $U;
        tolerance       1e-10;
        relTol          0;
    }
}
```

```
//
*********************************************************************
*** //
```

Appendix D: Contents of enclosed CD

The enclosed CD consist of the master thesis file and all the OpenFOAM case simulation main folders i.e. 0, constant and system folders. The contents can be listed as follows:

- Thesis file in .pdf format
- Shock tube case simulation files
- Extracted shock tube case data in .xlsx format
- Ladenburg case simulation files
- Jet simulation case files