Numerical Simulation Model on Irreversibility of Shock-Wave Process

Longhao Huang
Marquette University

Recommended Citation

Huang, Longhao, "Numerical Simulation Model on Irreversibility of Shock-Wave Process" (2013). *Master's Theses (2009 -)*. 228.
https://epublications.marquette.edu/theses_open/228

NUMERICAL SIMULATION MODEL ON IRREVERSIBILITY OF SHOCK-WAVE PROCESS

by

Longhao Huang, B.S.

A Thesis submitted to the Faculty of the Graduate School,
Marquette University,
in Partial Fulfillment of the Requirements for
the Degree of Master of Science

Milwaukee, Wisconsin

December 2013

ABSTRACT
NUMERICAL SIMULATION MODEL ON IRREVERSIBILITY OF SHOCK-WAVE PROCESS

Longhao Huang, B.S.

Marquette University, 2013

The objective of this research is to develop a better understanding of the irreversibilities associated with the shock compaction of matter, especially as a result of impact. Due to complex shock processes, experimentation alone cannot fix the material state, since properties such as internal energy, entropy as well as the shock process are not measurable. Thus, in addition to experimentation, analytical and numerical methods are also used to completely characterize the shock process, although they are restricted by underlying constitutive assumptions. Instead of using artificial irreversibility, such as artificial viscosity to simplify and stabilize the numeric shock model, this work will directly incorporate and solve the correct constitutive relations that describe the sources of irreversibility.

Shock wave processes in gas and water are simulated and two equations of state (EOS) are discussed. For a one-dimensional shock wave in gas, results from simulations at two different non-dimensional scales utilizing two different EOS are comparable to the idealized analytical solution and experimental data. Besides, the Mie-Grüneisen (M-G) equation of state, which has been used for solids, is extended to study gas and liquid. The value of Mie-Grüneisen constant, which is a function of atom oscillator frequency and specific volume, is hard to detect from experiment. Based on statistical mechanics, a relationship between the gas Mie-Grüneisen constant and specific heat ratio is derived analytically, which makes Mie-Grüneisen EOS available for gas. The M-G constant is also derived from shock jump condition and Mie-Grüneisen EOS for water and a sensitivity analysis is done based on the simulation result.

ACKNOWLEDGMENTS

Longhao Huang, B.S.

First and foremost I would like to thank my advisor Dr. John P. Borg, a real gentleman in academic research, for give me the opportunity to advance my engineering knowledge. I benefit a lot from his patient assistance and guidance in this research. I also would like to thank Dr. Jon D. Koch and Dr. Margaret Mathison for their helpful advice and technical insight in my entire master research and study. I would like to thank my wife and parents for their full support. Finally, I would like to appreciate the Graduate School and the Department of Mechanical Engineering at Marquette University for providing me such a good platform of study.

TABLE OF CONTENTS

Chapter 1

Introduction

1.1 Objectives

Research related to shock wave processes has a long history. Compared with elastic deformation process, shock wave process can make irreversible change to the material state. According to material states, constitutive models can be divided into gas, liquid and solid groups, which vary from simple to complex respectively. Due to complex shock processes, there are various kinds of ways to research shock processes. One approach is to design an experiment to directly collect material property data and capture the shock wave profile, such as velocity, density or pressure. Based on experimental data, the shock Hugoniot compression curve can be established, which is used to estimate the shock compressed state. While shock temperature is usually obtained by a theoretical calculations based on the various thermodynamic quantities or shock Hugoniot data available. So experimental methods alone are insufficient to establish the complete equation of state and the preciseness need to be improved. Numeric methods can assist in fixing the complete equation of state. Results from numeric simulations can be compared to specific points on a thermodynamic surface to assess the effectiveness of simulation procedure.

The objective of this research is to develop a better understanding of the irreversibility associated with the shock compaction of matter, especially as a result of impact, based on numerical simulation. In classical numeric formulations the cells size of the simulation is large compared to the shock thickness, thus the shock is not numerically resolved. As a result, the evolution of irreversibilities within the shock process is not resolved. In order to simplify or stabilize the shock model, artificial irreversibility, such as artificial viscosity, are introduced and applied [1-3]. For one-dimensional shock wave processes in monatomic gas, Navier-Stoke (NS) equations and ideal gas equation of state (EOS) are able to describe the generation of irreversibilities [4]. The use of Navier-Stoke equations makes for a more complete analysis including thermodynamically consistent of temperature [5]. In order to simulate shock interactions that are under resolved, artificial viscosity is included in most simulations in order to dampen-out spurious numeric instabilities. In this work the complete classic Newtonian viscous stress tensor is directly incorporated in the simulation. In order to resolve the gradients required by the Newtonian viscous stress tensor, the very sharp, very thin (<1 μm) shock fronts must be resolved.

Furthermore, shock wave processes in liquids are more complex than those in gases. In order to make one-dimensional Navier-Stoke formulation more robust, additional modifications can be considered, such as non-dimensional scales, different equations of state (EOS), parameter studies and the inclusion of the Newtonian compressibility terms and second viscosity. Specific experimental data and analytical simulation results are to be used to check the accuracy of these simulations. In addition, a sensitivity analysis is presented to analyze the effects of irreversibility in shock wave process.

1.2 Methodology

This work compares numerical simulation, analytical simulation and specific experimental data to accomplish stated objectives.

Experiments are usually conducted under Lagrange coordinate, while parameters from Lagrange coordinate are not convenient to use in numeric simulations, so Euler coordinates are used instead. Work starts from assumptions, which are made from experimental design, physics phenomenon and mathematics derivation process. Then, finite difference techniques are used in the discretization based on simplified equations. A second-order central difference is chosen to discretize numerical equations from analytical Navier-Stoke equations, similar to a Crank–Nicolson method. The scheme is programed in FORTRAN.

Two kinds of equations of state are used in these simulations: ideal gas EOS and Mie-Grüneisen EOS. Sub-models of irreversibility come from the full Newtonian fluid stress tensor, where the transport constants for viscosity are assumed constant or follow an experimentally observed trend line, such as power law of viscosity or Maxwell method of viscosity. In order to shorten the simulation time and keep accuracy high, two kinds of non-dimensional scales are used with these equations and variables. This also helps extend the availability of the numerical model from gas to liquid.

Chapter 2

Ideal Gas EOS Model

2.1 Derivation of Dimensional NS Equations

The following is the general governing Navier-Stokes equations:

Continuity

$$\frac{\partial}{\partial t}\rho = -(\nabla \cdot \rho u) \tag{2.1.1}$$

Motion

$$\frac{\partial}{\partial t}\rho u = -[\nabla \cdot \rho u u] - \nabla p - [\nabla \cdot \tau] + \rho g \tag{2.1.2}$$

Energy

$$\frac{\partial}{\partial t}\rho\left(\hat{K} + \hat{U}\right) = -\left(\nabla \cdot \rho\left(\hat{K} + \hat{H}\right)u\right) - (\nabla \cdot q) - (\nabla \cdot [\tau \cdot u]) + \rho(u \cdot g) \tag{2.1.3}$$

where ρ stands for density, u for the velocity, T for the temperature, $\boldsymbol{\tau}$ for the stress and P is the pressure. In addition the kinetic and internal energy and enthalpy are defined as follows:

$\hat{K} = \frac{1}{2}u^2$ for the kinetic energy per unit mass,

$\hat{U} = C_v T$ for the internal energy per unit mass,

$\hat{H} = \hat{U} + \frac{p}{\rho}$ for the enthalpy per unit mass.

The heat flux due to conduction, q, is described by Fourier's Law: $q = -\kappa\nabla T$.

Assuming one-dimensional plane flow, u, T, ρ and p are functions of space in the x direction and time t, noted as u(x,t), T(x,t), ρ(x,t) and p(x,t). The velocities in the y and z drop out of the formulation. The one-dimensional governing equations can be simplified as follows.

Continuity

$$\frac{\partial}{\partial \mathrm{t}}\rho = -\frac{\partial}{\partial \mathrm{x}}\rho u_x \tag{2.1.4}$$

Use u to replace u_x and set the momentum $\mathrm{j} = \rho u_x$, inserting into equation above, results in

$$\frac{\partial \rho}{\partial \mathrm{t}} + \frac{\partial \mathrm{j}}{\partial \mathrm{x}} = 0 \tag{2.1.5}$$

Motion

$$\frac{\partial}{\partial \mathrm{t}}\rho \mathrm{u}_x = -\frac{\partial \rho \mathrm{u}_x u_x}{\partial \mathrm{x}} - \frac{\partial \mathrm{p}}{\partial \mathrm{x}} - \frac{\partial \tau_{xx}}{\partial \mathrm{x}} \tag{2.1.6}$$

Rearrange equation (2.1.6) to get

$$\frac{\partial \mathrm{j}}{\partial \mathrm{t}} + \frac{\partial \mathrm{ju}}{\partial \mathrm{x}} + \frac{\partial \mathrm{p}}{\partial \mathrm{x}} = -\frac{\partial \tau_{xx}}{\partial \mathrm{x}} \tag{2.1.7}$$

Energy

$$\frac{\partial}{\partial \mathrm{t}}\rho\left(\widehat{K} + \widehat{U}\right) = -\frac{\partial \rho(\widehat{K} + \widehat{H})u}{\partial x} - \frac{\partial q_x}{\partial \mathrm{x}} - \frac{\partial \tau_{xx} u}{\partial \mathrm{x}} \tag{2.1.8}$$

Where the total energy per unit volume is defined as:

$$\mathrm{E} = \rho\left(\widehat{K} + \widehat{U}\right) = K + U \tag{2.1.9}$$

In equation (2.1.9), the U stands for internal energy per unit volume and K stands for kinetic energy per unit volume.

Then, combined with their respective definitions the following equations result:

$$\mathrm{U} = \rho C_v T \tag{2.1.10}$$

$$\mathrm{K} = \frac{1}{2}\rho u^2 \tag{2.1.11}$$

$$\mathrm{E} = \frac{1}{2}\rho u^2 + \rho C_v T \qquad (2.1.12)$$

The thermodynamic relationships for the specific heatsC_p, C_v, and γ and R can be written as

$$\mathrm{R} = C_p - C_v \qquad (2.1.13)$$

$$\gamma = \frac{C_p}{C_v} \qquad (2.1.14)$$

where γ is the specific heat ratio and,

R is the gas constant.

Combining the equations (2.1.13) and (2.1.14), the following results are

$$C_v = \frac{R}{\gamma - 1} \qquad (2.1.15)$$

$$C_p = \frac{R}{\gamma-1} + R = \frac{R\gamma}{\gamma-1} \qquad (2.1.16)$$

This can be combined with the ideal gas equation of state

$$\mathrm{p} = \rho \mathrm{RT} \qquad (2.1.17)$$

$$\rho C_v T = \frac{\rho RT}{\gamma-1} = \frac{p}{\gamma-1} \qquad (2.1.18)$$

Total energy can be expressed by ρ,γ and u, as

$$\mathrm{E} = \frac{1}{2}\rho u^2 + \frac{p}{\gamma-1} \qquad (2.1.19)$$

The enthalpy is defined in the usual way and can be combined with the kinetic energy for simplification:

$$\mathrm{H} = \frac{E+p}{\rho} = \frac{\rho(\widehat{K}+\widehat{U})+p}{\rho} = \widehat{K} + \left(\widehat{U} + \frac{p}{\rho}\right) = \widehat{K} + \widehat{H} \qquad (2.1.20)$$

The energy evolution equation can be obtained by substituting E and H into energy equation (2.1.8) to get

$$\frac{\partial}{\partial \mathrm{t}}\mathrm{E} = -\frac{\partial(jH)}{\partial x} - \frac{\partial q_x}{\partial \mathrm{x}} - \frac{\partial(\tau_{xx}u)}{\partial \mathrm{x}} \tag{2.1.21}$$

Finally the energy equation becomes,

Energy

$$\frac{\partial}{\partial \mathrm{t}}\mathrm{E} + \frac{\partial(jH)}{\partial x} + \frac{\partial q_x}{\partial \mathrm{x}} = -\frac{\partial(\tau_{xx}u)}{\partial \mathrm{x}} \tag{2.1.22}$$

where q is the Fourier Law flux defined as:

$$\mathrm{q} = q_x = -\kappa\frac{\partial T}{\partial x} \tag{2.1.23}$$

Heat conductivity

Some of the transport properties can be combined into dimensionless parameters in order to simplify the governing equations. Since the Prandlt number, Pr, is defined as

$$\mathrm{Pr} = \frac{C_p\eta}{\kappa} \tag{2.1.24}$$

Then rearranging for the heat conduction, k, yields,

$$\kappa = \frac{C_p\eta}{Pr} \tag{2.1.25}$$

Insert the expression for the specific heat, C_p, equation (2.1.16) to get

$$\kappa = \frac{\gamma R\eta}{(\gamma-1)Pr} \tag{2.1.26}$$

The Newtonian stress tensor is defined in equation 2.1.27.

$$\tau_{xx} = -\eta\left[2\frac{\partial u_x}{\partial x}\right] + (\frac{2}{3}\eta - \mu)(\nabla\cdot u) \tag{2.1.27}$$

For a monatomic noble gas with low density, where there are no free electrons in the valence shells, one can assume the second viscosity μ is zero [7]. This is born out by experimental results. As a result, one arrives at the simplified one dimensional stress

$$\tau_{xx} = -\frac{4}{3}\eta\frac{\partial u}{\partial x} \tag{2.1.28}$$

One should note that equation 2.1.28 retains the compressibility terms from 2.1.27. But it has been simplified greatly. Finally, one should note that in order to resolve this scheme the gradients in equations 2.1.28 and 23 must be resolved.

2.2 Non-Dimensional NS Equations – Ideal Gas Scheme

Since the upstream densityρ_∞, mean-free path λ and sound speed C_∞ can be obtained from experiments and the gas constant R is a constant, they are well suited to be used as non-dimensional scales. In addition the ratio of specific heats, γ is used to dimensionalize the energy equation. The mean free path for a perfect gas is from G.A.Bird [6]. It can be manipulated to the following form,

$$\lambda = \frac{4\eta}{(\rho\sqrt{2\pi RT}\Omega)} \tag{2.2.1}$$

where, Ω is a coefficient from mean collision rate of molecule, which is defined as,

$$\Omega = \frac{5(\alpha+1)(\alpha+2)}{\alpha(7-2\omega)(5-2\omega)} \tag{2.2.2}$$

where, α is an exponent in the variable molecular model. When $\alpha\ = 1$, equation (2.2.1) is applicable for hard sphere assumption and ω is viscosity index, a coefficient related to fluid.

Using scales above mentioned, the following basic dimensionless parameters can be formulated as $\rho = \tilde{\rho}\rho_\infty$, $u = \tilde{u}C_\infty$, $p = \tilde{p}\rho_\infty C_\infty^2$, $x = \tilde{x}\lambda$, $t = \tilde{t}\frac{\lambda}{C_\infty}$

Substituting these into the one dimensional governing equations (2.1.5, 2.1.7 and 2.1.22), yields:

$$\frac{\partial\tilde{\rho}\rho_{\infty}}{\partial\tilde{t}\frac{\lambda}{C_{\infty}}}+\frac{\partial\tilde{\rho}\tilde{u}\rho_{\infty}C_{\infty}}{\partial\tilde{x}\lambda}=0 \qquad (2.2.3)$$

where the momentum flux is defined as:

$$\mathrm{j} = \rho\mathrm{u} = \tilde{\rho}\tilde{u}\rho_{\infty}C_{\infty}. \qquad (2.2.4)$$

Combining these two and factoring yields

$$\frac{\rho_{\infty}C_{\infty}}{\lambda}\left(\frac{\partial\tilde{\rho}}{\partial\tilde{t}}+\frac{\partial\tilde{\rho}\tilde{u}}{\partial\tilde{x}}\right)=0 \qquad (2.2.5)$$

which simplifies to

$$\frac{\partial\tilde{\rho}}{\partial\tilde{t}}+\frac{\partial\tilde{\rho}\tilde{u}}{\partial\tilde{x}}=0. \qquad (2.2.6)$$

The stress can be non-dimensionalized as

$$\tau_{xx}=-\frac{4}{3}\eta\frac{\partial u}{\partial x}=-\frac{4}{3}\eta\frac{\partial\tilde{u}C_{\infty}}{\partial\tilde{x}\lambda} \qquad (2.2.7)$$

Inserting into the momentum equation yields

$$\frac{\partial\tilde{\rho}\tilde{u}\rho_{\infty}C_{\infty}}{\partial\tilde{t}\frac{\lambda}{C_{\infty}}}+\frac{\partial\tilde{\rho}\tilde{u}\rho_{\infty}C_{\infty}\tilde{u}C_{\infty}}{\partial\tilde{x}\lambda}+\frac{\partial\tilde{p}\rho_{\infty}C_{\infty}^{2}}{\partial\tilde{x}\lambda}=-\frac{\partial\frac{4}{3}\eta\frac{\partial\tilde{u}C_{\infty}}{\partial\tilde{x}\lambda}}{\partial\tilde{x}\lambda} \qquad (2.2.8)$$

By collect terms the following equation results,

$$\frac{\rho_{\infty}C_{\infty}^{2}}{\lambda}\left(\frac{\partial\tilde{\rho}\tilde{u}}{\partial\tilde{t}}+\frac{\partial\tilde{\rho}\tilde{u}^{2}}{\partial\tilde{x}}+\frac{\partial\tilde{p}}{\partial\tilde{x}}\right)=-\frac{4}{3}\eta\frac{C_{\infty}}{\lambda^{2}}\frac{\partial^{2}\tilde{u}}{\partial\tilde{x}^{2}} \qquad (2.2.9)$$

$$\left(\frac{\partial\tilde{\rho}\tilde{u}}{\partial\tilde{t}}+\frac{\partial\tilde{\rho}\tilde{u}^{2}}{\partial\tilde{x}}+\frac{\partial\tilde{p}}{\partial\tilde{x}}\right)=-\frac{4}{3}\frac{\eta}{\lambda\rho_{\infty}C_{\infty}}\frac{\partial^{2}\tilde{u}}{\partial\tilde{x}^{2}} \qquad (2.2.10)$$

The left side of equation (2.2.10) is dimensionless, in order to keep the right side dimensionless, the following requirement is enforced.

$$\frac{\eta}{\lambda\rho_{\infty}C_{\infty}}=dimensionless \qquad (2.2.11)$$

To get non-dimensional scale for η, $\eta = \tilde{\eta}(\lambda\rho_{\infty}C_{\infty})$, the momentum equation can be non-dimensionalized as follows:

Momentum

$$\left(\frac{\partial\tilde{\rho}\tilde{u}}{\partial\tilde{t}}+\frac{\partial\tilde{\rho}\tilde{u}^2}{\partial\tilde{x}}+\frac{\partial\tilde{p}}{\partial\tilde{x}}\right)=-\frac{4}{3}\tilde{\eta}\frac{\partial^2\tilde{u}}{\partial\tilde{x}^2}=-\frac{\partial\tilde{\tau}_{xx}}{\partial\tilde{\mathrm{x}}}=\frac{\partial\tilde{\Pi}_{xx}}{\partial\tilde{\mathrm{x}}} \tag{2.2.12}$$

where the stress tensor is

$$\tilde{\Pi}_{xx}=-\tilde{\tau}_{xx}$$

The dimensionless total energy can be written as

$$\mathrm{E}=\frac{1}{2}\rho u^2+\frac{p}{\gamma-1}=\frac{1}{2}\tilde{\rho}\rho_\infty(\tilde{u}C_\infty)^2+\frac{\tilde{p}\rho_\infty C_\infty^2}{\gamma-1}=\rho_\infty C_\infty^2\left(\frac{1}{2}\tilde{\rho}\tilde{u}^2+\frac{\tilde{p}}{\gamma-1}\right)=\rho_\infty C_\infty^2\,\tilde{E}$$

(2.2.13)

where the dimensionless enthalpy is defined as

$$\mathrm{H}=\frac{E+p}{\rho}=\frac{\rho_\infty C_\infty^2\,\tilde{E}+\tilde{p}\rho_\infty C_\infty^2}{\tilde{\rho}\rho_\infty}=C_\infty^2\,\frac{\tilde{E}+\tilde{p}}{\tilde{\rho}}=C_\infty^2\tilde{H} \tag{2.2.14}$$

From the ideal gas equation of state $\mathrm{p}=\rho\mathrm{RT}$, the dimensionless temperature can be derived,

$$\mathrm{T}=\frac{p}{\rho R}=\frac{\tilde{p}\rho_\infty C_\infty^2}{\tilde{\rho}\rho_\infty R} \tag{2.2.15}$$

The various thermodynamic variables can be made non-dimensional, which results in

$$\mathrm{T}=\frac{\gamma\tilde{p}C_\infty^2}{\gamma\tilde{\rho}R}=\tilde{T}\frac{C_\infty^2}{\gamma R} \tag{2.2.16}$$

Further, the heat flux can be non-dimensionalized as

$$\mathrm{q}=q_x=-\kappa\frac{\partial T}{\partial x}=-\kappa\frac{\partial\tilde{T}\frac{C_\infty^2}{\gamma R}}{\partial\tilde{x}\lambda}=-\kappa\frac{C_\infty^2}{\lambda\gamma R}\frac{\partial\tilde{T}}{\partial\tilde{x}} \tag{2.2.17}$$

The dimensionless variables $\tilde{\mathrm{E}}$, $\tilde{\mathrm{H}}$, $\tilde{\mathrm{j}}$, $\tilde{\mathrm{T}}$, $\tilde{\tau}_{xx}$ and $\tilde{\mathrm{q}}$ with corresponding parameters can be substituted into the energy equation to get

$$\frac{\partial}{\partial\tilde{t}\frac{\lambda}{C_\infty}}\rho_\infty C_\infty^2\,\tilde{E}+\frac{\partial\left(\tilde{\rho}\tilde{u}\rho_\infty C_\infty C_\infty^2\tilde{H}\right)}{\partial\tilde{x}\lambda}+\frac{\partial-\frac{C_\infty^2}{\lambda\gamma R}\tilde{q}}{\partial\tilde{x}\lambda}=\frac{\partial(\frac{4}{3}\tilde{\eta}\lambda\rho_\infty C_\infty\frac{\partial\tilde{u}C_\infty}{\partial\tilde{x}\lambda}\tilde{u}C_\infty)}{\partial\tilde{x}\lambda} \tag{2.2.18}$$

which reduces to

$$\frac{\rho_\infty C_\infty^3}{\lambda}(\frac{\partial\tilde{E}}{\partial\tilde{t}}+\frac{\partial(\tilde{j}\tilde{H})}{\partial\tilde{x}})+(-\kappa\frac{C_\infty^2}{\lambda^2\gamma R})\frac{\partial^2\tilde{T}}{\partial\tilde{x}^2}=\frac{4\tilde{\eta}\rho_\infty C_\infty^3}{3\lambda}\frac{\partial}{\partial\tilde{x}}(\tilde{u}\frac{\partial\tilde{u}}{\partial\tilde{x}}) \tag{2.2.19}$$

Since both sides are multiply by $\frac{\lambda}{\rho_\infty C_\infty^3}$, then the above simplifies to

$$\frac{\partial\tilde{E}}{\partial\tilde{t}}+\frac{\partial(\tilde{j}\tilde{H})}{\partial\tilde{x}}+(\frac{-\kappa}{\rho_\infty C_\infty\lambda\gamma R})\frac{\partial^2\tilde{T}}{\partial\tilde{x}^2}=\frac{4\tilde{\eta}}{3}\frac{\partial}{\partial\tilde{x}}(\tilde{u}\frac{\partial\tilde{u}}{\partial\tilde{x}}) \tag{2.2.20}$$

The right side of this equation is dimensionless, which requires

$$\frac{-\kappa}{\rho_\infty C_\infty\lambda\gamma R}=dimensionless \tag{2.2.21}$$

Assuming the non-dimensional scale for heat conductivity is

$$\kappa=\tilde{\kappa}\rho_\infty C_\infty\lambda\gamma R. \tag{2.2.22}$$

Energy equation reduces to

$$\frac{\partial\tilde{E}}{\partial\tilde{t}}+\frac{\partial(\tilde{j}\tilde{H})}{\partial\tilde{x}}-\tilde{\kappa}\frac{\partial^2\tilde{T}}{\partial\tilde{x}^2}=-\frac{4}{3}\tilde{\eta}[(\frac{\partial\tilde{u}}{\partial\tilde{x}})^2+\tilde{u}\frac{\partial^2\tilde{u}}{\partial\tilde{x}^2}]=-\frac{\partial(\tilde{\tau}_{xx}\tilde{u})}{\partial\tilde{x}}=\frac{\partial(\tilde{\Pi}_{xx}\tilde{u})}{\partial\tilde{x}} \tag{2.2.23}$$

where, $\tilde{\Pi}_{xx}=-\tilde{\tau}_{xx}$.

So finally on arrives at a consistent dimensionless scheme for gas behavior:

$$u\sim C_\infty,\ x\sim\lambda,\ t\sim\frac{\lambda}{C_\infty},\ \rho\sim\rho_\infty,\ P\sim\rho_\infty C_\infty^2,$$

$$T\sim\frac{C_\infty^2}{\gamma R},\ \eta\sim\lambda\rho_\infty C_\infty,\ \kappa\sim\rho_\infty C_\infty\lambda\gamma R.$$

2.3 Non-Dimensional NS Equations – General Fluid Scheme

The non-dimensional scales used in the previous derivation are for gas simulation. This is because the ideal gas equation of state is explicit in these assumptions. In order to extend the applicability of the non-dimensional scheme to fluids and solids, a set of more general non-dimensional scales needs to be introduced. As upstream density ρ_∞, mean-free path λ and sound speed C_∞ can be also taken from experiments and analytical calculation in liquid, they can be used for this purpose. To get more general the gas constant R can be replaced by the specific heat Cv. This eliminates the need for the gas specific heat ratio γ.

Thus the more general non-dimensional scales are as following:

Fundamental scales

$\rho = \tilde{\rho}\rho_\infty$, $u = \tilde{u}C_\infty$, $p = \tilde{p}\rho_\infty C_\infty^2$, $x = \tilde{x}\lambda$, $t = \tilde{t}\frac{\lambda}{C_\infty}$,

where, C_∞ is no longer local sound speed, but bulk sound speed.

The secondary scales can be expressed as follows:

$\eta = \tilde{\eta}(\lambda\rho_\infty C_\infty)$, $T = \tilde{T}\frac{C_\infty^2}{C_v}$, $k = \tilde{k}(C_v\lambda\rho_\infty C_\infty)$, $q = \tilde{q}(\rho_\infty C_\infty^3)$

During the dimensionless process, the main differences between the gas and the more general fluid/solid dimensionless process is in terms related to temperature. The temperature does not directly appear in the continuity equation and momentum equation. However they are loosely coupled through the transport properties. So the dimensionless forms of these two equations are the same as equation (2.2.6) and (2.2.12).

In order to keep the same unit of heat conductivity, a non-dimensional scale of heat conductivity is introduced, $C_v\lambda\rho_\infty C_\infty$, instead of $\rho_\infty C_\infty\lambda\gamma R$.

Thus the energy equation is

$$\frac{\partial\tilde{E}}{\partial\tilde{t}}\frac{\rho_\infty C_\infty^3}{\lambda} + \frac{\partial\left((\tilde{\rho}\tilde{u}\tilde{H})\rho_\infty C_\infty^3\right)}{\partial\tilde{x}\lambda} + \frac{\partial\tilde{q}\rho_\infty C_\infty^3}{\partial\tilde{x}\lambda} = \frac{\partial(\frac{4}{3}\tilde{\eta}\lambda\rho_\infty C_\infty\frac{\partial\tilde{u}C_\infty}{\partial\tilde{x}\lambda}\tilde{u}C_\infty)}{\partial\tilde{x}\lambda} \tag{2.3.1}$$

which simplifies to,

$$\frac{\partial \tilde{E}}{\partial \tilde{t}} + \frac{\partial (\tilde{J}\widetilde{H})}{\partial \tilde{x}} + \frac{\partial \tilde{q}}{\partial \tilde{x}} = \frac{4}{3}\frac{\partial (\tilde{\eta}\frac{\partial \tilde{u}}{\partial \tilde{x}}\tilde{u})}{\partial \tilde{x}} = \frac{4}{3}\frac{\partial (\widetilde{\Pi}_{xx}\tilde{u})}{\partial \tilde{x}} \tag{2.3.2}$$

where, $\widetilde{\Pi}_{xx} = -\tilde{\tau}_{xx}$.

2.4 Gas Viscosity Sub-Models

Viscosity η and heat conductivity κ are functions of temperature. Four models of viscosity have been investigated in the work, constant viscosity model, power law model, Maxwell model [6] and Chapman-Enskog model [7].

Power Law Model

The power law viscous model is derived from simple observation of experimental measurements that illustrate that viscosity varies with respect to temperature. Experimental data is found to fit a power law form of the following:

$$\eta = \eta_{\infty}(\frac{T}{T_{\infty}})^{\omega} \tag{2.4.1}$$

where, η_{∞} is upstream viscosity, T_{∞} is upstream temperature, and ω is viscosity index based on the data in Chapman and Cowling (1970) [6].

Equation (2.4.1) can be converted into dimensionless form

$$\tilde{\eta} = \tilde{\eta}_{\infty}(\frac{\tilde{T}}{\tilde{T}_{\infty}})^{\omega} \tag{2.4.2}$$

Since the dimensionless upstream boundary condition for temperature is $\tilde{T}_{\infty} = 1.0$, the viscosity simples to

$$\tilde{\eta} = \tilde{\eta}_{\infty}(\tilde{T})^{\omega} \tag{2.4.3}$$

Maxwell model

The Maxwell model is an analytic viscosity model derived from ridged body particle behavior and a Boltzman velocity distribution [7]. The result is

$$\eta = \frac{2}{3}\frac{\sqrt{mkT/\pi}}{\pi d^2} \tag{2.4.4}$$

where, m is molecular mass, k is Boltzmann's constant, d is the diameter of gas molecule, such as argon, $kT = \tilde{k}\tilde{T}\rho_\infty\lambda^3 C_\infty^2$, $m = \tilde{m}\rho_\infty\lambda^3$, $d = \tilde{d}\lambda$.

Inserting and simplifying yields

$$\tilde{\eta} = \frac{2}{3\pi\tilde{d}^2}\sqrt{\frac{\tilde{m}\tilde{k}\tilde{T}}{\pi}} \tag{2.4.5}$$

Chapman-Enskog Model

The Chapman-Enskog model is based a semi-theoretical assumption by Chapman and Enskog. It's an extension of the rigid ball/Botzman distribution model of Maxwell that includes weak attractive forces. The resulting equation is

$$\eta = \frac{5}{16}\frac{\sqrt{\pi mkT}}{\pi\sigma^2\Omega_u} \tag{2.4.6}$$

where, m is molecular mass, k is Boltzmann's constant, σ is the diameter of gas molecule, Ω_u is dimensionless collision integrals, which can be derived from the trend line as a function of kT/ε, which is an empirical parameter.

Inserting dimensional properties, $m = \tilde{m}\rho_\infty\lambda^3$, $\sigma = \tilde{\sigma}\lambda$, and $kT = \tilde{k}\tilde{T}\rho_\infty\lambda^3 C_\infty^2$ into equation (2.4.6) one arrives at the following simplified result:

$$\tilde{\eta} = \frac{5}{16\Omega_u\tilde{\sigma}^2}\sqrt{\frac{\tilde{m}\tilde{k}\tilde{T}}{\pi}} \tag{2.4.7}$$

With regard to the non-dimensional process, these viscosity models are not affected by the choice of non-dimensional scales.

2.5 Discretization

A second order, central difference, finite difference method is used to discretize the analytic equations presented above. The resulting numerical equations and schemes are similar to a Crank–Nicolson method, where parameters with even subscript are face grids and node grids are marked by odd subscripts. This effectively stagers the space grid and stabilizes the solution integration.

The following finite difference scheme is used:

Continuity

Analytical Form,

$$\frac{\partial \tilde{\rho}}{\partial \tilde{t}} + \frac{\partial \tilde{\rho} \tilde{u}}{\partial \tilde{x}} = 0 \tag{2.5.1}$$

Finite difference form,

$$(\frac{\partial \tilde{\rho}}{\partial \tilde{t}})_i^n + \frac{\partial \tilde{j}(i)}{\partial \tilde{x}} = 0 \tag{2.5.2}$$

The subscript i is corresponding to space change and superscript n is for time change.

The discrete form can be written as follows:

$$\frac{\tilde{\rho}^{n+1}(i) - \tilde{\rho}^{n}(i)}{\Delta \tilde{t}} + \frac{\tilde{j}(i+1) - \tilde{j}(i-1)}{2\Delta \tilde{x}} = 0 \tag{2.5.3}$$

The hat (or carrot) is used to denote time change,

$$\tilde{\rho}^{n+1}(i) = \hat{\rho}(i) \tag{2.5.4}$$

therefore the forward time density can be expressed as:

$$\hat{p}(i) = \tilde{p}(i) - \frac{\Delta \tilde{t}}{2\Delta \tilde{x}} [\tilde{j}(i+1) - \tilde{j}(i-1)] \tag{2.5.5}$$

Momentum

The generalized non-dimensional differential form of the energy equation can be written as,

$$\left(\frac{\partial\tilde{\rho}\tilde{u}}{\partial\tilde{t}}+\frac{\partial\tilde{\rho}\tilde{u}^2}{\partial\tilde{x}}+\frac{\partial\tilde{p}}{\partial\tilde{x}}\right)=-\frac{4}{3}\tilde{\eta}\frac{\partial^2\tilde{u}}{\partial\tilde{x}^2} \qquad (2.5.6)$$

The finite difference form is,

$$\left(\frac{\partial\tilde{\jmath}}{\partial\tilde{t}}\right)_i^n+\frac{\partial\tilde{\jmath}\tilde{u}}{\partial\tilde{x}}+\frac{\partial\tilde{p}}{\partial\tilde{x}}=-\frac{\partial\tilde{\tau}_{xx}}{\partial\tilde{x}}=\frac{\partial\tilde{\Pi}_{xx}}{\partial\tilde{x}} \qquad (2.5.7)$$

where the discrete form can be written as:

$$\frac{\tilde{\jmath}^{n+1}(i)-\tilde{\jmath}^{n}(i)}{\Delta\tilde{t}}=\frac{{\tilde{\Pi}_{xx}}^{n+1}(i+1)-{\tilde{\Pi}_{xx}}^{n}(i-1)}{2\Delta\tilde{x}}-\frac{\tilde{p}(i+1)-\tilde{p}(i-1)}{2\Delta\tilde{x}}-\frac{\tilde{\jmath}(i+1)-\tilde{\jmath}(i-1)}{2\Delta\tilde{x}} \qquad (2.5.8)$$

Again, the hat notation denotes time change

$$\tilde{\jmath}^{n+1}(i)=\hat{\jmath}(i) \qquad (2.5.9)$$

Thus the momentum can be expressed as:

$$\hat{\jmath}(i)=\tilde{\jmath}(i)+\frac{\Delta\tilde{t}\left[{\tilde{\Pi}_{xx}}^{n+1}(i+1)-{\tilde{\Pi}_{xx}}^{n}(i-1)-\tilde{p}(i+1)+\tilde{p}(i-1)-\tilde{\jmath}(i+1)+\tilde{\jmath}(i-1)\right]}{2\Delta\tilde{x}} \qquad (2.5.10)$$

where the viscous terms can be written as:

$$\Pi_{xx}=\frac{4}{3}\tilde{\eta}(i)\frac{\partial\tilde{u}}{\partial\tilde{x}}=\frac{4}{3}\frac{\tilde{\eta}(i+1)+\tilde{\eta}(i-1)}{2}\frac{\tilde{u}(i+1)-\tilde{u}(i-1)}{2\Delta\tilde{x}} \qquad (2.5.11)$$

Energy

The analytical form of the energy equation is:

$$\frac{\partial\tilde{E}}{\partial\tilde{t}}+\frac{\partial(\tilde{\jmath}\tilde{H})}{\partial\tilde{x}}-\tilde{\kappa}\frac{\partial^2\tilde{T}}{\partial\tilde{x}^2}=-\frac{3}{4}\tilde{\eta}\left[\left(\frac{\partial\tilde{u}}{\partial\tilde{x}}\right)^2+\tilde{u}\frac{\partial^2\tilde{u}}{\partial\tilde{x}^2}\right]=\frac{\partial(\tilde{\Pi}_{xx}\tilde{u})}{\partial\tilde{x}} \qquad (2.5.12)$$

The finite difference form reduces to:

$$\left(\frac{\partial\tilde{E}}{\partial\tilde{t}}\right)_i^n=-\frac{\partial(\tilde{\jmath}\tilde{H})}{\partial\tilde{x}}+\tilde{\kappa}\frac{\partial^2\tilde{T}}{\partial\tilde{x}^2}+\frac{\partial(\tilde{\Pi}_{xx}\tilde{u})}{\partial\tilde{x}} \qquad (2.5.13)$$

where the discrete form is:

$$\frac{\tilde{E}^{n+1}(i) - \tilde{E}^{n}(i)}{\Delta\tilde{t}} = \frac{\tilde{\Pi}_{xx}(i+1)\tilde{u}(i+1) - \tilde{\Pi}_{xx}(i-1)\tilde{u}(i-1)}{2\Delta\tilde{x}}$$

$$-\frac{\tilde{J}(i+1)\tilde{H}(i+1) - \tilde{J}(i-1)\tilde{H}(i-1)}{2\Delta\tilde{x}} - \frac{\tilde{q}(i+1) - \tilde{q}(i-1)}{2\Delta\tilde{x}} \quad (2.5.14)$$

$$\tilde{E}^{n+1}(i) = \hat{E}(i) \quad (2.5.15)$$

Discrete forms for velocity $\hat{u}(i)$ and $\hat{\rho}(i)$ have been derived from the Navier-Stokes (NS) equation. Furthermore, pressure $\hat{p}(i)$ and temperature $\hat{T}(i)$ need to be determined. Constitutive equations can be applied to solve for these two variables. Because $\hat{E}(i)$ is known by definition,

$$\hat{\mathrm{E}}(\mathrm{i}) = \frac{1}{2}\hat{\rho}(i) \cdot \hat{u}^2(i) + \frac{\hat{p}(i)}{\gamma - 1} \quad (2.5.15)$$

then, $\hat{p}(i)$ can be written as:

$$\hat{p}(i) = (\gamma - 1)[\hat{\mathrm{E}}(\mathrm{i}) - \frac{1}{2}\hat{\rho}(i) \cdot \hat{u}^2(i)] \quad (2.5.16)$$

According to the ideal gas equation of state,

$$\hat{p}(i) = \hat{\rho}(i)R\hat{T}(i) \quad (2.5.17)$$

Since temperature can be written as:

$$\mathrm{T} = \tilde{T}\frac{C_\infty^2}{\gamma R} \quad (2.5.18)$$

$\hat{T}$(i) can be resolved from equation 2.5.17 and 18, as:

$$\hat{T}(\mathrm{i}) = \frac{\gamma\hat{p}(i)}{\hat{\rho}(i)} \quad (2.5.19)$$

Heat flux

The heat flux is defined by Fourier Law as:

$$\mathrm{q} = -\kappa\frac{\partial T}{\partial x} \quad (2.5.20)$$

Heat conductivity can be written in a function related to viscosity and gas parameters, γ, R and Prandtl number Pr, as:

$$\kappa = \frac{\gamma R \eta}{(\gamma-1)Pr} \tag{2.5.21}$$

Inserting this into equation 2.5.20, the following expression for the heat flux can be written:

$$\tilde{q}(\mathrm{i}) = -\frac{\gamma R \eta}{(\gamma-1)Pr}\frac{\partial T(i)}{\partial x} = -\frac{1}{(\gamma-1)Pr}\frac{\tilde{\eta}(i+1)+\tilde{\eta}(i-1)}{2}\frac{\tilde{T}(i+1)-\tilde{T}(i-1)}{2\Delta\tilde{x}} \tag{2.5.22}$$

Face grids are assigned the average of adjoining node values [4], as:

Mass flux,

$$j(i) = \frac{j(i+1)+j(i-1)}{2} \tag{2.5.23}$$

Velocity,

$$u(i) = \frac{u(i+1)+u(i-1)}{2} \tag{2.5.24}$$

Pressure,

$$p(i) = \frac{p(i+1)+p(i-1)}{2} \tag{2.5.25}$$

Density,

$$\rho(i) = \frac{\rho(i+1)+\rho(i-1)}{2} \tag{2.5.26}$$

Total Energy,

$$E(i) = \frac{E(i+1)+E(i-1)}{2} \tag{2.5.27}$$

Enthalpy,

$$H(i) = \frac{E(i)+P(i)}{\rho(i)} \tag{2.5.28}$$

i are odd numbers for faces and even numbers for nodes as following:

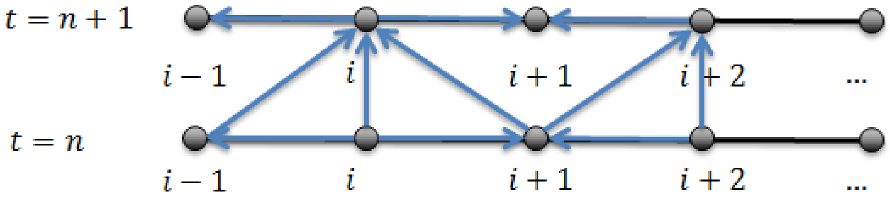

Figure 2.5.1 Numerical Scheme

2.6 Boundary Condition (B.C.)

The left hand side boundary condition can be determined by measurements and analytical calculation or fixed under specific conditions that are reasonable. While the right hand side boundary conditions are more difficult to determine, They depend on a jump relation which include material specific behavior. Two kinds of methods are used to determine right hand side B.C. of shock wave in gas medium. One is the normal shock equations, which is restricted to ideal gas behavior; the other is Rankine-Hugoniot jump condition equations, which is available both for both gas and liquid medium.

2.6.1 Normal Shock Equations

The ideal gas specific shock jump equations are as following:

Pre-shock Mach number is:

$$Ma_1 = \frac{u_1}{\sqrt{\gamma R T_1}} \tag{2.6.1.1}$$

Post-shock Mach number is:

$$Ma_2 = \sqrt{\frac{Ma_1^2 + \frac{2}{\gamma - 1}}{\frac{2 Ma_1^2 \gamma}{\gamma - 1} - 1}} \tag{2.6.1.2}$$

Pressure ratio is:

$$\frac{P_2}{P_1} = \frac{1+\gamma Ma_1^2}{1+\gamma Ma_2^2} \tag{2.6.1.3}$$

Temperature ratio is:

$$\frac{T_2}{T_1} = \frac{1+\frac{(\gamma-1)Ma_1^2}{2}}{1+\frac{(\gamma-1)Ma_2^2}{2}} \tag{2.6.1.4}$$

Density ratio is:

$$\frac{\rho_2}{\rho_1} = \frac{P_2 T_1}{P_1 T_2} \tag{2.6.1.5}$$

Usually, it is more convenient to present the boundary conditions as ratios across shock wave conditions. So, dimensionless parameters are often specified and the right hand side B.C. is determined from the ratios. In this thesis state 1 stands for pre-shock condition, state 2 stands for post-shock condition.

2.6.2 Rankine-Hugoniot Equations

In order to specify boundary conditions for non-ideal gas behavior with the context of this numerical model, the following Rankine-Hugoniot Equations are incorporated, which are more general than normal shock equations.

Continuity

$$\rho_2 U_2 = \rho_1 U_1 \tag{2.6.1.5}$$

Momentum

$$\rho_2 U_2^2 + P_2 = \rho_1 U_1^2 + P_1 \tag{2.6.1.6}$$

Energy

$$\rho_2 U_2 (e_2 + \frac{U_2^2}{2} + \frac{P_2}{\rho_2}) = \rho_1 U_1 (e_1 + \frac{U_1^2}{2} + \frac{P_1}{\rho_1}) \tag{2.6.1.7}$$

2.7 Entropy Calculation Equations

In order to specify the reliability of the numerical simulation, the values of entropy change from theoretical calculation and numerical simulation are compared.

Theoretical entropy change:

$$S_2 - S_1 = C_v \cdot \ln\left(\frac{T_2}{T_1}\right) + R \cdot \ln(\frac{\rho_1}{\rho_2}) \tag{2.7.1}$$

Integrate form of entropy change is used in numerical simulation:

$$S_2 - S_1 = \frac{1}{\dot{m}}(\int \frac{\tau}{T}\frac{dv}{dx}dx - \int \frac{1}{T}\frac{dq}{dx}dx) \tag{2.7.2}$$

Where, $\dot{m}$ is mass flow rate, τ is shear force, q is heat flux.

2.8 Theoretical Solution for Gas

Theoretical solution is set up on one-dimensional compressible flow in a stationary shock wave condition and assumes steady flow with constant viscosity, heat conductivity and specific heat. Second viscosity is neglected for low density gas. [7]

Subscript 1 is corresponding to upstream conditions. The dimensionless velocity distribution equation $\phi(\xi)$ can be written as:

$$\frac{1-\phi}{(\phi-\alpha)^{\alpha}} = \exp[\beta Ma_1(1-\alpha)(\xi-\xi_0)] \quad (\alpha < \phi < 1) \tag{2.8.1}$$

where, ϕ is dimensionless velocity, $\phi = \frac{v_x}{v_1}$; ξ is dimensionless coordinate, $\xi = \frac{x}{\lambda}$; Ma_1 is the Mach number at the upstream condition,

$$Ma_1 = \frac{v_1}{\sqrt{\gamma R T_1/M}}; \tag{2.8.2}$$

α and β are group terms as follows:

$$\alpha = \frac{\gamma-1}{\gamma+1} + \frac{2}{\gamma+1}\frac{1}{{Ma_1}^2}, \tag{2.8.3}$$

$$\beta = \frac{9}{8}(\gamma + 1)\sqrt{\pi/8\gamma}\ . \tag{2.8.4}$$

The reference mean free path λ is defined as:

$$\lambda = \frac{3u_1}{\rho_1}\sqrt{\frac{\pi M}{8RT_1}} \tag{2.8.5}$$

2.9 Mie-Grüneisen EOS

Comparing with section 2.3, this section uses Mie-Grüneisen EOS instead of ideal gas EOS to find the relationship between these two kinds of EOS [8-9].

Mie-Grüneisen equation of state (M-G EOS)

$$P - P_0 = \frac{\gamma}{\nu}(\varepsilon - \varepsilon_0) \tag{2.9.1}$$

where, P is the pressure, ε is the internal energy, P_0 and ε_0 are pressure and internal energy at zero-kelvin states, they can be calculated out by polynomial equations, γ is the Mie-Grüneisen parameter, ν is the specific volume.

Zero-kelvin curve can be evaluated using the Hugoniot as a reference curve,

$$P_H - P_0 = \frac{\gamma}{\nu}(\varepsilon_H - \varepsilon_0) \tag{2.9.2}$$

With equation of state of material,

$$U_s = c + sU_p \tag{2.9.3}$$

Hugoniot pressure can be expressed,

$$P_H = \frac{\rho_0 c^2 x}{(1-sx)^2} \tag{2.9.4}$$

where, $x = 1 - \frac{V}{V_0} = 1 - \frac{\rho_0}{\rho}$.

Corresponding internal energy is

$$\varepsilon_{H} = \frac{P_H \nu_0 x}{2} \tag{2.9.5}$$

A good approximation (2.9.6) from experiments is also used during derivation process.

$$\frac{\gamma}{v} = \frac{\gamma_0}{v_0} = Contant \tag{2.9.6}$$

Plug above equations into equation (2.9.2) to get

$$\frac{\gamma_0}{v_0}\varepsilon_0 + \frac{\partial \varepsilon_0}{\partial v} + \frac{\rho_0 c^2 x}{(1-sx)^2}\left(1 - \frac{\gamma_0}{2}x\right) = 0 \tag{2.9.7}$$

4th order polynomial equation of internal energyε_0 is written as:

$$\varepsilon_0 = \varepsilon_{00} + \varepsilon_{01}x + \varepsilon_{02}x^2 + \varepsilon_{03}x^3 + \varepsilon_{04}x^4 \tag{2.9.8}$$

4th order polynomial equation of P_0 is:

$$P_0 = -\frac{\partial \varepsilon_0}{\partial v} = \rho_0(\varepsilon_{01} + 2\varepsilon_{02}x + 3\varepsilon_{03}x^2 + 4\varepsilon_{04}x^3) \tag{2.9.9}$$

ε_0 and P_0 in equation (2.9.7) can be replaced with equation (2.9.8) and (2.9.9).

Form any strain value, equation (2.9.7) is established generally. So coefficients can be determined.

$$\varepsilon_{01} = \gamma_0\varepsilon_{00}, \tag{2.9.10}$$

$$\varepsilon_{02} = \frac{1}{2}(c^2 + \gamma_0^2\varepsilon_{00}), \tag{2.9.11}$$

$$\varepsilon_{03} = \frac{1}{6}(4sc^2 + \gamma_0^3\varepsilon_{00}), \tag{2.9.12}$$

$$\varepsilon_{04} = \frac{1}{24}(18s^2c^2 - 2\gamma_0 sc^2 + \gamma_0^4\varepsilon_{00}), \tag{2.9.13}$$

Plug these coefficients from equation (2.9.10) to (2.9.13) into equation (2.9.7) and neglect 4th order term; M-G EOS is expanded as following,

$$P = \rho_0 c^2\left[x + \left(2s - \frac{\gamma_0}{2}\right)x^2 + s(3s - \gamma_0)x^3\right] + \frac{\gamma_0\varepsilon}{v_0} \tag{2.9.14}$$

Its corresponding dimensionless form is

$$\tilde{P} = \left[x + \left(2s - \frac{\gamma_0}{2}\right)x^2 + s(3s - \gamma_0)x^3\right] + \gamma_0\tilde{\varepsilon} \qquad (2.9.15)$$

During shock wave transmitting, corresponding to iteration in numerical simulation, density, pressure and internal energy can be solved by NS equations and M-G EOS. Temperature is derived from equation (2.9.16)

$$\varepsilon - \varepsilon_0 = \int_{T_0}^{T} Cv dT \qquad (2.9.16)$$

For constant Cv, specific heat at constant volume, temperature T is

$$T = T_0 + \frac{\varepsilon - \varepsilon_0}{Cv} \qquad (2.9.17)$$

Its dimensionless form is:

$$\tilde{T} = \widetilde{T_0} + \tilde{\varepsilon} - \widetilde{\varepsilon_0} \qquad (2.9.18)$$

Actually, Cv is a function related to temperature, which is discussed in chapter 4 sensitive analysis.

2.10 Mie-Grüneisen Parameter – Gas

2.10.1 General Derivation

M-G EOS

$$P - P_0 = \frac{\gamma}{\nu}(\varepsilon - \varepsilon_0) \qquad (2.10.1.1)$$

where, P_0, ε_0 and $\frac{\gamma}{\nu}$ are constant.

Differential both sides of equation (2.10.1.1) to get

$$dP = \frac{\gamma}{\nu} d\varepsilon \qquad (2.10.1.2)$$

Rearrange equation (2.10.1.2), an equation for γ is derived,

$$\gamma = \nu \frac{dP}{d\varepsilon} \qquad (2.10.1.3)$$

For gas, relation between pressure and internal energy is $\varepsilon = \frac{P}{\rho(\gamma-1)}$, so $\frac{dP}{d\varepsilon} = \rho(\gamma - 1)$,

where, γ is specific heat ratio. In order to distinguish M-G parameter γ with specific heat ratio γ. γ_h is used to replace specific heat ratio.

A relation between M-G parameter γ and specific heat ratio γ_h is derived.

$$\gamma = \gamma_h - 1 \tag{2.10.1.4}$$

2.10.2 Reference State

As known from equation (2.9.8) and (2.9.9), for reference state, x = 0,

$$\varepsilon_0 = \varepsilon_{00} \tag{2.10.2.1}$$

$$P_0 = \rho_0\varepsilon_{01} \tag{2.10.2.2}$$

Solving these two equations above with $\varepsilon_{01} = \gamma_0\varepsilon_{00}$ to get

$$\varepsilon_0 = \frac{P_0}{\rho_0\gamma_0} \tag{2.10.2.3}$$

Compare with ideal gas internal energy equation $\varepsilon_0 = \frac{P_0}{\rho_0(\gamma-1)}$, γ_0 can be determined.

$$\gamma_0 = (\gamma - 1) \tag{2.10.2.4}$$

From above derivation, M-G γ is constant for gas. The physical meaning of M-G γ is the resistance of compression at certain temperature, which means the resistance of compression for low density gas is not sensitive to temperature change.

2.10.3 Eulerian and Lagrangian Referential

In reality, gas can transmit in wind tunnel in several Mach number, while shock-wave transmits though liquid phase materials and experiments are usually conducted in Lagrangian referential. This model set meshes fixed in space, in order to extend the availability of this model. Lagrangian referential is to be transferred to Eulerian referential. The relationship between particle velocity and shock wave velocity are as following:

$$U_1 = U_s - U_{p1} \tag{2.10.3.1}$$

$$U_2 = U_s - U_{p2} \tag{2.10.3.2}$$

where,

U_1 and U_2 are material pre-shock and post-shock velocities in Lagrangian referential,

U_s is shock wave velocity, $U_s = sU_{p2} + c$,

U_{p1} is pre-shock particle velocity. $U_{p1} = 0$, for stationary materials,

U_{p2} is post-shock particle velocity.

This model mainly simulates first shock caused by impact. For stationary materials, there is only one particle velocity $U_{p1} = 0$, $U_{p2} = U_p$. Equation (2.10.3.1) and equation (2.10.3.2) are simplified,

$$U_1 = U_s \tag{2.10.3.3}$$

$$U_2 = U_s - U_p \tag{2.10.3.4}$$

Solving above linear equation system to get

$$U_p = U_1 - U_2 \tag{2.10.3.5}$$

2.10.4 Extended Equation of State of Material (Us-Up curve)

Equation of state of material is usually used for liquid and solid. There is no accurate reference for gas, while coefficients of S and C for gas can be solved by analytical way. From chapter 2.6, boundary conditions of shock wave are determined. There are two more equations, M-G EOS and Us-Up curve, as following

$$P = \rho_0 c^2 \left[x + \left(2s - \frac{\gamma_0}{2}\right) x^2 + s(3s - \gamma_0) x^3\right] + \frac{\gamma_0 \varepsilon}{v_0} \tag{2.10.4.1}$$

$$U_s = sU_p + C \tag{2.10.4.2}$$

where, Pressure P, density ρ_0, internal energy ε, M-G gamma γ_0, stain x, shock velocity U_s and particle velocity U_p are known. There are two equations and two unknown. So Hugoniot slope S and bulk sound speed C can be solved from this equation system.

Table 2.10.4.1 S and C values of Air under different Mach number

Ma	1.1	1.4	1.7	2.0	2.3	2.6	2.9
S	3.148	0.969	0.649	0.517	0.443	0.395	0.360
C	208.013	293.862	381.402	470.106	559.651	649.824	740.481

Us-Up curves under different Mach numbers are shown in Figure 2.10.4.1. Intersections between every two curves are the conditions, which are both workable for Rankine-Hugoniot jump condition equation and equation of state of material. Figure 3.4.1 shows that Us-Up relation of air is not linearly, which means that Hugoniot slope S and bulk sound speed C of air vary according to Mach number.

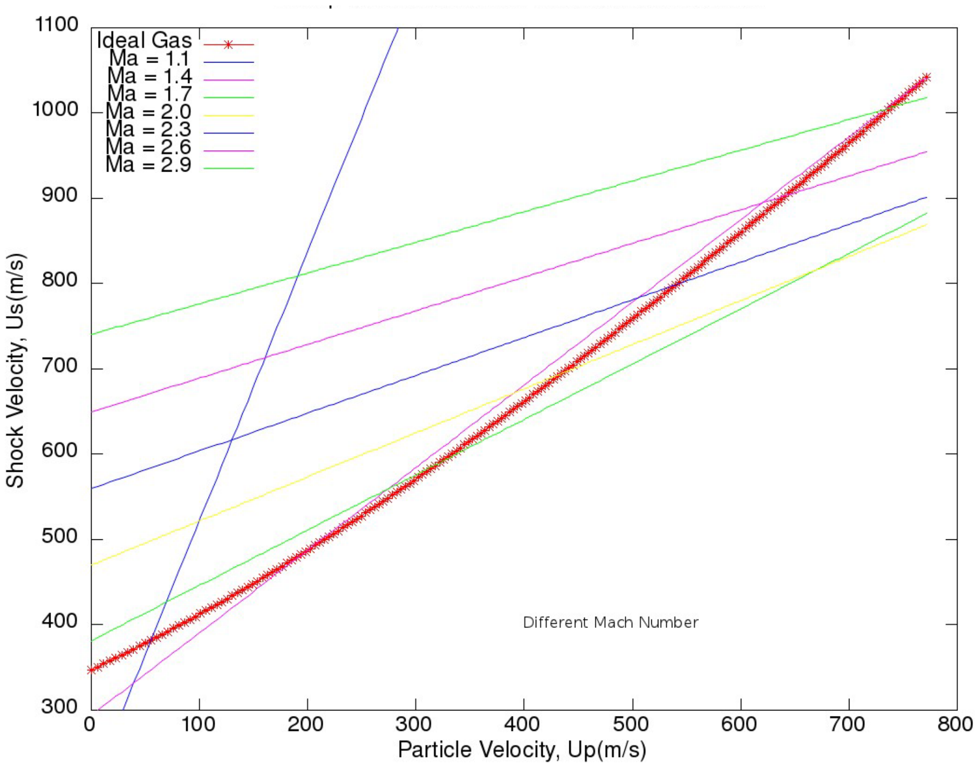

Figure 2.10.4.1: Red curve is the converted Us-Up relation of air based on upstream and downstream Mach numbers. The others are linear Us-Up relation. Intersections are conditions that are both available for Rankine-Hugoniot equations and Us-Up relation.

Chapter 3

Gas Simulation Results

3.1 Gas Dimensionless Scales Based Simulation

3.1.1 B.C. & Parameters Evaluation

In this section the left hand side boundary conditions (B.C.) are defined as follows:

$$u = Ma,\ P = \frac{1}{\gamma},\ \ T = 1.0, \rho = 1.0, e = \frac{P}{\gamma - 1}$$

where, Ma is Mach number, γ is specific heat ratio, e is internal energy per unit mass.

The right hand side B.C. can either be determined based on the downstream Mach number supplemented with normal shock equations or which is a subset of the more generalized Rankine-Hugoniot jump equations.

Parameters evaluations for argon and air are presented in Tables 3.1.1.1 and 3.1.1.2.

Table 3.1.1.1 Parameters Evaluation, Argon

Mach#	γ	ω	Pr	h_x	$Nodes$
1.55	5/3	0.81	2/3	1/8	4800
3.38				1/8	4800
3.38				1/16	9600
9.00				1/16	9600
9.00				1/32	19200

Table 3.1.1.2 Parameters Evaluation, Air

Mach#	γ	ω	Pr	h_x	$Nodes$
1.6	1.4	0.77	0.715	1/16	9600

The dimensionless B.C.s for Argon are presented in Tables 3.1.1.3 through 3.1.1.4.

Table 3.1.1.3 Dimensionless B.C. of Argon, Ma = 1.55

	u	ρ	P	T	e
LHS	1.550	1.000	0.600	1.000	0.900
RHS	0.871	1.779	1.652	1.548	2.478
Ratio	0.562	1.779	2.753	1.548	2.753

Table 3.1.1.4 Dimensionless B.C. of Argon, Ma = 3.38

	u	ρ	P	T	e
LHS	3.380	1.000	0.600	1.000	0.900
RHS	1.067	3.168	8.418	4.429	12.627
Ratio	0.316	3.168	14.031	4.429	14.031

Table 3.1.1.5 Dimensionless B.C. of Argon, Ma = 9.0

	u	ρ	P	T	e
LHS	9.000	1.000	0.600	1.000	0.900
RHS	2.333	3.857	60.600	26.185	90.900
Ratio	0.259	3.857	100.999	26.185	100.999

3.1.2 Validation Check

In order to validate the model, Figures 3.1.2.1 presents the shock wave profile of the numerical simulation compared both the theoretical solution [7] and experimental values [10]. The numeric and analytic solutions are nearly identical; the RMS error is 0.2e-4. This is an indication that the numeric scheme is functioning well. However, neither the analytic nor numeric solutions agree well with the experimental data. As can be seen in the figure, the experimentally measured thickness of the shock is wider and the gradients within the shock are smaller as compared to the analytic and numeric solutions. The reasons for this difference in shock thickness can be attributed to the formulation of the analytic solution. Namely the viscous and diffusive transport properties are assumed constant in the analytic solution. However in reality, temperature variations in the shock front cause significant deviations in the transport properties resulting in a smearing or thickening of the wave. From the experimental result, the thickness of shock front is around 10 times of gas mean free path.

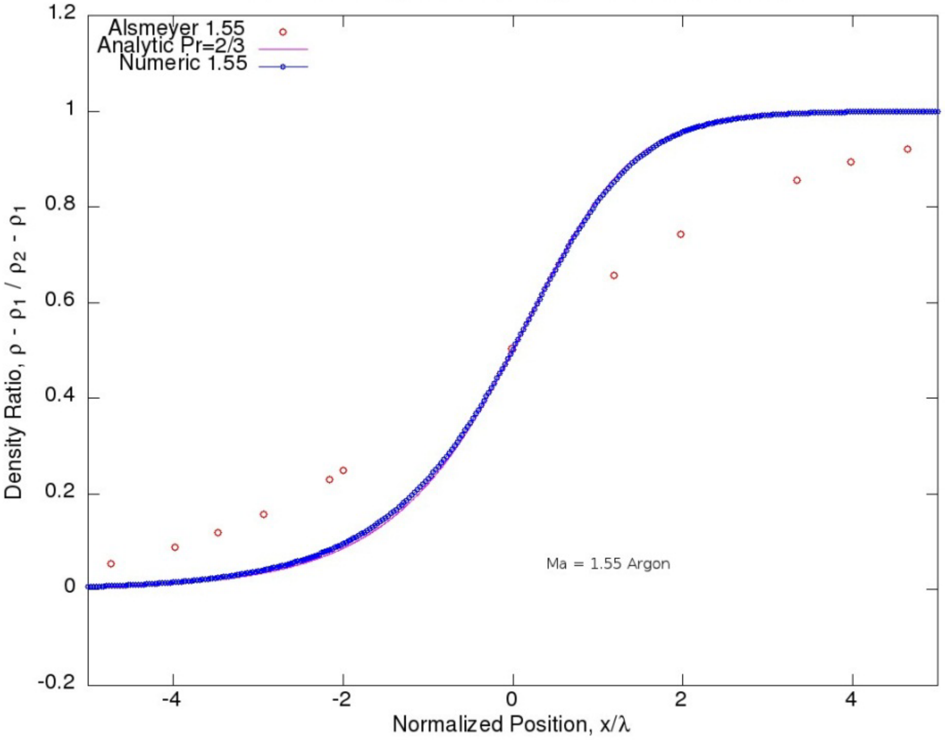

Figure 3.1.2.1: Argon shock profile for Mach number 1.55. Nodes = 4800, hx = 1/8. Analytic and numeric solutions assume constant viscosity.

Figure 3.1.2.2 presents the effect of node numbers on shock wave profile. In order to set boundary in infinity, 300 times mean free path domain is selected experientially firstly. Even though node number doubles, which means the total domain doubles, the gradient and thickness of shock front does not change. Two kinds of marker are identical in values. Thus 300 times mean free path is enough to be recognized as infinity and larger domain does not affect the shock profile.

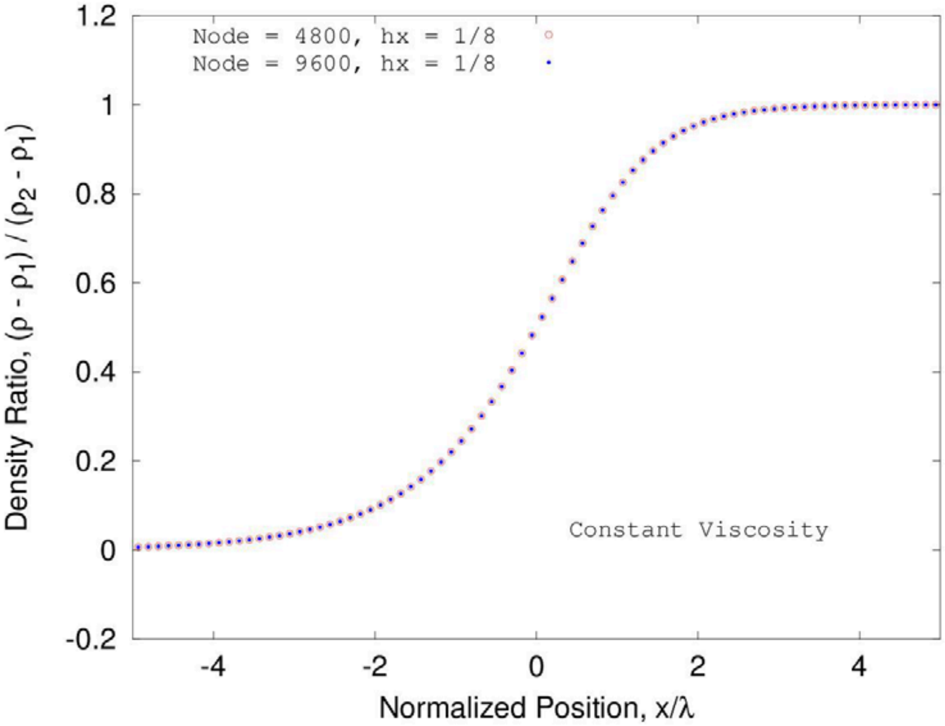

Figure 3.1.2.2 Argon shock wave profile for Mach number 1.55, hx = 1/8. Shock wave profiles of node number 4800 and 9600 are identical.

Figure 3.1.2.3 presents the effect of grid spacing, hx, on shock wave profile. In order to keep the accuracy of the simulation, adequate nodes, i.e. resolution, in shock front are need. The value of the parameter hx, distance between two nodes, corresponds to the number of nodes is related to resolving local gradient. The figure presents the shock profile with two different values of hx, both results in a change in the domain. However, the gradient and thickness of shock front do not change. Thus the domain is converged for values of hx of 1/8th or smaller, which is to say the solution is invariant to grid resolution for hx<1/8.

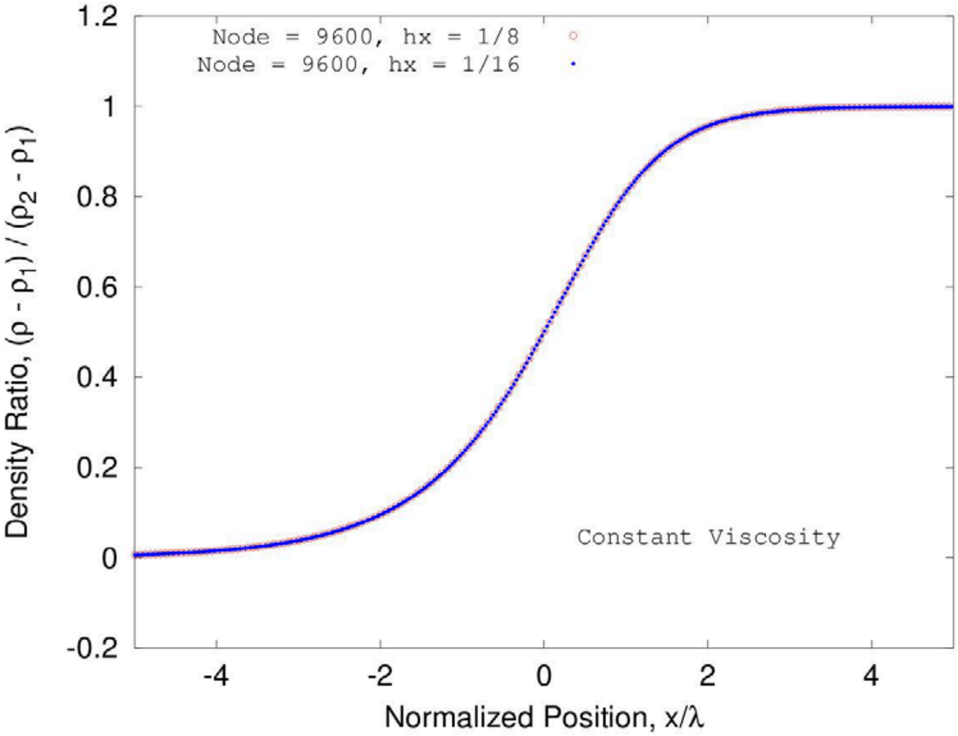

Figure 3.1.2.3 Argon shock wave profile for Mach number 1.55, Node = 9600. Shock wave profiles of hx =1/8 and 1/16 are identical.

Another important aspect of these solutions is the location of the far stream boundary condition relative to the location of the shock thickness. In the above analysis, the far field boundary conditions, which are applied at infinity, are place 30 times the shock. In the same condition except node number and hx value, this numerical model is not sensitive to these two parameters. Ultimately a value of 300 times of gas mean free was used as the total domain size.

Additionally, parameter ht, time step, can affect the stability of the numerical models. According to the Courant stability criteria [11], the requirement for ht is that the distance local disturbances would travel per time step must be no more than the distance between two nodes. Usually, a Courant-Friedrichs-Lewy (CFL) number ranges from 0.1 to 0.5 [12] is used in order to keep gets accurate and stable simulation.

3.1.3 Argon Simulation Analysis – Gas Dimensionless Scale

Figure 3.1.3.1 presents the total entropy change across the shock as a function of iteration number. In addition the theoretical entropy change, equation 2.7.1, across a shock is presented for comparison. The numerically determined entropy converges to the theoretical entropy with iteration, and is nearly identical to the theoretical results from 6.0 in logarithm axis (1 million iterations). Thus one million iterations are enough to get converge the entropy difference. In the following simulations with Mach number of 1.55, iterations are set to 1 million.

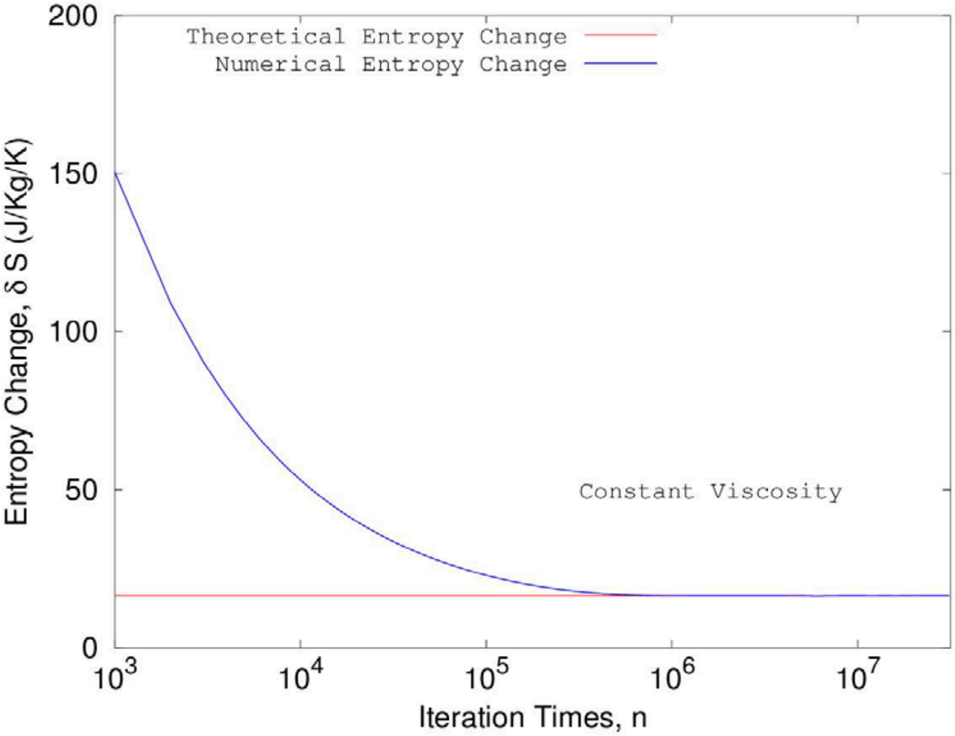

Figure 3.1.3.1: Argon, Mach number 1.55, Nodes = 4800, hx = 1/8. Entropy changes converge and are identical to theoretical profile within iteration increasing.

Figure 3.1.3.2 presents the change in the sum of entropy generation as a function of iteration number for the four viscosity sub-models. Thus as the simulation proceeds and the solution converges, the total increase in entropy asymptotically approaches a steady value very close to the theoretical value.

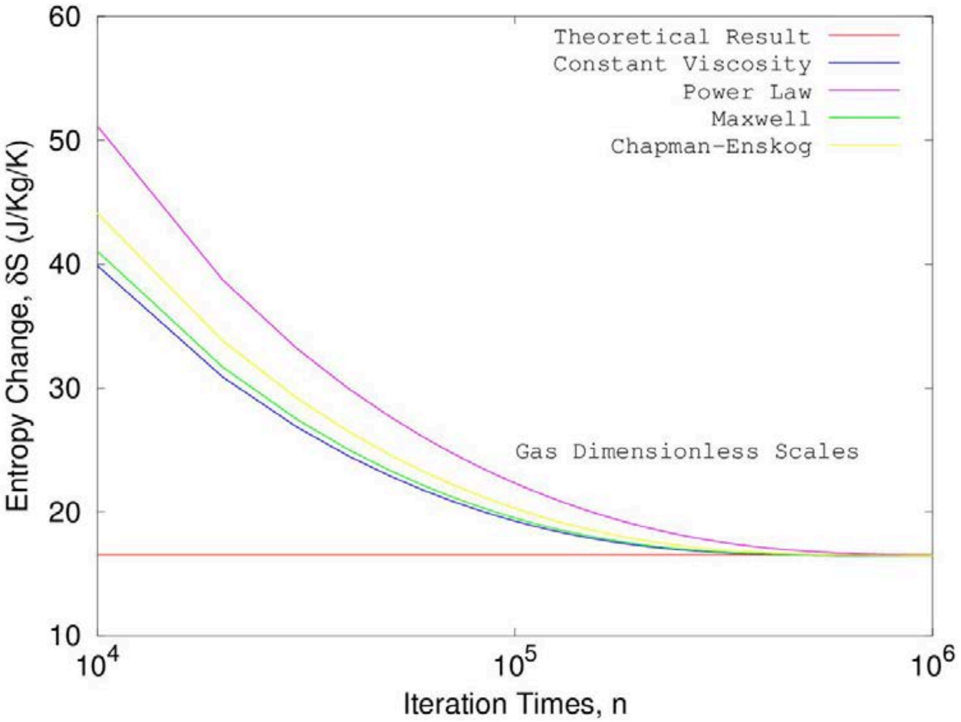

Figure 3.1.3.2: Argon, Mach number 1.55, Nodes = 4800, hx = 1/8. 4 types of viscosity models are built. Entropy changes of these sub-models converge in 1M iteration and approach to theoretical value.

Figure 3.1.3.3 presents the shock profiles using several different viscosity models alongside the analytic solution and experimental data for a Mach number 1.55 in argon. The simulations were evolved over 1 million iterations. Shock wave thickness ranges between 10 to 20 times of mean free path of argon. These four sub-models profiles are set in the area between experimental result and analytical result. The power law model is the closest to experimental profile, which consequently has the largest entropy change, as would be expected.

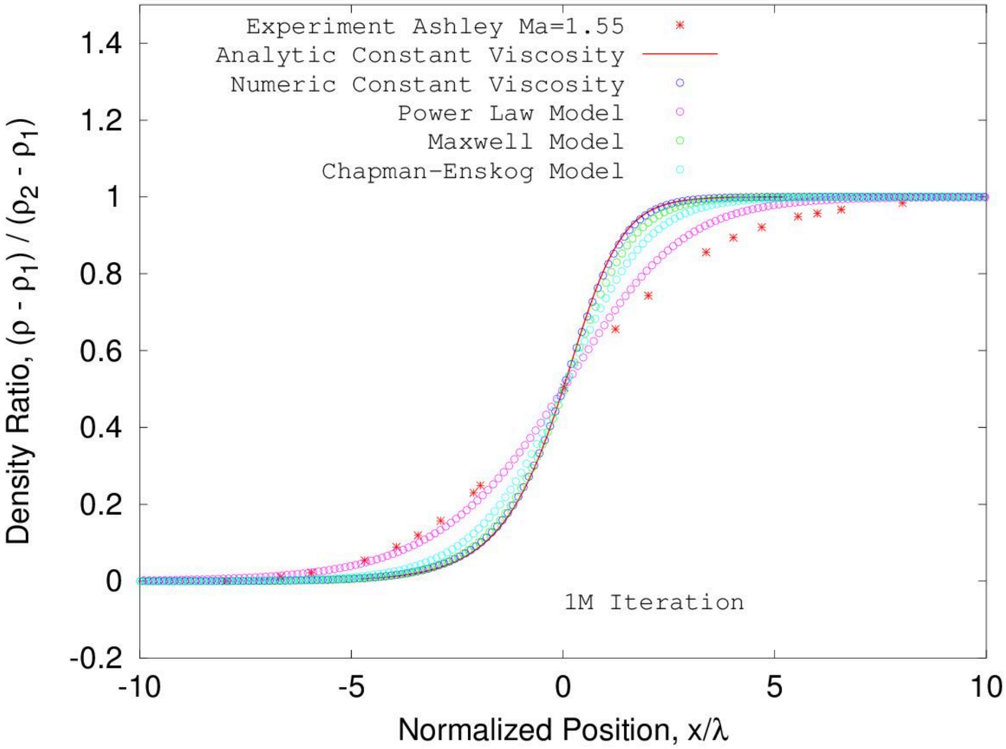

Figure 3.1.3.3: Argon, Mach number 1.55, Nodes = 4800, hx = 1/8, Domain 300. Shock profiles of 4 different types of viscosity models compare with experimental and analytical solutions in domain 20.

With an increase in Mach number, the shock thickness decreases, thus the number of nodes and the node spacing, hx, are adjusted in order to better resolve the shock profile. Each adjustment related to the number of nodes and node spoacing, hx, is companied with a sensitivity analysis. The relationship between hx, nodes number and total domain size is

$$\text{Domain} = \frac{\text{Nodes}\times\text{hx}}{2} \tag{3.1.3.1}$$

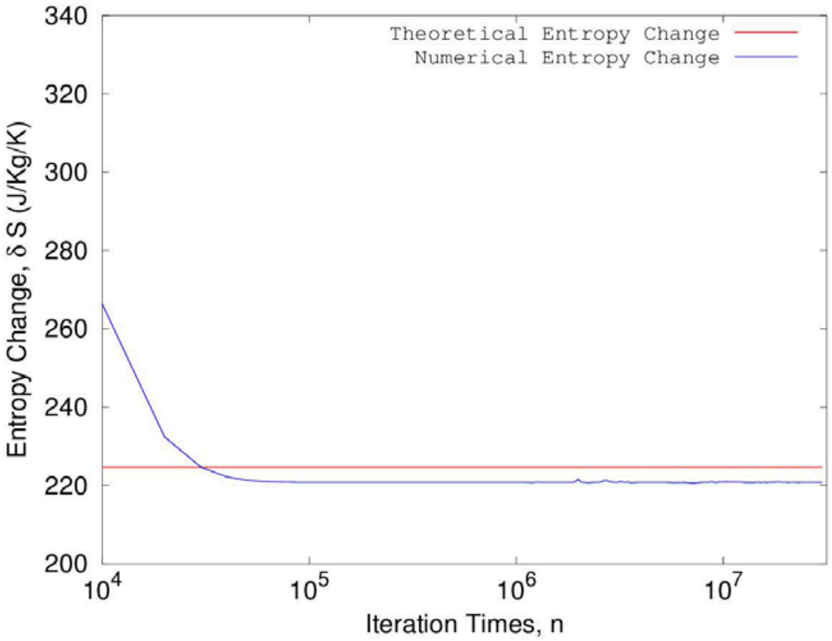

Figure 3.1.3.4: Entropy Change Error within 30M Iteration. Nodes = 4800, hx = 1/8.

The figure shows that it takes less than 0.1M iteration for convergence of entropy change. Theoretical entropy change value is 224.6224 J/Kg/K, Numerical result converge at 220.7 J/Kg/K. There is a 4 J/Kg/K entropy difference compared with theoretical value.

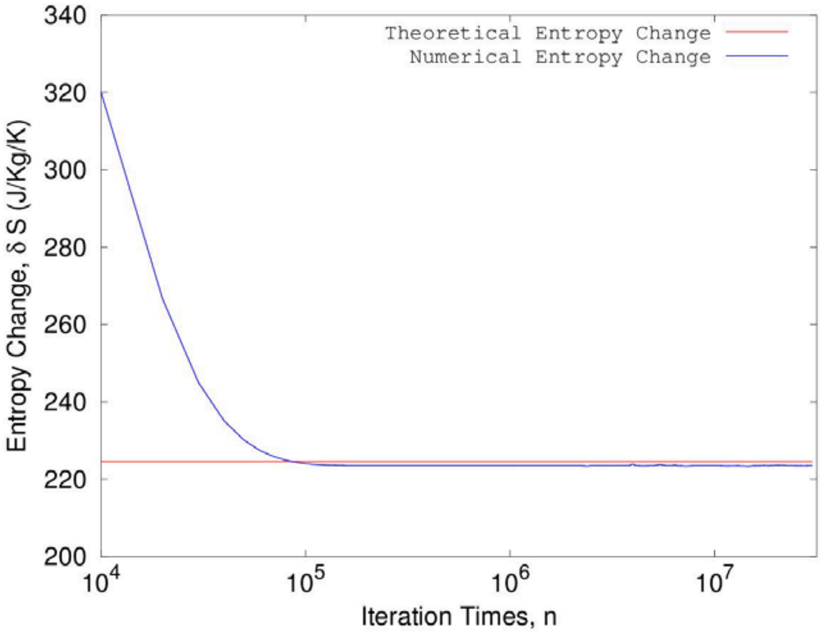

Figure 3.1.3.5: Entropy Change Error within 30M Iteration. Nodes = 9600, hx = 1/16.

The convergence of entropy change takes around 0.1M iteration. But the difference between theoretical and numerical results is much smaller under the same y-axis gradation, compared with Figure 3.1.3.4.

1M Iteration. Nodes = 9600, hx= 1/16. Theoretical entropy change value is 224.6224 J/Kg/K, Numerical result converge at 223.5 J/Kg/K, which is much better than the value from Nodes 4800, hx = 1/8. In sum, more nodes and smaller grids distance make larger variation of entropy change and longer iteration for convergence, while more accurate value of entropy changes.

According the sensitive analysis of nodes scales, the number of nodes is set to 9600 and the grid spacing, hx, is set to 1/16 of the mean free path of argon in the following viscosity sub-model simulations.

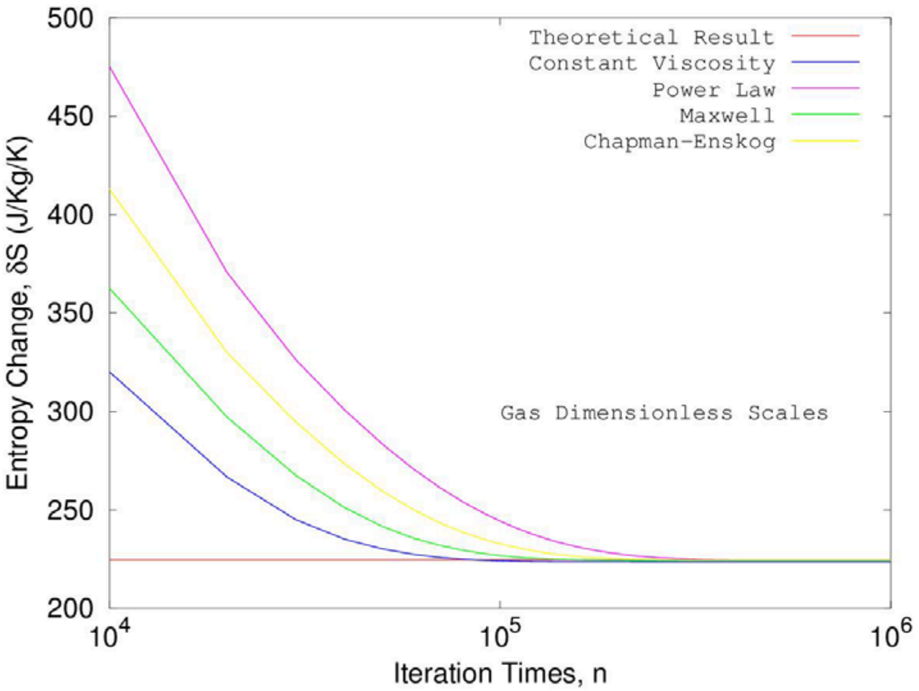

Figure 3.1.3.6: Entropy Change Trends of Different Viscosity Models, Nodes 9600, hx = 1/16. Entropy changes of these sub-models converge in 1M iteration and approach to theoretical value.

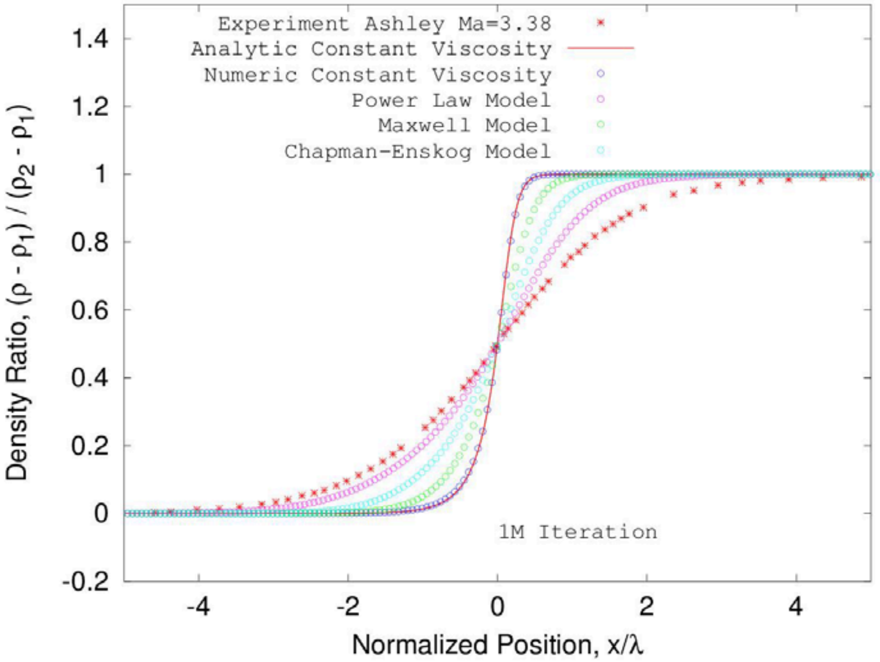

Figure 3.1.3.7: Comparison of Different Viscosity Sub-Models Profiles.

According to the above figure, under Mach number 3.38, shock wave thickness ranges between 10 to 20 times of mean free path of argon. These four sub-models profiles are also set in the area between experimental result and analytical result. Power law model is closest to experimental profile, while with largest entropy change.

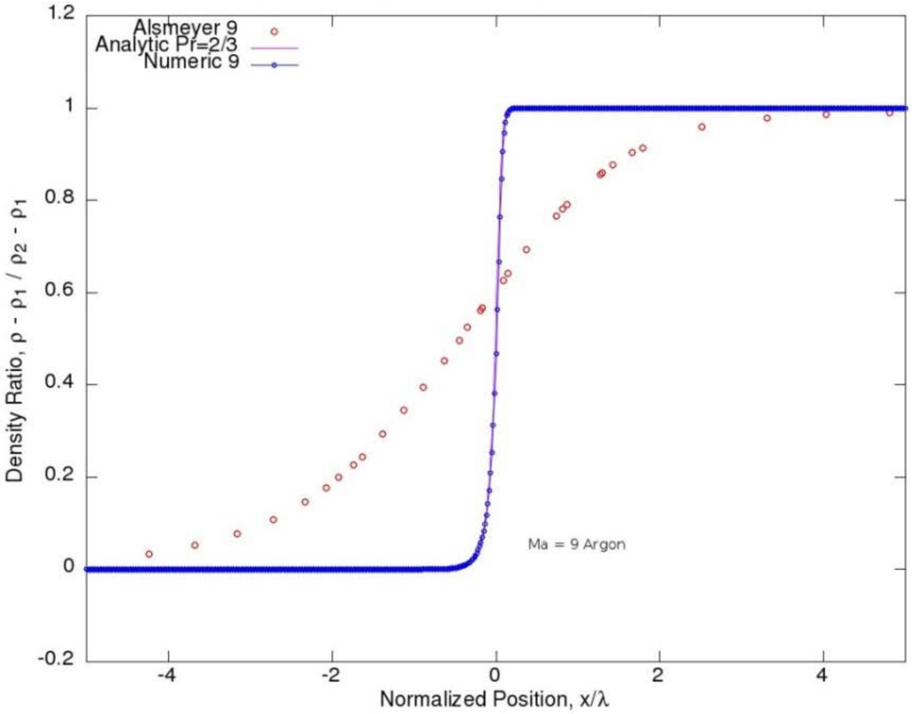

Figure 3.1.3.8: Comparison of Analytical and Numerical Results at Constant Viscosity at Mach number 9. This figure shows that when using constant viscosity sub-model, shock wave profile matches analytical simulation, even under large march number.

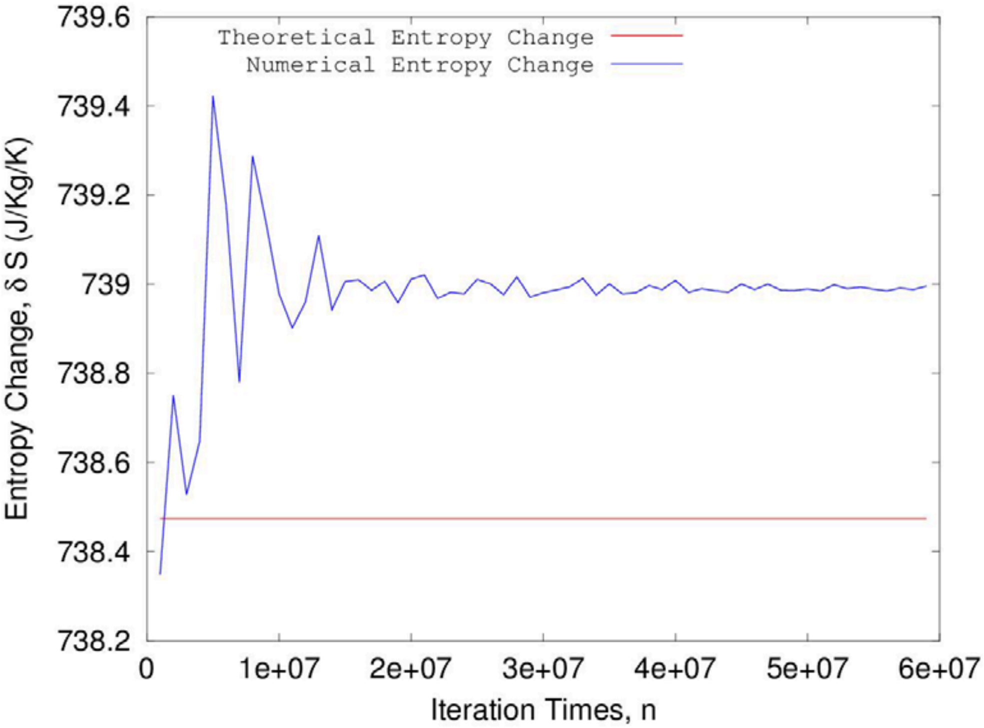

Figure 3.1.3.9: Entropy Change Error at Constant Viscosity. Node = 9600, hx = 1/16.

The figure also shows that errors of entropy change converge within iteration increasing and entropy changes get stable over 30M iterations. According to figure, theoretical value is around 738.4 J/Kg/K, numerical values converge at 739 J/Kg/K.

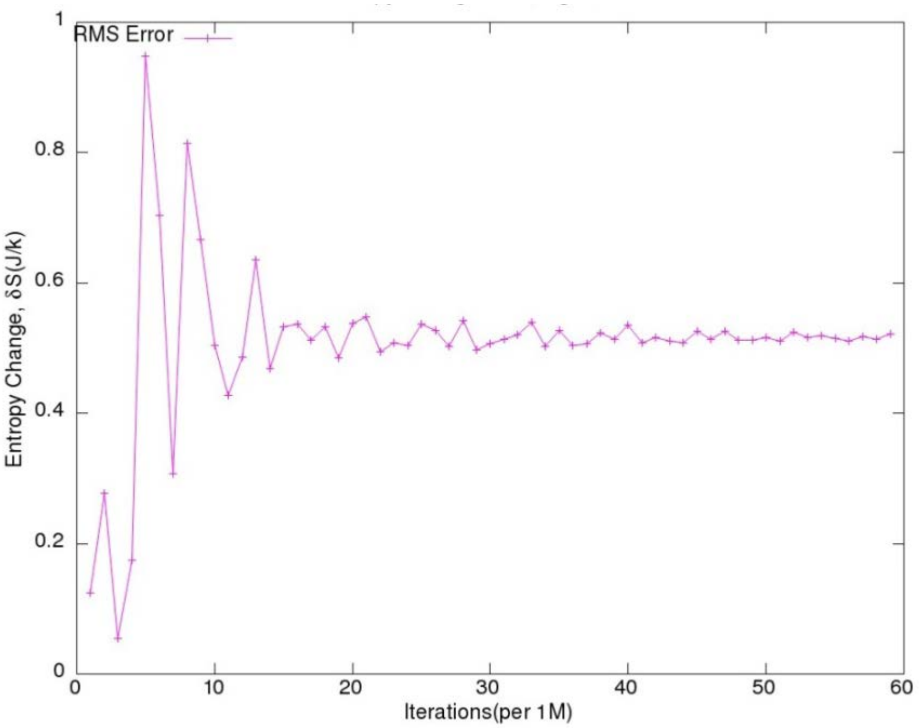

Figure 3.1.3.10: RMS Error of Entropy Changes at Constant Viscosity, Mach number 9.

The figure shows that RMS Errors get stable over 30M iterations and around 0.5. Difference between top and bottom is around 1 J/Kg/K. In order to keep the accuracy of simulation, the total number of nodes was increased to 19,200 for the Mach number 9.0 simulation in order to insure the shock slope was well resolved.

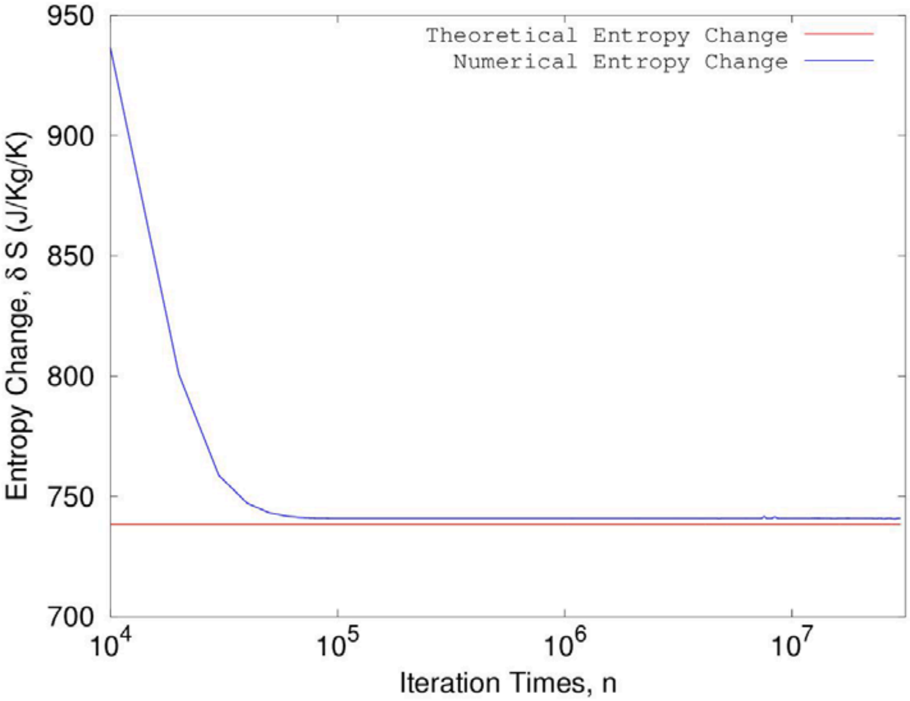

Figure 3.1.3.11: Entropy Change Error within 30M Iteration. Nodes = 19200, hx = 1/32.

According to figure, theoretical value is around 738.4 J/Kg/K, numerical values converge at 740.8 J/Kg/K for constant viscosity. Compared with Figure 2.7.1.2.12, convergence process under this scale is much faster, which is much shorter than 30M iteration to get stable state, while with larger entropy variation, which is around 200 J/Kg/K.

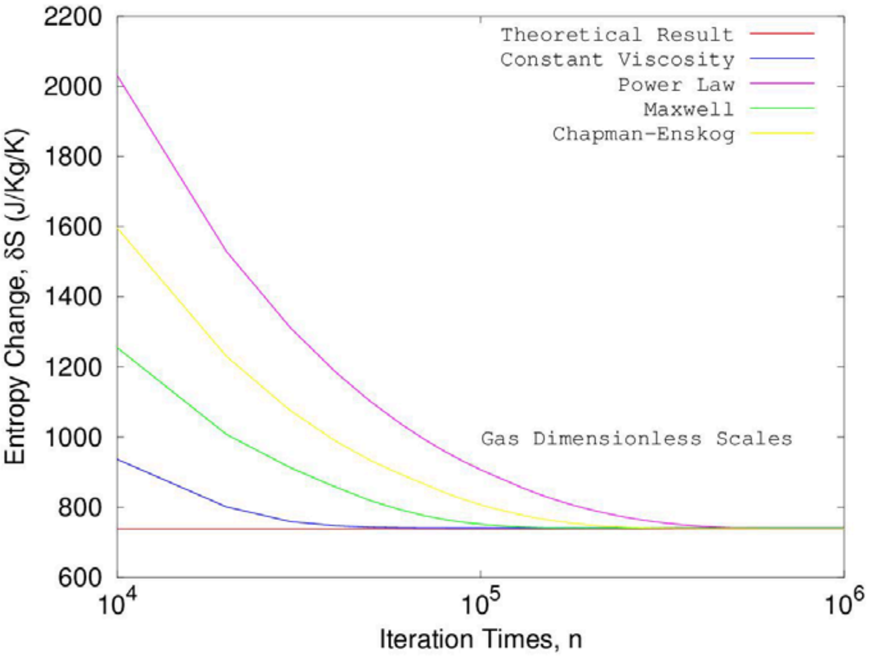

Figure 3.1.3.12: Entropy Change Trends of Different Viscosity Models, Nodes=19200, hx = 1/32. Entropy changes of these sub-models converge in 1 million iteration and approach to theoretical value.

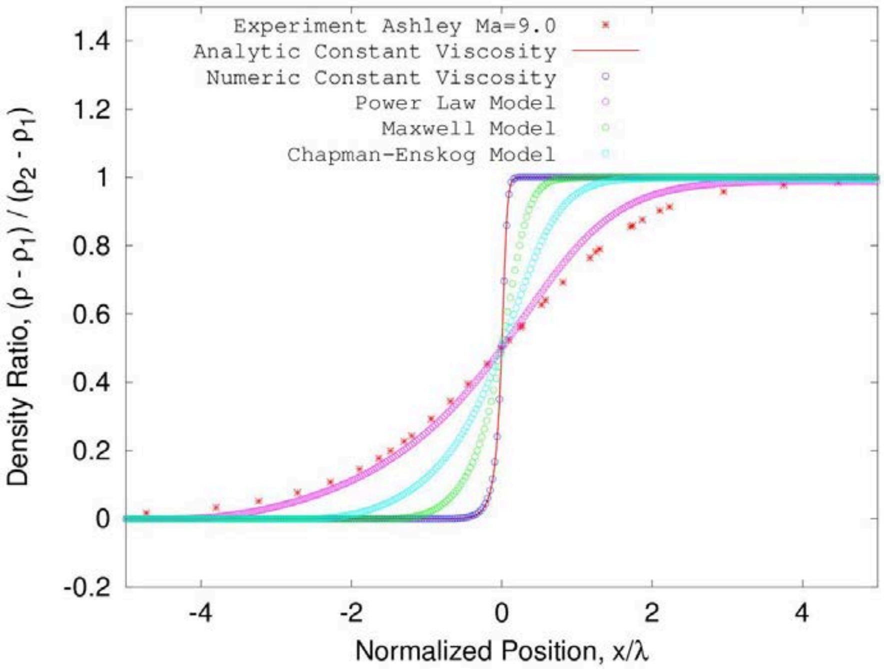

Figure 3.1.3.13: Comparison of Different Viscosity Sub-Models Profiles, Ma = 9.0, Nodes = 19200, hx = 1/32. Domain = 10λ.

According to the above figure, under Mach number 9.0, shock wave thickness ranges between 10 to 20 times of mean free path of argon. These four sub-models profiles are also set in the area between experimental profile and analytical profile. Power law model is closest to experimental profile, while with largest entropy change.

Table 3.1.1.1 Value of Entropy Change, Argon, Specific Gas Dimensionless Scale

Mach Number	1.55	3.38	9.0
Nodes Number	4800	9200	19200
hx	1/8	1/16	1/32
	Entropy Change (J/Kg/K)		
Theoretical Solution	16.51354	224.62240	738.47375
Constant Viscosity	16.46207	223.53973	740.88910
Power Law Model	16.50063	224.52104	738.56009
Maxwell Model	16.46703	224.30592	740.19109
Chapman-Enskog Model	16.47625	224.48048	739.18252

3.2 General Dimensionless Scales Based Simulation

3.2.1 Argon Simulation Analysis – General Dimensionless Scales

In order to keep the same B.C. as chapter 2.7.1 for argon,

Left hand side B.C.

$$u = Ma,\ P = \frac{1}{\gamma},\ \ T = \frac{1}{\gamma(\gamma-1)}, \rho = 1.0, e = T$$

Right hand side boundary condition is also can be determined Rankine-Hugoniot jump equations.

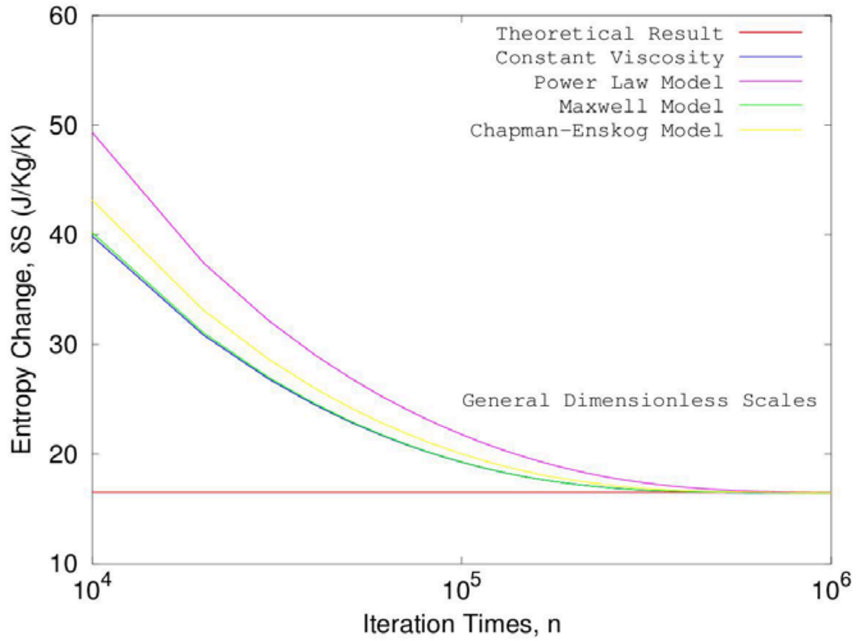

Figure 3.2.1.1: Entropy Change Trends of Different Viscosity Models, Nodes 4800, hx = 1/8, Ma = 1.55. Entropy changes of these sub-models converge in 1M iteration and approach to theoretical value 16.4917 J/Kg/K. Each sub-model's stable entropy change value is shown in Table 3.2.1.1

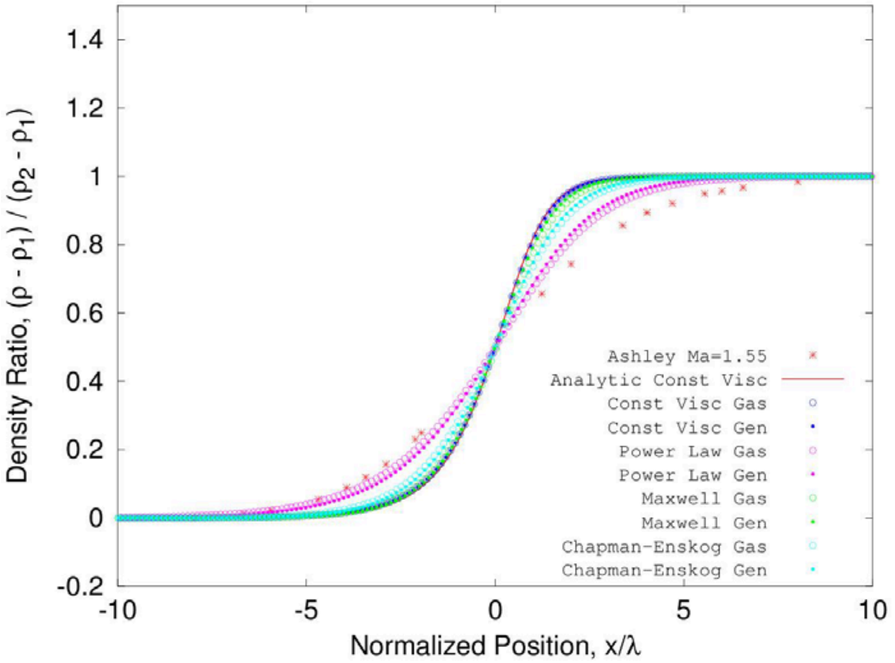

Figure 3.2.1.2 Comparison of Different Dimensionless Scales, Ma = 1.55, Domain = 20 λ.

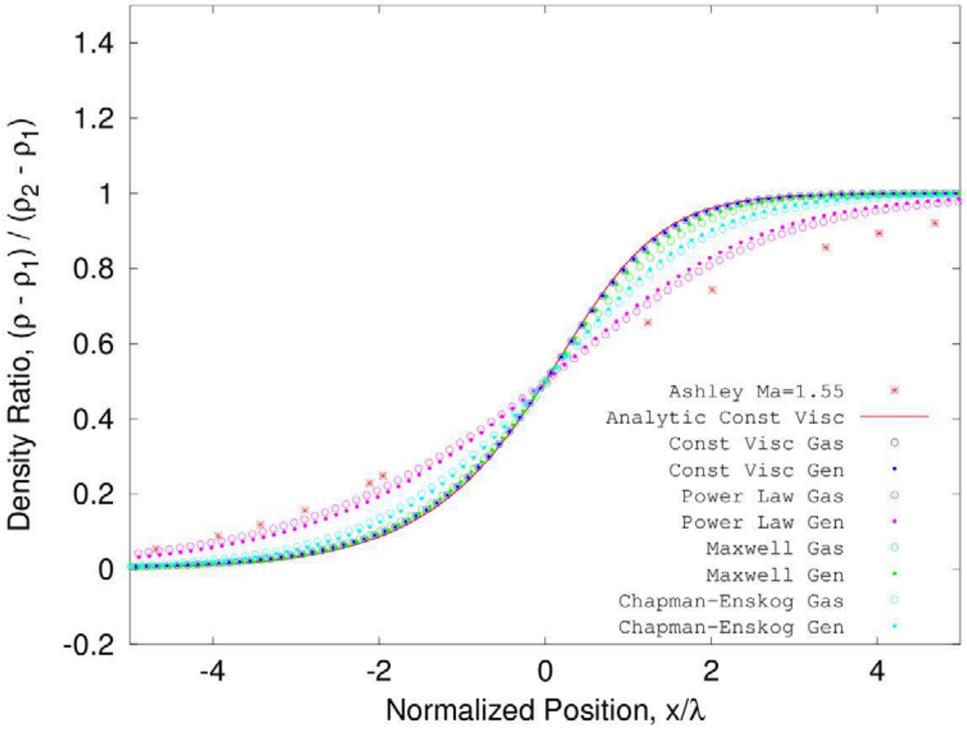

Figure 3.2.1.3: Comparison of Different Dimensionless Scales. Ma = 1.55, Domain = 10λ.

According to the above figure, under Mach number 1.55, the thicknesses of two shock wave profiles are larger than10 times of mean free path of argon. These four sub-models profiles are all set in the area between experimental profile and analytical profile, even thought, profiles under the same B.C. are mismatch. This condition is caused by two types of dimensionless scales. Power law model is closest to experimental profile, while with largest entropy change.

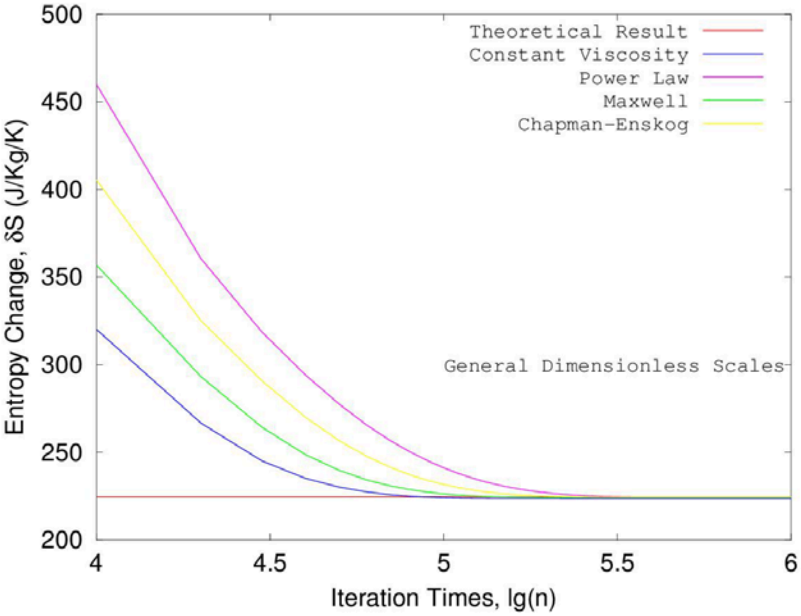

Figure 3.2.1.4: Entropy Change Trends of Different Viscosity Models, Nodes = 9600, hx = 1/16, Ma=3.38. Entropy changes of these sub-models converge in 1M iteration and approach to theoretical value 16.4917 J/Kg/K. Each sub-model's stable entropy change value is shown in Table 3.2.1.1.

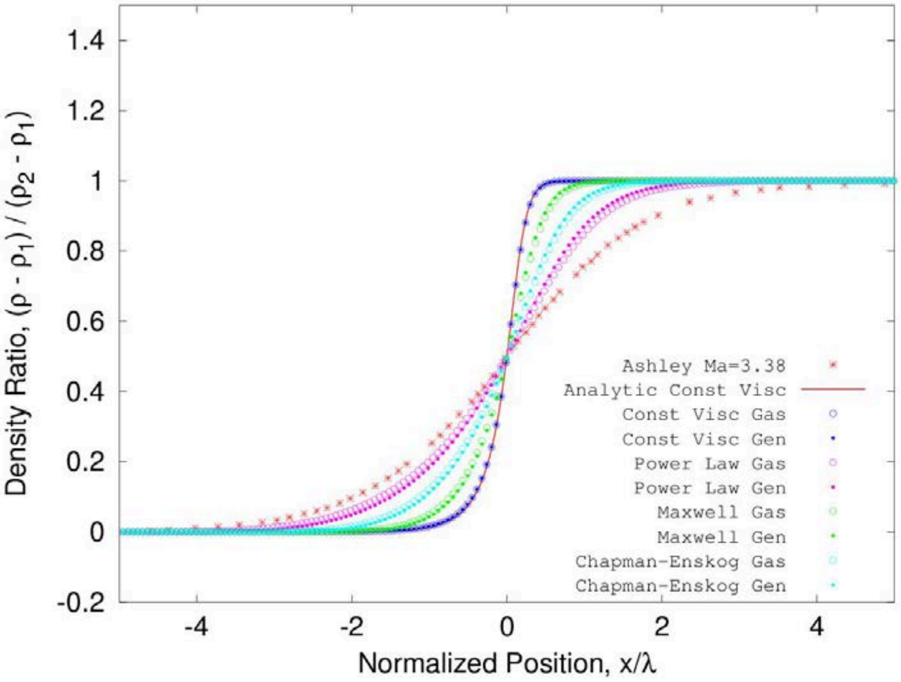

Figure 3.2.1.5: Comparison of Different Dimensionless Scales. Ma = 3.38.

Comparing with profiles of Mach number 1.55, all the thicknesses of shock wave profiles are smaller than10 times of mean free path of argon. These four sub-models profiles are all set in the area between experimental profile and analytical profile, Power law model is closest to experimental profile, while with largest entropy change.

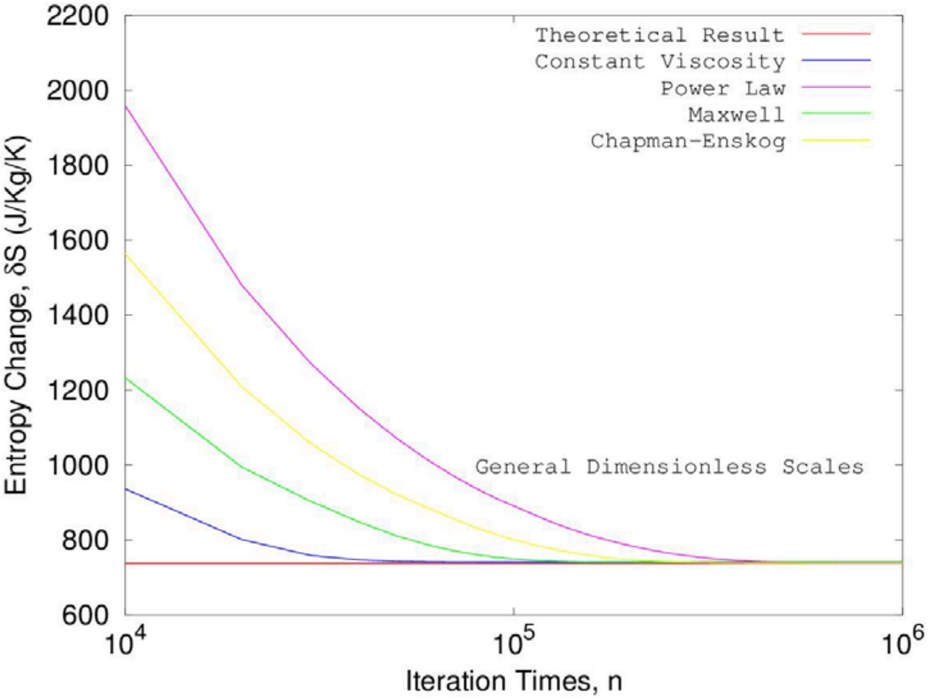

Figure 3.2.1.6: Entropy Change Trends of Different Viscosity Models, Nodes = 19200, hx = 1/32, Ma = 9.0. Entropy changes of these sub-models converge in 1M iteration and approach to theoretical value 16.4917 J/Kg/K. Each sub-model’s stable entropy change value is shown in Table 3.2.1.1

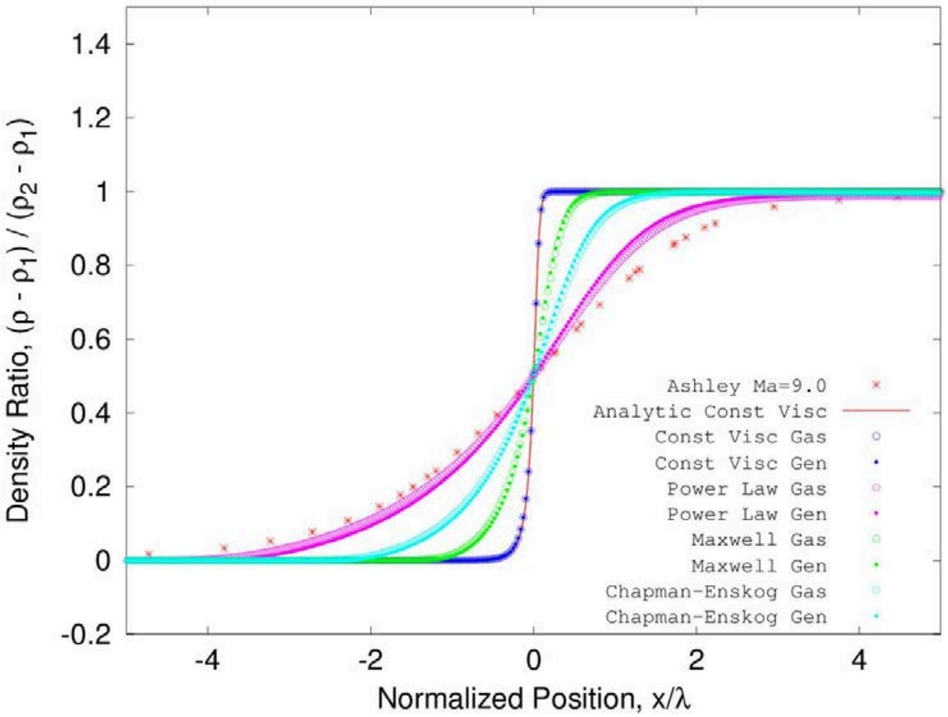

Figure 3.2.1.7: Comparison of Different Dimensionless Scales. Ma = 9.0. Comparing with profiles of Mach number 1.55 and 3.38, profiles under the same B.C trend to match each other.

Table 3.2.1.1 Value of Entropy Change, Argon, General Dimensionless Scale

Mach Number	1.55	3.38	9.0
Nodes Number	4800	9200	19200
hx	1/8	1/16	1/32
	Entropy Change (J/Kg/K)		
Theoretical Solution	16.49170	224.54799	738.31049
Constant Viscosity	16.46207	223.53973	740.88910
Power Law Model	16.49263	224.51563	738.65994
Maxwell Model	16.46400	224.27904	740.40087
Chapman-Enskog Model	16.47370	224.47266	739.28363

3.2.2 Air Simulation Analysis – Ideal Gas EOS

Parameters evaluation of air are shown as Table 3.2.2.1

Table 3.2.2.1 Parameters Evaluation, Air, Nodes = 4800, hx = 1/8.

Mach#	γ_0	ω	Pr	s	C_0
1.4	0.4	0.77	0.715	0.968793	293.8619

Dimensional left hand side B.C. for air simulation is shown in Table 3.2.2.2.

Table 3.2.2.2 Parameters Evaluation, Air, Nodes = 4800, hx = 1/8.

Mach#	ρ_1(Kg/m^3)	P$_1$(Pa)	T$_1$(K)	u_1(m/s)
1.4	1.205	103.320	300.0	486.064193

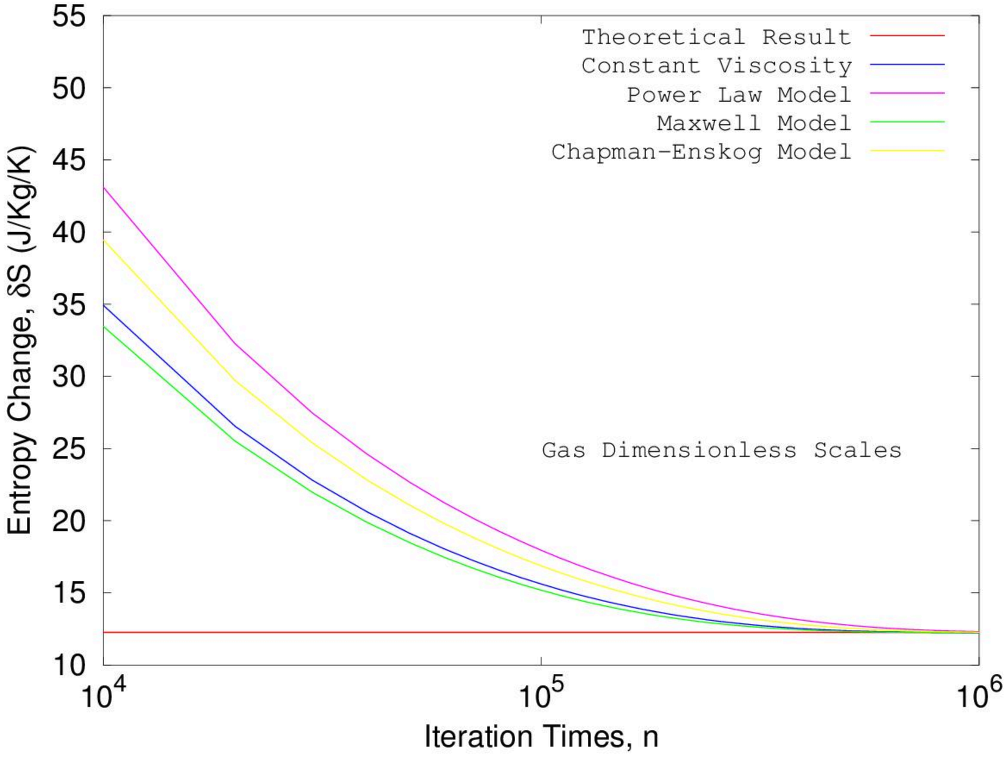

Figure 3.2.2.1: Entropy Change Trends of Different Viscosity Models, Air, Ma = 1.4, Nodes 4800, hx = 1/8.

Entropy changes of these sub-models converge in 1M iteration and approach to theoretical value 12.25622 J/Kg/K. Each sub-model's stable entropy change value is shown in Table 3.2.2.3.

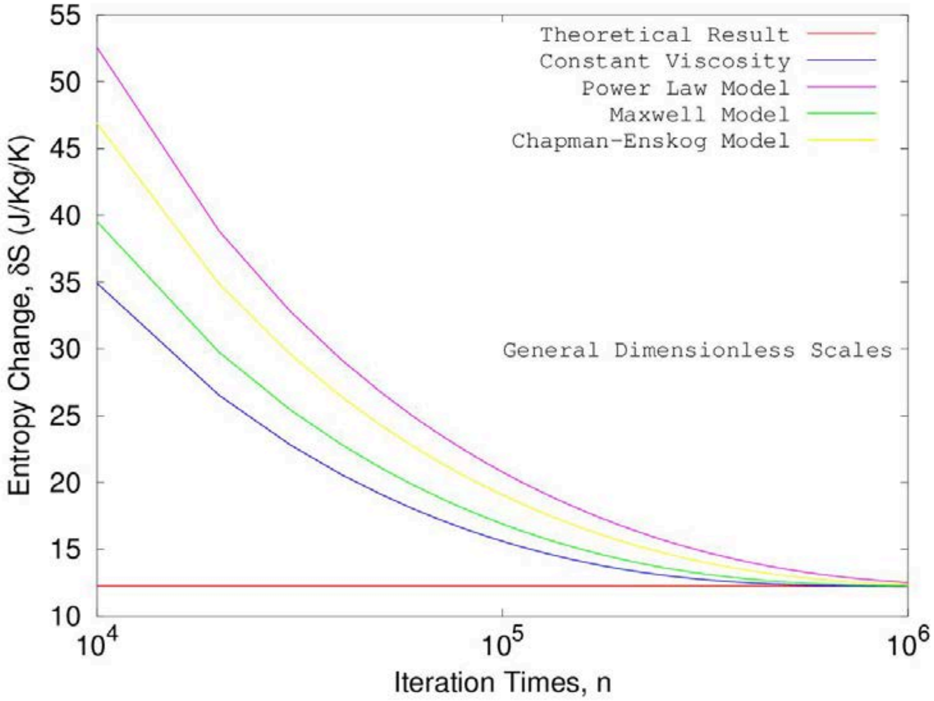

Figure 3.2.2.2: Entropy Change Trends of Different Viscosity Models, Air, Ma = 1.4, Nodes 4800, hx = 1/8. Entropy changes of these sub-models converge in 1M iteration and approach to theoretical value 12.25622 J/Kg/K. Each sub-model's stable entropy change value is shown in Table 3.2.2.3.

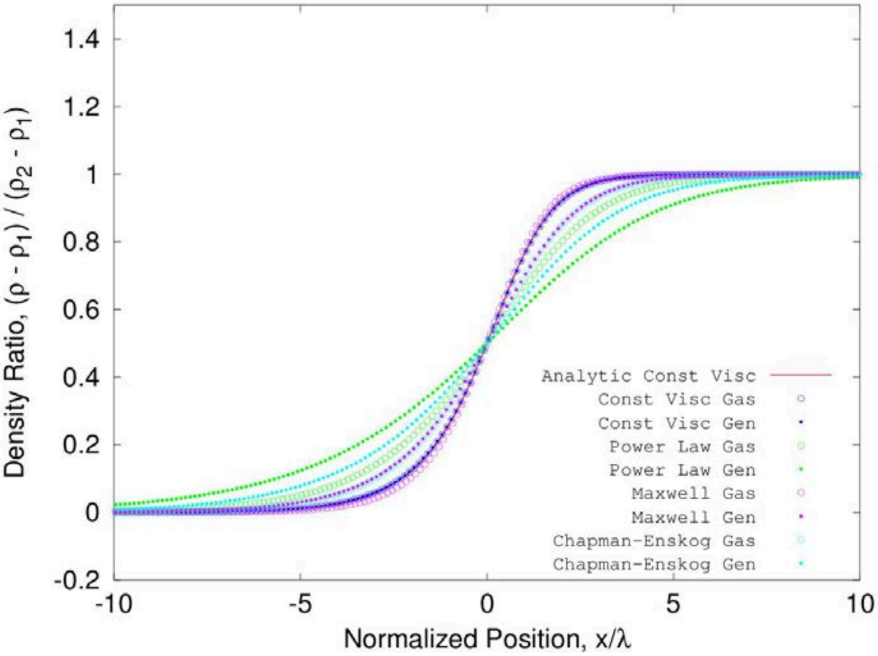

Figure 3.2.2.3: Comparison of Different Dimensionless Scales. Ma = 1.4, Nodes = 4800, hx = 1/8, Domain 20 λ.

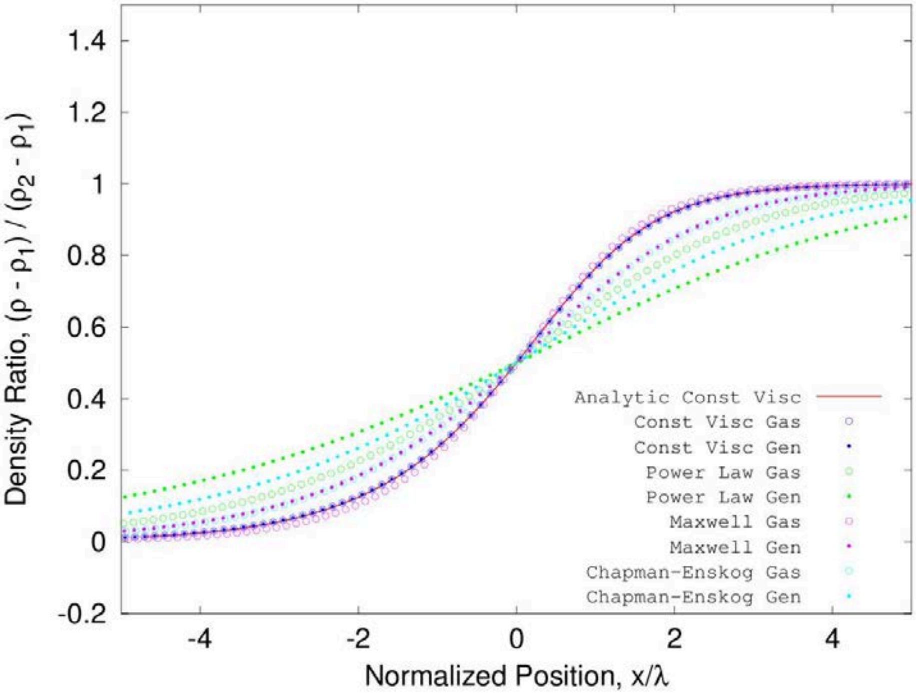

Figure 3.2.2.4: Comparison of Different Dimensionless Scales. Ma = 1.4, Nodes = 4800, hx = 1/8, Domain 10λ.

Comparing with profiles of argon, these four sub-models profiles have similar trends. Shock wave thicknesses of Power Law profile are 20 times of mean free path of air. Power law model has largest entropy change.

Table 3.2.2.3 Value of Entropy Change, Air, Ideal Gas EOS

Mach Number	1.4	
Nodes Number	4800	
hx	1/8	
Viscosity Sub-Models	Entropy Change (J/Kg/K)	
	Gas Dimensionless Scales	General Dimensionless Scales
Theoretical Solution	12.25622	12.25622
Constant Viscosity	12.24533	12.24533
Power Law Model	12.30687	12.49248
Maxwell Model	12.23883	12.27202
Chapman-Enskog Model	12.27070	12.36467

3.3 Mie-Grüneisen EOS Model - Argon

In order to research the availability of M-G EOS on gas medium, parameter s and C0 need to be determined as chapter 2 mentioned. Mach number 1.55 is set to keep same B.C. in ideal gas EOS simulation.

Table 3.3.1 Parameters Evaluation, Argon, Nodes = 4800, hx = 1/8.

Mach#	s	C_0 (m/s)
1.55	1.00095565	207.664409

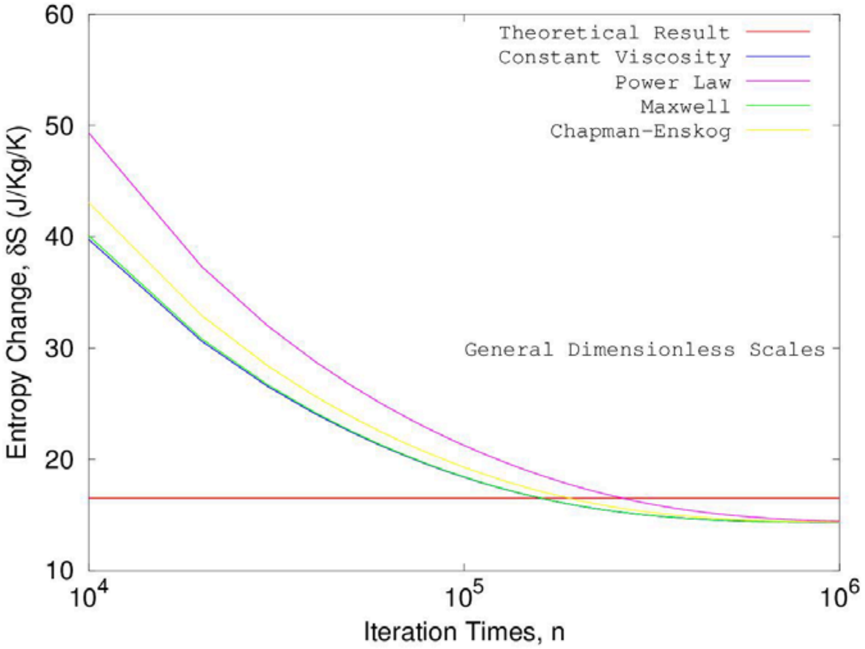

Figure 3.3.1: Entropy Change Trends of Different Viscosity Models, M-G EOS, Argon, Ma = 1.55, Nodes 4800, hx = 1/8.

Entropy changes of these sub-models converge in 1M iteration and approach to value 14.33 J/Kg/K, which is off the theoretical value of entropy change 16.49 J/Kg/K from ideal gas EOS.

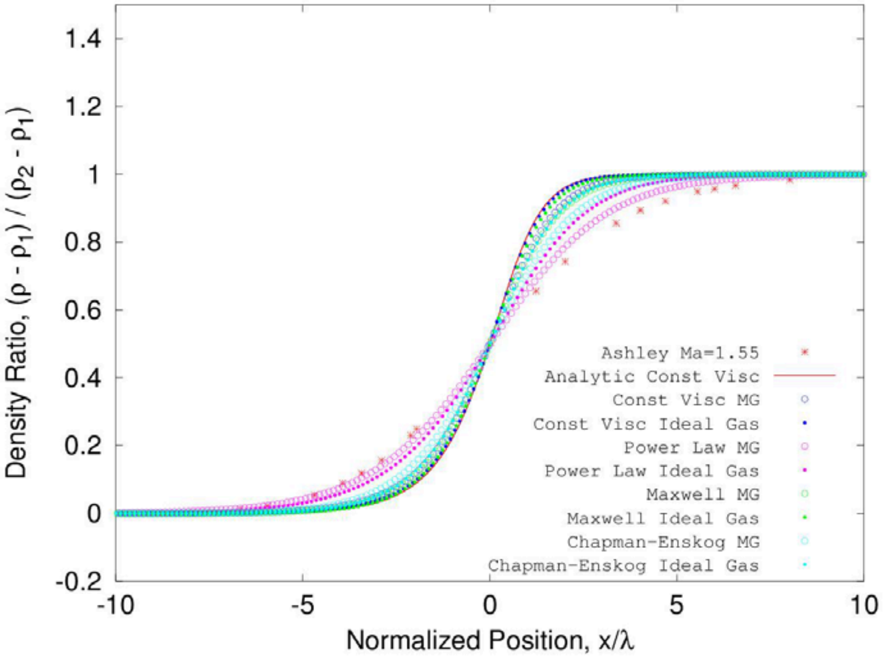

Figure 3.3.2: Comparison of Different Dimensionless Scales, M-G EOS, Argon, Ma = 1.55, Nodes = 4800, hx = 1/8, Domain 20 λ, λ is the mean free path of argon.

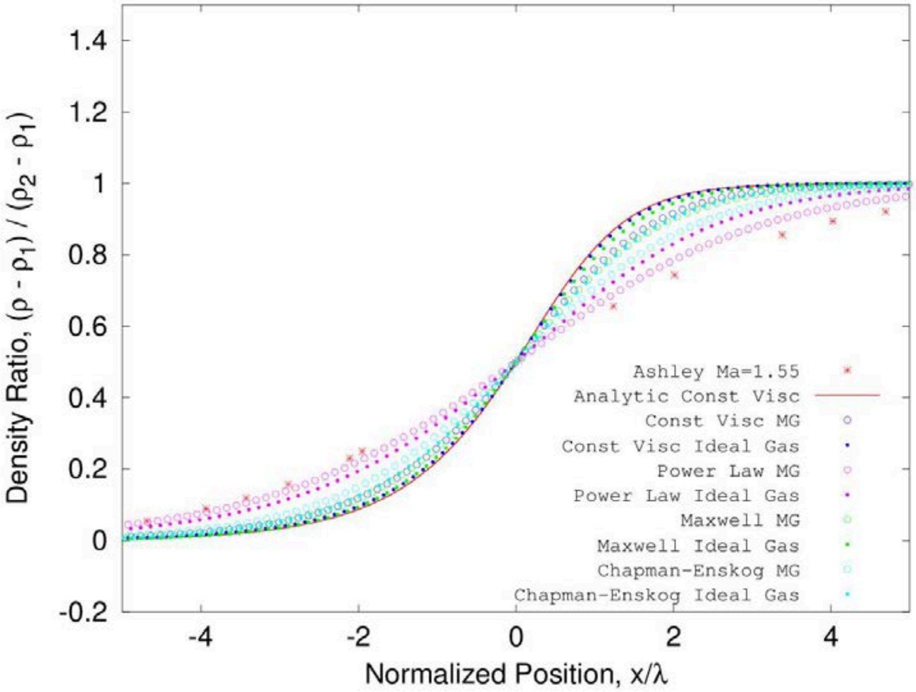

Figure 3.3.3: Comparison of Different Dimensionless Scales, M-G EOS, Argon, Ma = 1.55, Nodes = 4800, hx = 1/8, Domain 10λ.

The shock wave density profiles are similar to those from ideal gas EOS, Power Law model is the closet one to experimental result, then Chapman-Enskog model. Maxwell model almost overlaps with constant viscosity model. The difference is that all the viscosity sub-models in M-G EOS are more approaching experimental result. Especially; Power Law model result is closer to experimental result. Shock wave thicknesses range from 10 to 15 times of mean free path.

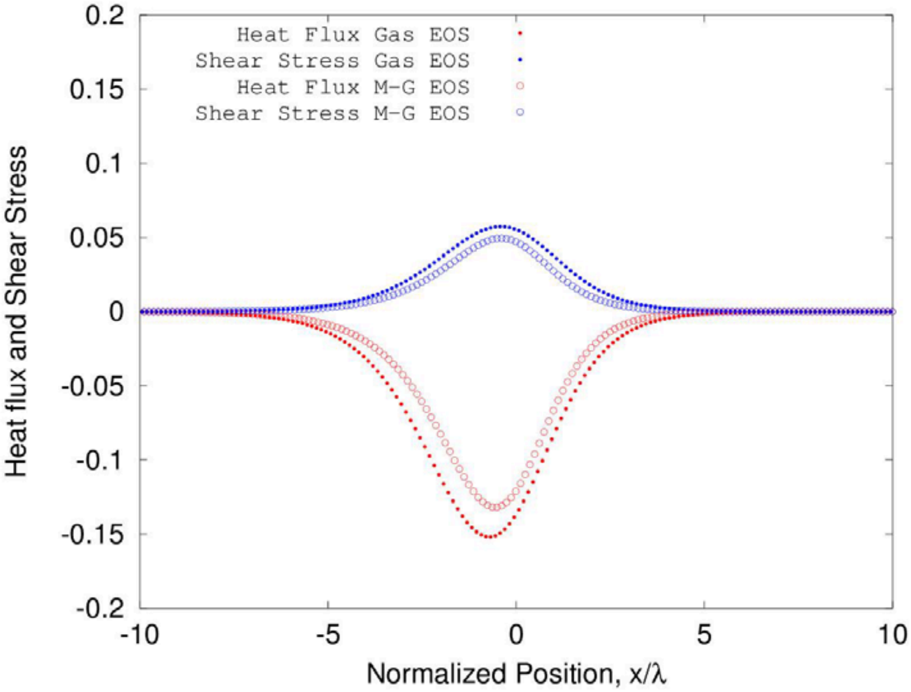

Figure 3.3.4: Heat Flux and Stress relationship, Constant Viscosity, Argon, Ma = 1.55, Nodes = 4800, hx = 1/8. M-G EOS result has smaller absolute value in both heat flux and stress.

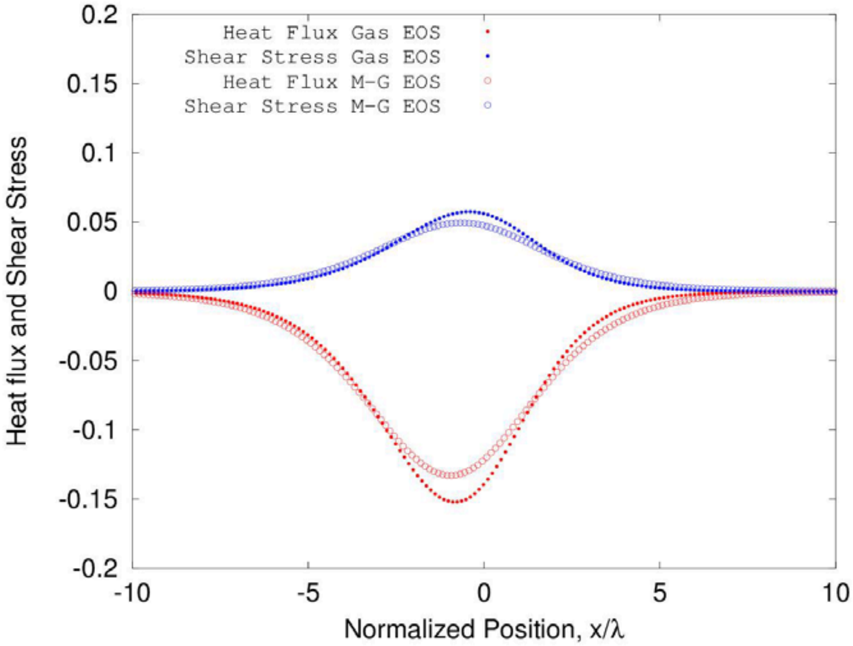

Figure 3.3.5: Heat Flux and Stress relationship, Power Law model, Argon, Ma = 1.55, Nodes = 4800, hx = 1/8.

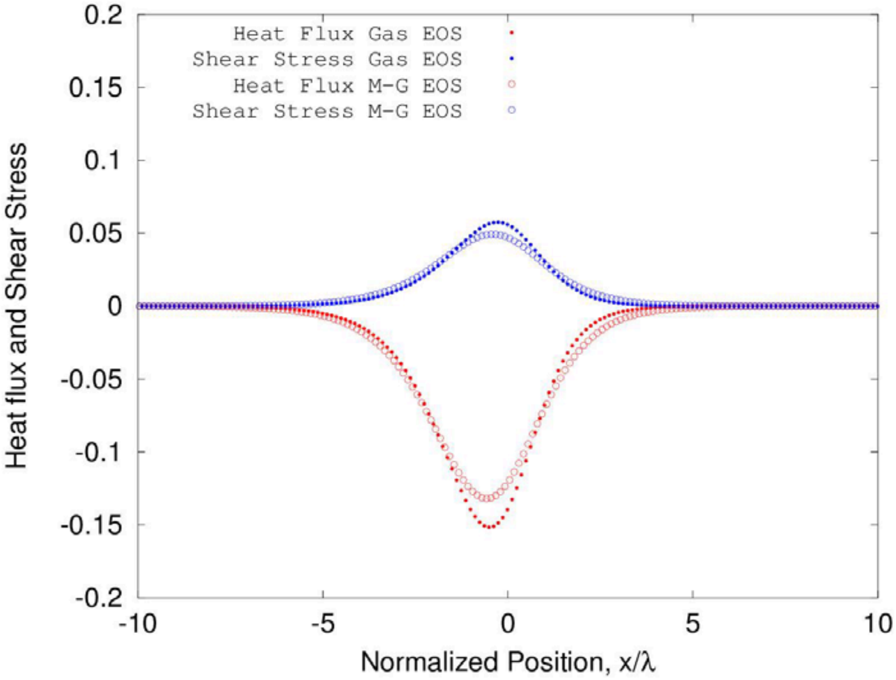

Figure 3.3.6: Heat Flux and Stress relationship, Maxwell model, Argon, Ma = 1.55, Nodes = 4800, hx = 1/8.

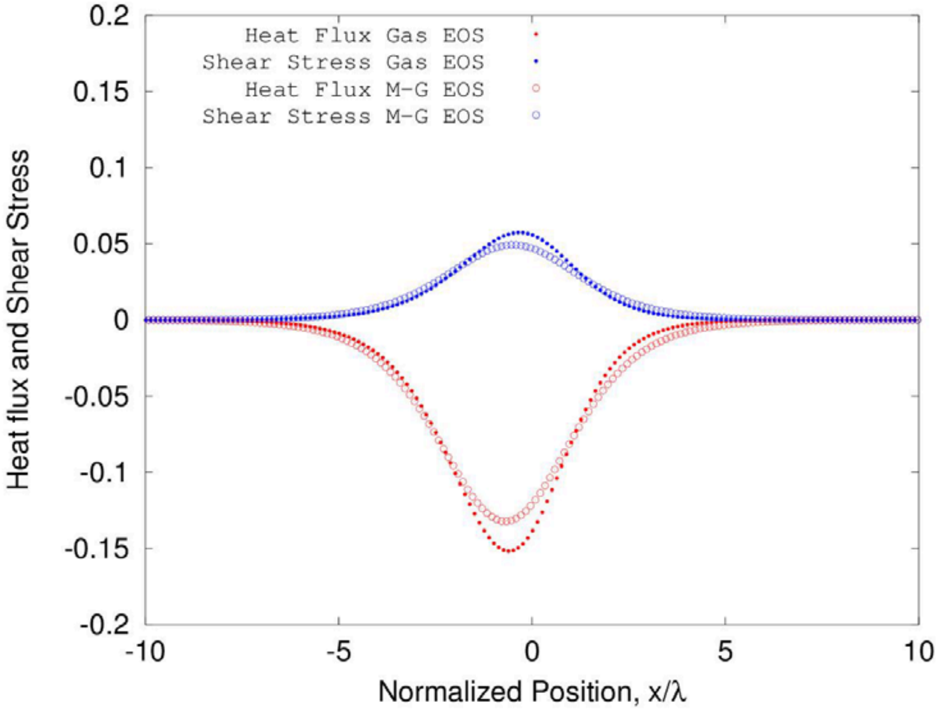

Figure 3.3.7: Heat Flux and Stress relationship, Chapman-Enskog model, Argon, Ma = 1.55, Nodes = 4800, hx = 1/8.

Table 3.2.2.3 Value of Entropy Change, Argon, General Dimensionless Scales

Mach Number	1.55	
Nodes Number	4800	
hx	1/8	
Viscosity Sub-Models	Entropy Change (J/Kg/K)	
	Ideal Gas EOS	M-G EOS
Theoretical Solution	16.49170	16.49170
Constant Viscosity	16.46207	14.33946
Power Law Model	16.49263	14.43678
Maxwell Model	16.46400	14.34159
Chapman-Enskog Model	16.47370	14.36217

3.4 Summary

In chapter 3, two types of dimensionless scales are used and compared in argon and air simulation. Four types of viscosity modes are used to simulated shock profiles in argon and air for different Mach numbers. Entropy change errors of these sub-models are smaller than ten thousandth. The Power Law model profile is closest to the experimental result, followed with Chapman-Enskog and Maxwell model. The power law is derived from a fit to experimental data whereas Maxwell model and Chapman-Enskog model are hard sphere and soft sphere theoretical models respectively. Shock wave thickness becomes thinner when Mach number increases and as a result more difficult to resolve.

M-G EOS and Ideal Gas EOS are derived based on different assumption and physical principles from statistic mechanics. M-G EOS mainly focus on the internal energy contributed from vibration energy of oscillators, while for gas, most of the internal energy are contributed by translational kinetic energy not rotation or vibration energy. That’s one main explanation for the differences in entropy change across the shock for the different equations of state.

Chapter 4

Mie-Grüneisen EOS Model - Water

4.1 Experimental Geometry

Measurement of shock wave pressure is conducted by University of Cambridge [13]. Figure 4.1.1 is the geometry of experiment.

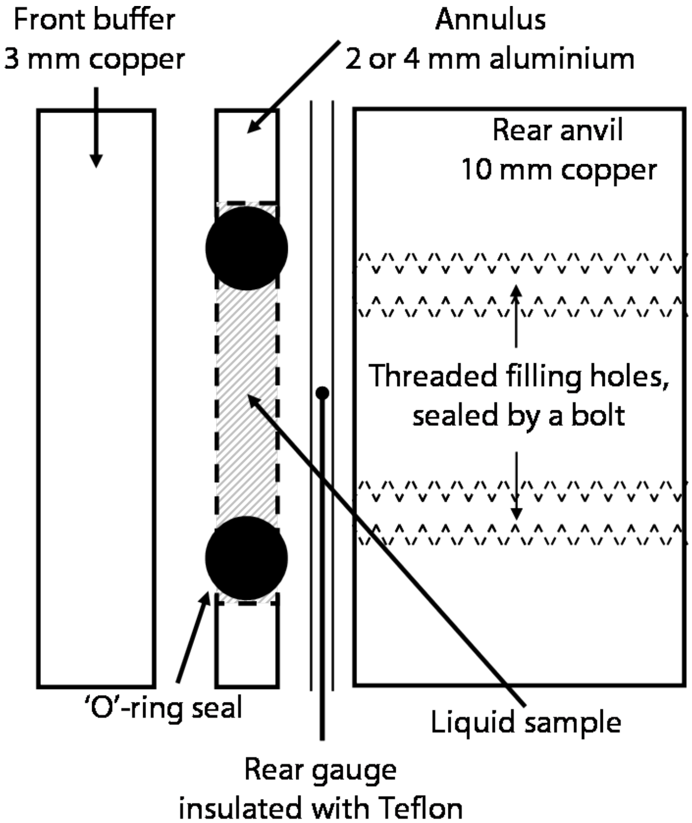

Figure 4.1.1: Experiment schematic of shock wave pressure in water.

Copper flyer hitting the O-ring causes shock wave transmitting in water, which is detected by rear gauge.The initial temperature for stationary water is 18 ± 2C. The copper flyer is traveling at 295 m/s towards O-ring. The thickness of gauge is neglected.

4.2 Determination of Shock Wave Velocity

An impedance matching technique in pressure-particle velocity, P-Up, space is used to determine shock velocity in water. [8]

$$P_{water} = \rho_{water} C_{water} U_p + \rho_{water} S_{water} U_p^2 \qquad (4.2.1)$$

$$P_{copper} = \rho_{copper} C_{copper} (U_{flyer} - U_p) + \rho_{copper} S_{copper} (U_{flyer} - U_p)^2 \qquad (4.2.2)$$

Solving equation (4.2.1) and equation (4.2.2), Up and Us can be derived. Table 4.2.1 and Figure4.2.1 are results of impedance match.

Table 4.2.1 Parameters and results of impedance match

Material	Flyer velocity(m/s)	Temperature(k)	S	C(m/s)	ρ(kg/m^3)	Up(m/s)
Copper	295	293.15	1.49	3940	8930	277.8668
Water			1.92	1650	1000	

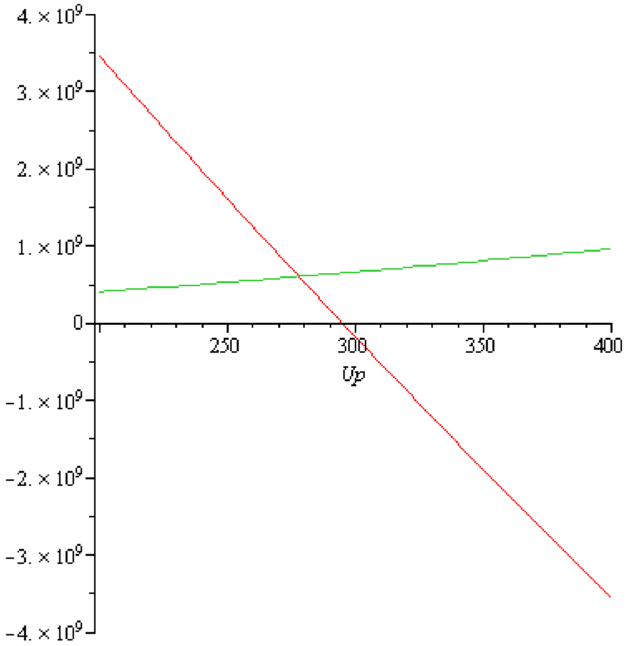

Figure 4.2.1: Impedance match of Us-Up lines.

The intersection is the final condition after impact. Up is the particle velocity of water, which can be used to calculate shock wave velocity in water. According to chapter 2, pre-shock and post-shock flow velocities, U1 and U2, can be converted from known Us and Up values.

4.3 Mie-Grüneisen Parameter

As chapter 2 mentioned, boundary conditions of shock wave in water also can be determined by Rankine-Hugoniot jump condition equations. The Mie-Grüneisen EOS is a function incorporating pressure, internal energy, Hugoniot slope, bulk sound speed, strains and Mie-Grüneisen gamma. As all the parameters except M-G gamma are determined though the above impedance matching process, M-G gamma can be derived based on the assumption that shock wave boundary conditions are both available in Rankine-Hugoniot jump condition equations and M-G EOS as shown in Figure 4.3.1.

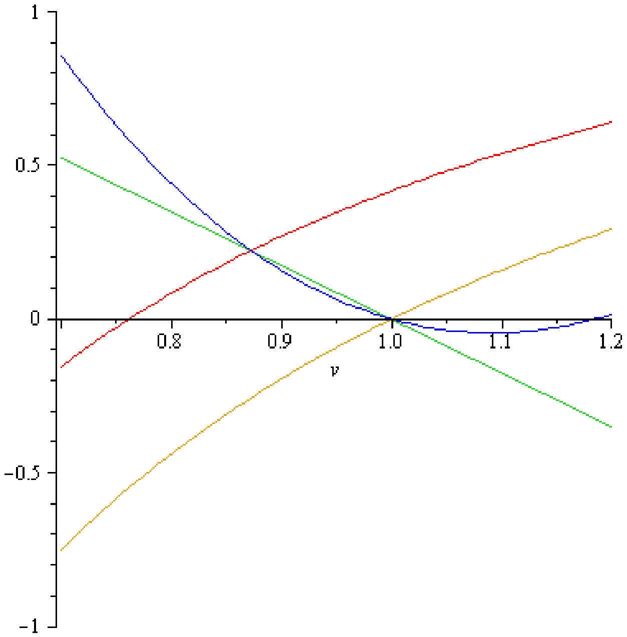

Figure 4.3.1: Dimensionless impedance match to derive M-G gamma. X axis is the specific volume; Y axis is the pressure. Blue is M-G EOS; Green is Rayleigh line; Red is post-shock pressure and yellow is pre-shock pressure.

4.4 Relation of Bulk Viscosity and Dilatational Viscosity

During the research of shock wave in water, in order to get better understanding and make use of literature data accurately, one of the difficulties is that notations in literatures can be different. For liquid, stress is not only related to first viscosity but also related to second viscosity. First viscosity is the dynamic viscosity. Second viscosity has at least two kinds of notations, bulk viscosity or dilatational viscosity. Bulk viscosity is used in *Compressible-Fluid Dynamics, Philip.A.T,* expressed in viscous stress equation

$$\sum\nolimits_{ik} = 2\mu\left(D_{ik} - \frac{1}{3}\delta_{ik}D_{mm}\right) + \mu_v\delta_{ik}D_{mm}, \tag{4.4.1}$$

Where,μ is the first viscosity,

$$D_{ik} = \frac{1}{2}\left(\frac{\partial v_j}{\partial x_i} + \frac{\partial v_i}{\partial x_j}\right), \tag{4.4.2}$$

$$D_{mm} = \left(\frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} + \frac{\partial v_z}{\partial z}\right), \tag{4.4.3}$$

Replace D_{ik} and D_{mm} in equation (4.4.1) and rearrange it, get

$$\sum\nolimits_{ik} = \mu\left(\frac{\partial v_j}{\partial x_i} + \frac{\partial v_i}{\partial x_j}\right) - \left(\frac{2}{3}\mu - \mu_v\right)\left(\frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} + \frac{\partial v_z}{\partial z}\right)\delta_{ik} \tag{4.4.4}$$

Equation (4.4.4) is just opposite to the equation in *Transport Phonomena, 2nd Edition, R.B.Bird.*

$$\tau_{ik} = -\mu\left(\frac{\partial v_j}{\partial x_i} + \frac{\partial v_i}{\partial x_j}\right) + \left(\frac{2}{3}\mu - \mu_v\right)\left(\frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} + \frac{\partial v_z}{\partial z}\right)\delta_{ik} \tag{4.4.5}$$

So, for one dimensional shock wave, simplified stress is

$$\Pi_{xx} = \sum\nolimits_{xx} = -\tau_{xx} \tag{4.4.6}$$

As a result, bulk viscosity is the same as dilatational viscosity in same coordinate.

$$\kappa = \mu_v \tag{4.4.7}$$

Second viscosity reference value is from *Compressible-Fluid Dynamics, Philip.A.T, Table 1.1.*

4.5 Numerical Simulation

Left hand side dimensional B.C.

$u = Ma,\ P = 1\ atm\ ,\ \ T = 300k, \rho = 1000kg/m^3, e = \frac{P}{\gamma_0}$

where, γ_0 is M-G gamma, e is internal energy per unit mass.

Right hand side B.C. is determined by impact impedance match and Rankine-Hugoniot jump equations.

Parameters evaluations for water are presented in Table 4.5.1 and Dimensionless B.C.s of water are shown in Table 4.5.2.

Table 4.5.1 Parameters Evaluation, Water

Mach#	Up	S	C0	γ_0	uv	h_x	$Nodes$
1.323	277.867	1.920	1650	4.984	3.100	1/8	4800

Table 4.5.2 Dimensionless B.C. of Water, Ma = 1.32

	u	ρ	P	T	ie
LHS	1.323	1.000	3.722e-5	0.455	7.467e-6
RHS	1.155	1.146	0.223	0.469	0.780
Ratio	0.872	1.146	5988.894	1.031	104459.622

4.6 Sensitivity Analysis

Datas from NIST [15] are incorporated in numerical code to conduct sensitivity analysis of shock wave thickness.

4.6.1 Specific Heat at Constant Volume (Cv)

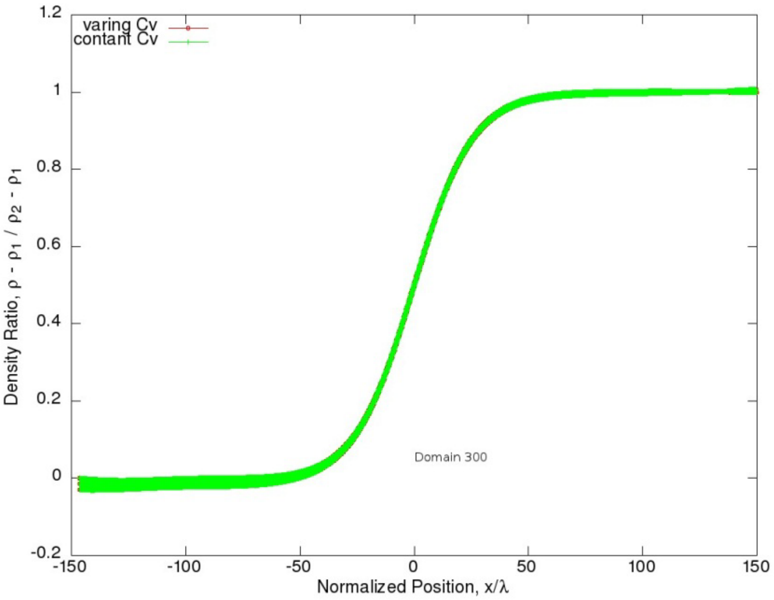

Figure 4.6.1: Specific Heat Effect on Shock Wave Thickness. This figure shows that varying specific heat contributes the same shockwave profile as constant specific heat. Conclusion is that shock wave thickness is not obviously sensitive to the specific heat.

4.6.2 Heat Conductivity (k)

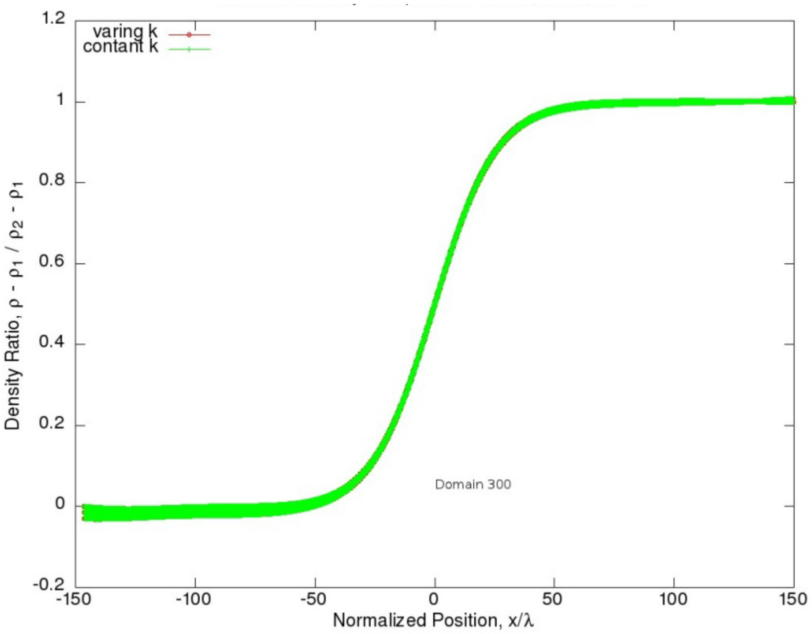

Figure 4.6.2: Heat Conductivity Effect on Shock Wave Thickness.

This figure shows that varying Heat Conductivity contributes the same shockwave profile as constant heat conductivity. Conclusion is that shock wave thickness is not obviously sensitive to the heat conductivity.

4.6.3 Viscosity (μ)

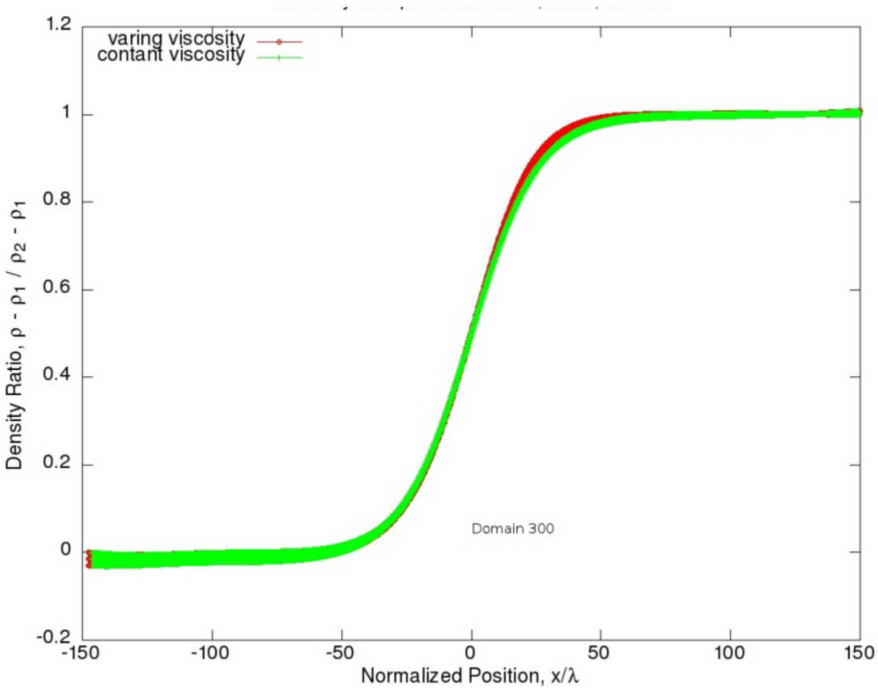

Figure 4.6.3: Viscosity Effect on Shock Wave Thickness.

This figure shows that varying viscosity contribute slightly difference to the profile compared with constant viscosity. As known, viscosity of water decreases when temperature rises. The slope of shock wave profile gets greater matching the conclusion from gas. So shock wave thickness is weakly sensitive to the viscosity.

4.6.3 Length Scale

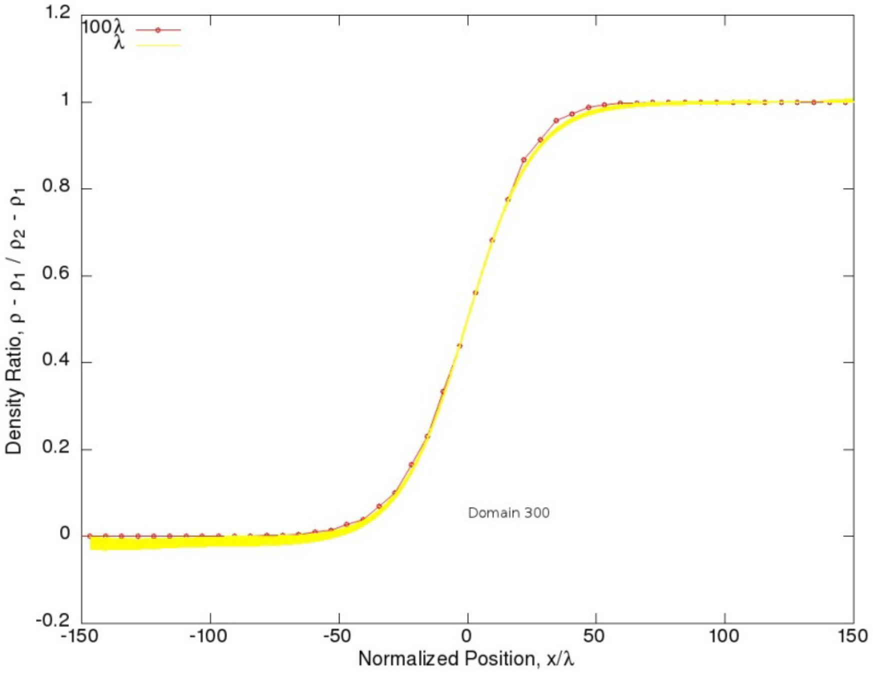

Figure 4.6.4: Effect of Length Scale on Shock Wave Thickness. This figure shows that varying lambda contribute the same shockwave profile as former molecular diameter. Conclusion is that shock wave thickness is not sensitive to the length scale.

Chapter 5

Conclusion

In order to develop a better understanding of the irreversibilities associated with the shock compaction of fluids, 3 materials (argon, air and water), 2 equations of state (EOS) (Ideal Gas EOS and Mie-Grüneisen EOS incorporated with Material EOS) and 4 viscosity sub models are studied in one-dimension via a numerical solution. The numerical solution is validated with the analytical solution for perfect gas behavior and compared to experimental data where available.

A non-dimensional numeric scheme based upon an explicit Eulerian finite volume method was developed. The spatial resolution converges at less than 1/8th of the mean free path, ensuring the stability of the numerical algorithm without resorting to implementing artificial dissipation. Space derivatives are approximated by second order central differences and time derivatives are approximated by first order forward differences.

Initial simulations focused on argon, i.e. a monatomic perfect gas. Simulation boundaries are set more than ten shock wave thickness away from the shock. Simulations indicate that shock wave thickness becomes thinner as Mach number increases up to a Mach number of 9. Simulated density profiles slightly differ from experimental results. From the shock profiles, the entropy change can be calculated. Entropy change provides a single metric as to the quality of the simulation, when compared to the theoretical entropy change. The theoretical entropy change for a perfect gas is around 0.3% different to the simulated entropy change of the viscous models considered in this work, the Maxwell model for viscosity gives results which are closed to the analytic solution, whereas the Power Law viscous model results in profiles that most resemble the experimental data. The Power Law viscous model results in the largest value of entropy change at low Mach numbers (Mach number 1.55 and 3.38), but the smallest value of entropy change at large Mach number (Mach number 9), compared with other viscous models.

Next, the shock wave profile in a non-monatomic gas, i.e. air, is simulated for Mach number 1.4. The results indicate similar phenomena as argon at low Mach number (Mach number 1.55).

While validating the Mie-Grüneisen (M-G) EOS for shock waves in fluids, several challenges are resolved. The shock Hugoniot equation, used as the reference curve for the M-G EOS for gas was derived. The relationship between equivalent M-G constant and the ratio of specific heat was derived. When modeling gaseous argon passing through a shock at Mach number 1.55, the M-G EOS predicts lower entropy change, 14.437 J/Kg/K, as compared to the ideal gas EOS, which predicts in 16.493 J/Kg/K. The main reason causing this difference is discussed in chapter 3. Finally, it is observed that heat flux dominates the entropy change for M-G EOS.

Finally the M-G EOS for water was numerically investigated and the results were compared to experimental data. The M-G constant is derived by impedance matching the M-G EOS and Rayleigh line. The bulk viscosity is also incorporated into the stress term. From a parametric study of transport properties, it was found that the shock wave thickness is weakly sensitive to the specific heat, heat conductivity, viscosity and grid dimension, whereas changes in the viscosity have the largest effect on the shock profile thickness. However, the shock wave thickness, even at varying viscosities still too thin compared to experimental data.

This research mainly focuses on deriving a suitable set of equations for doing direct numeric simulations of shock profiles in gasses and liquid and then numerically solving these equations using a finite volume technique. Simulations of shock wave in liquids and gases serve as a bridge to better understand the applicability of the M-G EOS for resolving the shock profiles in solids. This is especially true when considering the viscous dissipation mechanisms in solids across the shock. Further considerations from statistical mechanics might provide more insight in the future; by supporting assumptions with more precise governing equations; the simulated shock wave profiles may be better resolved.

REFERENCES

[1] VonNeumann, J.; Richtmyer, R. D.; 1950, "*A Method for the Numerical Calculation of Hydrodynamic Shocks*", J. of Appl. Physics, Vol. 21.

[2] Wilkins, M. L.; 1980, *"Use of Artificial Viscosity in Multidimensional Fluid Dynamic Calculations",* J. of Comp. Physics, Vol. 36.

[3] Caramana, E. J.; Shashkov, M. J.; Whalen, P. P.; 1998, "*Formulations of Artificial Viscosity for Multi-dimensional Shock Wave Computations*", J. of Comp. Physic, Vol. 144

[4] Tatiana, G. E.; Ivan, A. S.; Salvador, M.; 2005, "*Numerical Simulation of shock-wave structure for argon and helium*", Physics of Fluids, Vol. 17

[5] Arp, V.; Persichetti, M. J.; Guo-bang, C.; 1984, "*The Grüneisen Parameter in Fluids*", Journal of Fluids Engineering, Vol. 106

[6] G. A. Bird; 1994, "*Molecular Gas Dynamics and the Direct Simulation of Gas Flows*".

[7] R. Byron, Bird; Warren, E. Stewart; Edwin, N. Lightfoot; "*Transport Phenomena, 2nd edition*".

[8] Marc, A. Meyers; "*Dynamic Behavior of Materials*".

[9] Mark, L. Wilkins; "*Computer Simulation of Dynamic Phenomena*".

[10] H. Alsmeyer; 1976, "*Density profiles in argon and nitrogen shock waves measured by the absorption of an electron beam*", J. Fluid Mech, 74, 497.

[11] Courant, R.; Friedrichs, K.; Lewy, H.; (September 1956) [1928], "*On the partial difference equations of mathematical physics*", AEC Research and Development Report, NYO-7689, translated from the German by Phyllis Fox.

[12] Anderson; Lohn David; 1995, *"Computational fluid dynamics: the basics with applications"*.

[13] Mike J. Morley; David M. Williamson; *"Shock/reload response of water and aqueous solutions of ammonium nitrate"*, AIP 1426.

[14] Philip.A.T; *"Compressible-Fluid Dynamics"*.

[15] NIST Chemistry WebBook, http://webbook.nist.gov/chemistry/.

APPENDIX - FORTRAN CODE

```
     parameter(jj=4800)
      parameter(limit = 99999999)
      real*8 T(0:jj),rho(0:jj),p(0:jj),u(0:jj),E(0:jj),eta(0:jj)
      real*8 rhohat(0:jj),uhat(0:jj),Ehat(0:jj),phat(0:jj),jhat(0:jj)
      real*8 j(0:jj),PI(0:jj),q(0:jj),H(0:jj)
      real*8 Pr, gama,mu, R, Tinf, rhoinf, cinf, lambda, etainf
      real*8 hx, ht, Ma, w
      real*8 c1, x0, M1, v1, alpha, beta
      real*8 xa, xb, xx, dx, fa, fb
      real*8 psi(0:jj),etad(0:jj)
      real*8 m0, m, k0, k, sigma0, sigma
      real*8 ratioe, Tinter, x, omegau(0:jj)
      real*8 Cv,kappa, deltax, dsa, ds, dsu(0:jj), dsT(0:jj), dsus,
dsTs
      real*8 gama0, gama00,Pref,eref,rhoref, Pinter, entropy(0:limit)
      real*8 ie(0:jj),ke(0:jj)
      real*8 Pless, rholess, Tless, eless, kaless,kappa0
      real*8 Mmass, uv, e00, e01, e02, e03, e04, s, C, e0, einter,C0
      real*8 Temperature, Cv0,CvD, gam0, Us, Up, P0
      real*8 sa, sb, fs, u0, u1
      integer i,eos, counter, bug
      character*14 filen
c     character*14 entropy
      character*1 ans

c     M-G gamma: gama0  reference pressure: Pref
c     reference internal energy: eref

c      entropy = 'en00000000.dat'
      filen = 'ns00000000.dat'

c      open(unit=23,file='zeta.dat',form='formatted',status='unknown')

c............................Choose EOS.............................
c eos=0,1,2,4, Gas. eos = 3 liquid
c eos = 0,ideal gas EOS with Gas Specific Dimensionless Scales
c eos = 1,M-G for gas, General Dimensionless Scales
c eos = 2,M-G general dimensionless, need check
c eos = 3,water M-G, general dimensionless
c eos = 4,ideal gas eos, General Dimensionless scales

       eos = 1

c............................Parameters.............................
c------------------------------Gas--------------------------
       if(eos .ne. 3) then
c       s= 1.059d0    ! Hugoniot slope, Air, JAP experiment
c       C0 = 243.d0   ! Bulk sound speed, Air, JAP experiment
c       s = 0.4086132999  ! Ma=2.5 Air solved value from maple, 3rd
order eos
```

```
c       C0 = 619.7064063  ! Ma=2.5 Air solved value from maple, 3rd
order eos
c       s = 0.3878659442  ! Ma=2.5 Air solved value from maple, 4th
order eos
c       C0 = 632.3120897  ! Ma=2.5 Air solved value from maple, 4th
order eos
c       s  = 0.9687928698 ! Ma=1.4 Air solved value from maple, 3rd
order eos
c       C0 = 293.8619384  ! Ma=1.4 Air solved value from maple, 3rd
order eos
       s = 1.0009556d0   ! Ma = 1.55, Argon
       C0 = 207.66440d0  ! Ma = 1.55, Argon
       w  = 0.81d0       ! Argon
c       w = 0.77d0        ! Air
       Ma = 1.55d0       ! Argon
c       Ma =1.4d0         ! Air
c       Pr = 0.715d0      ! Air
       Pr = 2.d0/3.d0    ! Argon
c       gama = 1.4d0      !Air
       gama = 5.d0/3.d0  ! Argon
c       mu = 1.80d-5      ! kg/m/s Air JB
c       mu = 1.983d-5     ! Kg/m/s Air viscosity online data
c       mu = 2.27d-5      ! kg/m/s Argon viscosity online data
       mu = 1.3275d-5 !kg/m/s Argon NIST data
c       kappa =  0.0252793d0 ! W/m/k Air Forced kappa
c       kappa = 0.0240d0 ! W/m/k Air heat conductivity,
c Another way is force fromideal gas kappa = gama*R*eta/(gama-1)/Pr
       kappa = 0.0163d0  ! W/m.k  Argon heat conductivity
       R      = 208.1d0 !J/kg/K Argon
c       R = 287.0d0  ! J/Kg/K Air
c       Cv = 718.d0  ! J/Kg/K Air from online data
c       Cv = 717.5d0 ! J/kg/k Air from ideal gas Eos
       Cv = R/(gama-1.d0) ! J/kg/k Air ideal gas, analytical
c       Cv = 312.2d0     !J/kg/K Argon
c       Tinf = 300.d0    ! kelvin, k
c       Pref = 103.320d3 ! Pa
       Tinf   = 164.d0   ! Helium from Muntz, same for Argon and Air
       Tref = Tinf
c       rhoinf = 1.205d0 ! Kg/m^3 Air
c       rhoinf = 1.2d0   ! kg/m^3 Air JB
       rhoinf = 1.62d0   !kg/m^3 Argon - see BSL notes
       rhoref = rhoinf
       Pref = rhoinf*R*Tinf ! Analytical Pressure from Gas EOS
       cinf   = dsqrt(gama*R*Tinf)  ! [m/s] upstream sound speed
c       C0 = cinf ! for gas with analytical sound speed
       etainf = gama**(w-0.5d0)*5.d0*sqrt(2.d0*acos(-1.d0))/16.d0
     &         *(1.d0/gama)**w  !dimensionless far field viscosity
       lambda = 3.d0*mu*sqrt(acos(-1.d0)/8.d0/R/Tinf)/rhoinf ! hard
sphere mean free path
       eta(0) = mu/(rhoinf*cinf*lambda) ! dimensionless pre-shock
viscosity
c      eta(0) = etainf*(T(0))**w
c       m0 = 28.964d-3/(6.022d23) ! kg   molecule mass of Air
       m0 = 39.948d-3/(6.022d23) ! kg   molecule mass of Argon
       k0 = 1.38066d-23           ! J/kg Boltzmann's constant
c       sigma0 = 3.617d-10        ! m Diameter Air
       sigma0 = 3.432d-10         ! m Diameter Argon
       m = m0/(rhoinf*lambda**3)         ! dimensionless m
       k = k0/(rhoinf*R*gama*lambda**3)  ! dimensionless k
       sigma = sigma0/lambda             ! dimensionless sigma
       ratioe = 122.4d0                  ! K Kelvin, Argon
c       ratioe = 97.d0                   ! K Kelvin, Air
c       Tinter = cinf**2/(gama*R*ratioe) ! Gas Dimensionless Scales
       Tinter = cinf**2/(Cv*ratioe)      ! General Dimensionless Scales
       gama0 = gama - 1.d0
c       kaless = gama*R*rhoinf*cinf*lambda ! Gas Dimensionless Scales
       kaless = Cv*rhoinf*cinf*lambda      ! General Dimensionless
Scales
       kappa0 = kappa/kaless ! dimensionless heat conductivity, General
Dimesionless scale

c....................shock velocity from Ma number................
       u0 = Ma*cinf
       u1 = Ma*(2.d0+(gama-1.d0)*Ma**2)/((gama+1.d0)*Ma**2)*cinf
       Us = u0
       Up = Us - u1
       print *,'Mach Number       Ma', Ma
       print *,'Shock velocity    Us', Us
       print *,'Partical velocity Up', Up
c       pause
c       print *, kappa0,eta(0)/((gama-1.d0)*Pr)
c       pause
c       print *,'bulk and local sound velocity',cinf,C0
c       cinf = C0
c       print *, cinf

c------------------------------Liuquid-------------------------
       elseif (eos .eq. 3) then

c--------------------------EOS  parameters---------------------
       Up = 277.8668114d0 ! m/s ! cambridge expriment, impedance match
       s = 1.92d0    ! Hugoniot slope, Meyers book P133
       C0 = 1650.d0  ! m/s, Bulk sound speed, Meyers book P133
       Us = Up*s+C0
       gama0 = 4.984362390d0  ! Gruneisen gama, impedance match from
maple

c-----------------------------properties-----------------------
c       mu = 0.798d-3    !kg/m/s viscosity, water online data
       mu = 0.85258d-3   !kg/m/s viscosity, water,NIST(300k,1g/cm^3)
c       kappa = 0.58d0   !w/m/k  heat conductivity,water
       kappa = 0.61384d0 !w/m/k heat conductivity,water,NIST(300k,
1g/cm^3)
       Cp = 4159.3d0     !J/kg/K specific heat,water,NIST(300k,1g/cm^3)
       Cv = 4105.1d0     !J/kg/K specific heat,water,NIST(300k,1g/cm^3)
       Mmass = 18.01528d0 !g-mol molar mass, water
       R  = 8.31451d3/Mmass !J/kg/K water
```

```
c-------------------------dimensionless scale------------------
       Tref   = 300.d0  !k,upstream Temperature
       Tinf = Tref
       rhoinf = 1000.d0  !kg/m^3,water
       cinf   = C0       !m/s, upstream sound speed
       Pref = 1.01325d5 !pa, upstream pressure
       lambda = ((Mmass*0.001d0)/(rhoinf*6.02d23))**(1.d0/3.d0) !space
scale, average distance of water molecular
       print *, 'lambda',lambda
       pause
       eta(0) = mu/(rhoinf*cinf*lambda) ! dimensionless viscosity
       kappa0 = kappa/(Cv*rhoinf*cinf*lambda) !dimensionless heat
conductivity
       uv = 3.1d0         !dimensionless bulk viscosity
       endif

c................initial contant viscosity coefficient.........

      do i=0,jj
      eta(i) = eta(0) ! 10**5
      enddo

c.........................dimensionless scales..................
c      cinf = C0
      Tless   = cinf**2/Cv
      Pless   = rhoinf*cinf**2
      rholess = rhoinf
      eless   = Pless
c      print *, Tless, Pless, rholess, eless,Cv
c      pause

c---------------------initial spece and time step----------------
      ht=1.0d-8
      hx=0.25d0/2.d0
      deltax = hx*1.d0

c----------------------set boundary conditions-------------------
      if(eos .eq. 0) then
      p(0)   = 1.d0/gama
c      pressure = p(0)*gama
      rho(0) = 1.d0
      u(0)   = Ma
      T(0)   = gama*p(0)/rho(0)
      j(0)  = rho(0)*u(0)
      ie(0) = p(0)/(gama - 1.d0)
      ke(0)  = rho(0)*u(0)**2/2.d0
      e(0)  = ie(0) + ke(0)
      dsus = 0.d0                          ! sum of entropy form stress
      dsu(0) = 0.d0                        ! stress entropy term
      dsTs = 0.d0                          ! sum of entropy from T
      dsT(0) = 0.d0                        ! T entropy term

      omegau(0) = 1.16145d0/((Tinter*T(0))**0.14874d0)    ! Chapman-
Enskog
```

```
     &          + 0.52487d0/(exp(0.77320d0*Tinter*T(0)))
     &          + 2.16178d0/(exp(2.43787d0*Tinter*T(0)))
      eta(0) = 5.d0*dsqrt(m*k*T(0)/acos(-1.d0))
     &        /(16.d0*omegau(0)*sigma**2)

c      eta(0) = 2.d0*dsqrt(m*k*T(0)/acos(-1.d0))
c     &        /(3.d0*acos(-1.d0)*sigma**2)               ! Maxwell

c      x = T(0)*Tinf/vari0
c      omegau = -0.0799d0*x**5 + 0.743d0*x**4 - 2.7973d0*x**3
c     &       + 5.485d0*x**2
c     &       -5.9588d0*x + 4.1996d0
c      etad(0) = 5.d0*dsqrt(acos(-1.0)*m*k*T(0))/
c     &         ((16.d0*acos(-1.0)*sigma**2*omegau)*(rhoinf*cinf*lambda))
c      eta(0) = etad(0)/(rhoinf*cinf*lambda)

c       eta(0) = etainf*(T(0))**w                        ! Power Law
c
c      p(jj) = 4.5000*p(0)
c      T(jj) = 1.6875*T(0)
c      rho(jj) = 2.667*rho(0)
c      u(jj)   = 0.5774
c      j(jj)  = rho(jj)*u(jj)
c      e(jj)  = rho(jj)*u(jj)*u(jj)/2 + p(jj)/(gama-1)

       rho(jj) = rho(0)*(gama+1.d0)*Ma**2/((gama-1.d0)*Ma**2+2.d0)
       u(jj)  = rho(0)*u(0)/rho(jj)
       p(jj)  = rho(0)*u(0)**2 + p(0) - rho(jj)*u(jj)**2
       ie(jj) = (ie(0)+u(0)**2/2.d0+p(0)/rho(0)-u(jj)**2/2.d0
     &         -p(jj)/rho(jj))*rho(jj)
       print *, 'rankine', u(jj),p(jj),ie(jj)
       u(jj)   = u(0)*(2.d0+(gama-1.d0)*Ma**2)/((gama+1.d0)*Ma**2)
       p(jj)   = p(0)*(2.d0*gama*Ma**2-gama+1.d0)/(gama+1.d0)
       e(jj)   = rho(jj)*u(jj)*u(jj)/2.d0 + p(jj)/(gama-1.d0)
       ie(jj)  = p(jj)/(gama - 1.d0)
       print *, 'eos', u(jj),p(jj),ie(jj)
       pause
       T(jj)   = gama*p(jj)/rho(jj)
       dsu(jj) = 0.d0                          ! stress entropy term
       dsT(jj) = 0.d0                          ! T entropy term

c       eta(jj) = etainf*(T(jj))**w                       ! power law

c       eta(jj) = 2.d0*dsqrt(m*k*T(jj)/acos(-1.d0))
c     &          /(3.d0*acos(-1.d0)*sigma**2)             ! Maxwell

      omegau(jj) = 1.16145d0/((Tinter*T(jj))**0.14874d0)  ! Chapman-
Enskog
     &           + 0.52487d0/(exp(0.77320d0*Tinter*T(jj)))
     &           + 2.16178d0/(exp(2.43787d0*Tinter*T(jj)))
      eta(jj) = 5.d0*dsqrt(m*k*T(jj)/acos(-1.d0))
     &          /(16.d0*omegau(jj)*sigma**2)

c       x = T(jj)*Tinf/vari0
```

```
c       omegau = -0.0799d0*x**5 + 0.743d0*x**4 - 2.7973d0*x**3
c     &        + 5.485d0*x**2
c     &        -5.9588d0*x + 4.1996d0
c       etad(jj) = 5.d0*dsqrt(acos(-1.0)*m*k*T(jj))/
c     &         ((16.d0*acos(-1.0)*sigma**2*omegau)*(rhoinf*cinf*lambda))
c       eta(jj) = etad(jj)/(rhoinf*cinf*lambda)

c       Cv0 = ie(jj)/rho(jj)/T(jj)
c       CvD = Cv0*gama*R
c       print *,Pr, gama, mu, R, Tinf, rhoinf, cinf,lambda,eta(0)
c       print *, kappa0,eta(0)/((gama-1.d0)*Pr)
c       pause

      elseif(eos .eq. 4) then
c      pressure = p(0)*gama
      rho(0) = 1.d0
      u(0)   = Ma
      T(0)   = 1.d0/(gama*(gama-1.d0))
      p(0)  = rho(0)*(gama-1.d0)*T(0)
      j(0)  = rho(0)*u(0)
      ie(0) = p(0)/(gama - 1.d0)
      ke(0)  = rho(0)*u(0)**2/2.d0
      e(0)  = ie(0) + ke(0)
      dsus = 0.d0                          ! sum of entropy form stress
      dsu(0) = 0.d0                        ! stress entropy term
      dsTs = 0.d0                          ! sum of entropy from T
      dsT(0) = 0.d0                        ! T entropy term
c      omegau(0) = 1.16145d0/((Tinter*T(0))**0.14874d0)    ! Chapman-
Enskog
c     &          + 0.52487d0/(exp(0.77320d0*Tinter*T(0)))
c     &          + 2.16178d0/(exp(2.43787d0*Tinter*T(0)))
c      eta(0) = 5.d0*dsqrt(m*k*T(0)/acos(-1.d0))
c     &        /(16.d0*omegau(0)*sigma**2)

c      eta(0) = 2.d0*dsqrt(m*k*T(0)/acos(-1.d0))
c     &        /(3.d0*acos(-1.d0)*sigma**2)                ! Maxwell

c      x = T(0)*Tinf/vari0
c      omegau = -0.0799d0*x**5 + 0.743d0*x**4 - 2.7973d0*x**3
c     &       + 5.485d0*x**2
c     &       -5.9588d0*x + 4.1996d0
c      etad(0) = 5.d0*dsqrt(acos(-1.0)*m*k*T(0))/
c     &         ((16.d0*acos(-1.0)*sigma**2*omegau)*(rhoinf*cinf*lambda))
c      eta(0) = etad(0)/(rhoinf*cinf*lambda)

c       eta(0) = etainf*(T(0))**w                          ! Power Law
c
c      p(jj) = 4.5000*p(0)
c      T(jj) = 1.6875*T(0)
c      rho(jj) = 2.667*rho(0)
c      u(jj)   = 0.5774
c      j(jj)  = rho(jj)*u(jj)
c      e(jj)  = rho(jj)*u(jj)*u(jj)/2 + p(jj)/(gama-1)
```

```
       rho(jj) = rho(0)*(gama+1.d0)*Ma**2/((gama-1.d0)*Ma**2+2.d0)
       u(jj)  = rho(0)*u(0)/rho(jj)
       p(jj)  = rho(0)*u(0)**2 + p(0) - rho(jj)*u(jj)**2
       ie(jj) = (ie(0)+u(0)**2/2.d0+p(0)/rho(0)-u(jj)**2/2.d0
     &         -p(jj)/rho(jj))*rho(jj)
       print *, 'rankine', u(jj),p(jj),ie(jj)
       u(jj)   = u(0)*(2.d0+(gama-1.d0)*Ma**2)/((gama+1.d0)*Ma**2)
       p(jj)   = p(0)*(2.d0*gama*Ma**2-gama+1.d0)/(gama+1.d0)
       e(jj)   = rho(jj)*u(jj)*u(jj)/2.d0 + p(jj)/(gama-1.d0)
       ie(jj)  = p(jj)/(gama - 1.d0)
       print *, 'eos', u(jj),p(jj),ie(jj)
c       pause
       T(jj)   = p(jj)/(rho(jj)*(gama-1.d0))
       dsu(jj) = 0.d0                         ! stress entropy term
       dsT(jj) = 0.d0                         ! T entropy term
c       pause
c       eta(jj) = etainf*(T(jj))**w                        ! power law

c       eta(jj) = 2.d0*dsqrt(m*k*T(jj)/acos(-1.d0))
c     &          /(3.d0*acos(-1.d0)*sigma**2)              ! Maxwell

c      omegau(jj) = 1.16145d0/((Tinter*T(jj))**0.14874d0)  ! Chapman-
Enskog
c     &           + 0.52487d0/(exp(0.77320d0*Tinter*T(jj)))
c     &           + 2.16178d0/(exp(2.43787d0*Tinter*T(jj)))
c      eta(jj) = 5.d0*dsqrt(m*k*T(jj)/acos(-1.d0))
c     &          /(16.d0*omegau(jj)*sigma**2)

c       x = T(jj)*Tinf/vari0
c       omegau = -0.0799d0*x**5 + 0.743d0*x**4 - 2.7973d0*x**3
c     &        + 5.485d0*x**2
c     &        -5.9588d0*x + 4.1996d0
c       etad(jj) = 5.d0*dsqrt(acos(-1.0)*m*k*T(jj))/
c     &         ((16.d0*acos(-1.0)*sigma**2*omegau)*(rhoinf*cinf*lambda))
c       eta(jj) = etad(jj)/(rhoinf*cinf*lambda)

c       Cv0 = ie(jj)/rho(jj)/T(jj)
c       CvD = Cv0*gama*R
c       print *,Pr, gama, mu, R, Tinf, rhoinf, cinf,lambda,eta(0)
c       print *, kappa0,eta(0)/((gama-1.d0)*Pr)
c       pause

      elseif(eos .eq. 1) then ! M-G,AIR
c..................coefficients of epsilon in M-G................
      e00 = Cv*Tref/(cinf**2) ! positive or negative? question
      e01 = gama0*e00
      e02 = (C0**2/(cinf**2)+gama0**2*e00)/2.d0
      e03 = (4.d0*s*C0**2/(cinf**2)+gama0**3*e00)/6.d0
      e04 = ((18.d0*s**2*C0**2-2.d0*gama0*s*C0**2)/(cinf**2)
     &     +gama0**4*e00)/24.d0

c............................left BC.............................
      x = 0.d0    ! String is zero, pre-shock
      p(0)   = Pref/Pless
      print *, 'Pressure Direct',p(0),p(0)*Pless
      rho(0) = rhoref/rholess
c       Us = Up*s+C0
      u(0)   = Us/cinf
      T(0)   = Tref/Tless
      j(0)   = rho(0)*u(0)
      ke(0)  = rho(0)*u(0)**2/2.d0
      ie(0)  = Pref/gama0/Pless  ! Mass internal energy
      p(0)    = rho(0)*(e01 + 2.d0*e02*x + 3.d0*e03*x**2 + 4.d0*e04*x**3)
      print *, 'Pressure Poly  ',p(0),p(0)*Pless
      p(0) = (rhoinf*C0**2
     &       *(x + (2.d0*s - gama0/2.d0)*x**2 + s*(3.d0*s - gama0)*x**3))
     &      /(rhoinf*cinf**2) + gama0*rho(0)*(ie(0)/rho(0))
      print *, 'Pressure M-G   ',p(0),p(0)*Pless
      e(0)  = ke(0)+ie(0)
c       eta(0) = etainf*(T(0))**w                        ! Power Law
      print *,'Left B.C. Dimensionless', rho(0),u(0),p(0),T(0),
     &                                   ie(0)/rho(0)
      print *,'Left B.C. Dimensional  ', rho(0)*rhoinf,u(0)*cinf,
     &                                   p(0)*Pless,T(0)*Tless,
     &                                   ie(0)*eless/rhoinf

      dsus   = 0.d0             ! sum of entropy form stress
      dsu(0) = 0.d0             ! stress entropy term
      dsTs   = 0.d0             ! sum of entropy from T
      dsT(0) = 0.d0             ! Temperature entropy term

c..............................Right B.C..........................
c       rho(jj) = rho(0)*(gama+1.d0)*Ma**2/((gama-1.d0)*Ma**2+2.d0)
c       x = 1.d0 - rho(0)/rho(jj)
      u(jj) = (Us-Up)/cinf
cx       u(jj) = Us-Up
c       u(jj)   = u(0)*(2.d0+(gama0)*Ma**2)/((gama0+2.d0)*Ma**2)
c       u(jj) = rho(0)*u(0)/rho(jj) ! rankine hugoniot jump equations
for stationary Euler shock wave
      rho(jj) = rho(0)*u(0)/u(jj)
cx       rho(jj) = rhoinf*Us/u(jj)
      x = 1.d0 - rho(0)/rho(jj)
cx      x = 1.d0 - rhoinf/rho(jj)
      p(jj) = rho(0)*u(0)**2 + p(0) - rho(jj)*u(jj)**2
cx       p(jj) = rhoinf*Us**2 + Pref - rho(jj)*u(jj)**2
      print *,'Pressure Hugoniot',p(jj),p(jj)*Pless
cx       print *, 'pressure hugoniot', p(jj)
      ie(jj) = (rho(0)*u(0)*(ie(0)/rho(0) + u(0)**2/2.d0 + p(0)/rho(0))
     &        /(rho(jj)*u(jj)) - u(jj)**2/2.d0 - p(jj)/rho(jj))
     &        *rho(jj)           ! Mass internal energy
      print *,'Internal Energy Hugoniot',ie(jj),
     &        ie(jj)*eless,ie(jj)*eless/(rho(jj)*rhoinf)
cx      ie(jj) = (Cv*Tref/rhoinf+Us**2/2.d0 + Pref/rhoinf
cx     &        - u(jj)**2/2.d0 - p(jj)/rho(jj))
cx     &        *rho(jj)           ! Mass internal energy

c       ie(jj) = (p(jj) - (rho(0)*C0**2
c      &      *(x+(2.d0*s - gama0/2.d0)*x**2+s*(3.d0*s-gama0)*x**3))
c     &      /(rhoinf*cinf**2))/gama0/rho(0)
c      ie(jj) = ie(jj)*rho(jj)
c      print *,'internal energy eos', ie(jj)
      gam0 = 2.d0*(-p(jj) + C0**2/cinf**2*x + 2.d0*x**2*s*C0**2/cinf**2
     &       + 3.d0*s**2*x**3*C0**2/cinf**2)
     &     /(x**2*C0**2/cinf**2 + 2.d0*x**3*s*C0**2/cinf**2
     &       - 2.d0*ie(jj)/rho(jj))
c      gam0 = 2.d0*(p(jj) - x-2.d0*x**2*s-3.d0*s**2*x**3)
c     &      /(-x**2-2.d0*x**3*s+2.d0*rho(0)*ie(jj)/rho(jj))
      print *,'M-G Gamma Diff is/was',gam0,gama0
c      pause
      gama0 = gam0
      print *,'Values to find pressure'
      print *, 'rho(0)',rho(0)*rhoinf,'C0',
C0,'x',x,'s',s,'gama0',gama0
      print *, 'rhoinf', rhoinf,'cinf', cinf,
     &         'ie(jj)',ie(jj)*eless/(rho(jj)*rhoinf),
     &          'rho(jj)',rho(jj)*rhoinf, 'Pless', Pless, 'eless', eless,
     &         'rhoinf*cinf**2', rhoinf*cinf**2
c       p(jj) = -
gama0**5*x**4*e00/(24.d0*cinf**2)+gama0**2*x**4*s/12.d0
c     &       - 3.d0*gama0*x**4*s**2/4.d0
c      &       + (x+(2.d0*s - gama0/2.d0)*x**2 + s*(3.d0*s - gama0)*x**3)
c     &       + gama0*rho(0)*ie(jj)/rho(jj) ! 4th order eos
      p(jj) = (x + (2.d0*s - gama0/2.d0)*x**2 + s*(3.d0*s - gama0)*x**3)
     &        *C0**2/cinf**2 + gama0*ie(jj)/rho(jj)      ! 3rd order
eos
      print *, 'Prssure M-G       ',p(jj),p(jj)*Pless
c      p(jj) = rhoinf*C0**2
c     &       *(x + (2.d0*s - gama0/2.d0)*x**2 + s*(3.d0*s -
gama0)*x**3)
c     &       + gama0*ie(jj)*eless/rho(jj)
c      print *, 'prssure M-G DN     ',p(jj)/Pless,p(jj)
c      print *, rho(0), rhoinf
      ke(jj) = rho(jj)*u(jj)**2/2.d0
      e(jj) = ie(jj) + ke(jj)
      T(jj) = T(0) + ie(jj)/rho(jj) - ie(0)/rho(0)
      print *,'Temperature       ',T(jj),T(jj)*Tless
      e0 = e00+e01*x+e02*x**2+e03*x**3+e04*x**4
      P0 = rho(0)*(e01+2.d0*e02*x+3.d0*e03*x**2+4.d0*e04*x**3)
      print *, 'P(v)         ', P0-gama0*rho(0)*e0
      print *, 'P(T)         ', gama0*rho(0)*ie(jj)/rho(jj)
      print *, 'P with Strain', x+(2.d0*s-gama0/2.d0)*x**2
     &         +s*(3.d0*s-gama0)*x**3
      print *, 'P with Temp  ', gama0*rho(0)*ie(jj)/rho(jj)


c      print *,'p0 and p(0)', p0, p(0)
c      e0 = e00+e01*x+e02*x**2+e03*x**3+e04*x**4
c      print *,'e0 and e(0)', e0, e(0)
c      p(jj) = p0 + gama0*rho(0)*(ie(jj)/rho(jj) - e0)
c      print *, 'prssure changed e0', p(jj), p(jj)*Pless
      print *, 'Right B.C. Dimensionless ',rho(jj),u(jj),p(jj),T(jj),
     &                                     ie(jj)
```

```
      print *, 'Right B.C. Dimensional   ',rho(jj)*rhoinf,u(jj)*cinf,
     &                                     p(jj)*Pless,T(jj)*Tless,
     &                                     e(jj)*eless/(rho(jj)*rhoinf)
c      pause

c      eta(jj) = etainf*(T(jj))**w ! power law

      dsu(jj) = 0.d0                                ! stress entropy
term
      dsT(jj) = 0.d0                                ! T entropy term

      elseif(eos .eq. 2) then
      T(0) = Tinf/Tless
      p(0)   = (gama - 1.d0)*T(0)
      rho(0) = 1.d0
      u(0)   = Ma
c      T(0)   = 1.d0
      j(0)  = rho(0)*u(0)
      e(0)  = rho(0)*u(0)*u(0)/2.d0 + p(0)/(gama-1.d0)
      dsus = 0.d0                          ! sum of entropy form stress
      dsu(0) = 0.d0                        ! stress entropy term
      dsTs = 0.d0                          ! sum of entropy from T
c      omegau(0) =
1.16145/((Tinter*T(0))**0.14874)                     ! Chapman-Enskog
c     &          + 0.52487/(exp(0.77320*Tinter*T(0)))
c     &          + 2.16178/(exp(2.43787*Tinter*T(0)))
c      eta(0) = 5.0*sqrt(m*k*T(0)/acos(-1.0))/(16.0*omegau(0)*sigma**2)
c      eta(0) = 2.0*sqrt(m*k*T(0)/acos(-1.0))/(3.0*acos(-
1.0)*sigma**2) ! Maxwell

c      x = T(0)*Tinf/vari0
c      omegau = -0.0799*x**5 + 0.743*x**4 - 2.7973*x**3 + 5.485*x**2
c     &         -5.9588*x + 4.1996
c      eta(0) = 5.0*sqrt(acos(-1.0)*m*k*T(0))/
c     &        ((16.0*acos(-1.0)*sigma**2*omegau)*(rhoinf*cinf*lambda))
c      eta(0) = etad(0)/(rhoinf*cinf*lambda)
c       eta(0) = etainf*(T(0))**w                     ! Power Law
c
c      p(jj) = 4.5000*p(0)
c      T(jj) = 1.6875*T(0)
c      rho(jj) = 2.667*rho(0)
c      u(jj)   = 0.5774
c      j(jj)  = rho(jj)*u(jj)
c      e(jj)  = rho(jj)*u(jj)*u(jj)/2 + p(jj)/(gama-1)

       rho(jj) = rho(0)*(gama+1.d0)*Ma**2/((gama-1.d0)*Ma**2+2.d0)
       u(jj)   = u(0)*(2.d0+(gama-1.d0)*Ma**2)/((gama+1.d0)*Ma**2)
       p(jj)   = p(0)*(2.d0*gama*Ma**2-gama+1.d0)/(gama+1.d0)
       e(jj)   = rho(jj)*u(jj)*u(jj)/2.d0 + p(jj)/(gama-1.d0)
       ie(jj)  = p(jj)/(gama - 1.d0)
       T(jj)   = p(jj)/(gama - 1.d0)/rho(jj)
       dsu(jj) = 0.d0                        ! stress entropy term
       dsT(jj) = 0.d0                        ! T entropy term
c       Cv0 = ie(jj)/rho(jj)/T(jj)
```

```
c       CvD = Cv0*gama*R

c      omegau(jj) =1.16145/((Tinter*T(jj))**0.14874)   ! Chapman-Enskog
c     &           + 0.52487/(exp(0.77320*Tinter*T(jj)))
c     &           + 2.16178/(exp(2.43787*Tinter*T(jj)))
c      eta(jj) = 5.0*sqrt(m*k*T(jj)/acos(-1.0))
c     &          /(16.0*omegau(jj)*sigma**2)
c      eta(jj) = 2.0*sqrt(m*k*T(jj)/acos(-1.0))/(3.0*acos(-
1.0)*sigma**2)  ! Maxwell
c      eta(jj) = eta(0)
c      eta(jj) = etainf*(T(jj))**w

c-------------------------------liquid---------------------------
      elseif(eos .eq. 3) then    !M-G LIQUID
      print *, 'Us = ', Us, 'Up = ', Up

c..............................LHS BC.....................
      u(0) = Us/cinf
      rho(0) = rhoinf/rhoinf
      p(0) = Pref/Pless
      print *, 'pressure direct', p(0),p(0)*Pless
      T(0) = Tref/Tless
      ke(0) = rho(0)*u(0)*u(0)/2.d0
      ie(0) = p(0)/gama0
      e(0) = ke(0)+ ie(0)
      print *,'left HS DL', rho(0),u(0), p(0), T(0), e(0)
      print *,'left HS DN', rho(0)*rhoinf, u(0)*cinf, p(0)*Pless,
     &        T(0)*Tless, e(0)*eless

      dsus = 0.d0             ! sum of entropy form stress
      dsu(0) = 0.d0           ! stress entropy term
      dsTs = 0.d0             ! sum of entropy from T
      dsT(0) = 0.d0           ! T entropy term

c..............................RHS BC.....................
      u(jj) = (Us-Up)/cinf
      rho(jj) = u(0)*rho(0)/u(jj)
      x = 1.d0 - rho(0)/rho(jj)
      p(jj) = p(0) + rho(0)*u(0)**2 - rho(jj)*u(jj)**2
      print *,'pressure hugoniot',p(jj),p(jj)*Pless
      ie(jj) =(rho(0)*u(0)*(ie(0)/rho(0)+u(0)**2/2.d0 + p(0)/rho(0))
     &         /(rho(jj)*u(jj)) - u(jj)**2/2.d0 - p(jj)/rho(jj))
     &         *rho(jj)
      print *,'internal energy hugoniot',ie(jj),
     &        ie(jj)*eless, ie(jj)*eless/(rho(jj)*rhoinf)
      ke(jj) = rho(jj)*u(jj)**2/2.d0
      e(jj) = ie(jj) + ke(jj)
      gam0 = 2.d0*(p(jj)-x-2.d0*x**2*s-3.d0*s**2*x**3)
     &      /(-x**2-2.d0*x**3*s+2.d0*rho(0)*ie(jj)/rho(jj))
      print *,'Gamma diff is/was', gam0, gama0
      gama0 = gam0
      print *,'values to find pressure'
      print *, 'rho(0)',rho(0)*rhoinf,'C0',
C0,'x',x,'s',s,'gama0',gama0
```

```
      print *, 'rhoinf', rhoinf,'cinf', cinf,
     &         'ie(jj)',ie(jj)*eless/(rho(jj)*rhoinf),
     &          'rho(jj)',rho(jj)*rhoinf, 'Pless', Pless, 'eless', eless,
     &         'rhoinf*cinf**2', rhoinf*cinf**2
c       p(jj) = -
gama0**5*x**4*e00/(24.d0*cinf**2)+gama0**2*x**4*s/12.d0
c     &       - 3.d0*gama0*x**4*s**2/4.d0
c     &       + (x+(2.d0*s - gama0/2.d0)*x**2 + s*(3.d0*s - gama0)*x**3)
c     &      + gama0*rho(0)*ie(jj)/rho(jj) ! 4th order eos
      p(jj) = (x + (2.d0*s - gama0/2.d0)*x**2 + s*(3.d0*s - gama0)*x**3)
     &      + gama0*rho(0)*ie(jj)/rho(jj) ! 3rd order eos
      print *, 'prssure M-G DL     ',p(jj), p(jj)*Pless
      T(jj) = T(0) + ie(jj)/rho(jj) - ie(0)/rho(0)
      print *,'Temp DL', T(jj), T(jj)*Tless
      print *, 'right HS DL', rho(jj),u(jj),p(jj),T(jj),e(jj)
      print *, 'right HS DN', rho(jj)*rhoinf,u(jj)*cinf,p(jj)*Pless,
     &                       T(jj)*Tless,e(jj)*eless

c      pause

      dsu(jj) = 0.d0                            ! stress entropy term
      dsT(jj) = 0.d0                            ! T entropy term

      endif
c-----------------------printout jump condition------------------

      print *,'Jump Conditions'
c     Print *,'dimensional Tem',T(0)*cinf**2/Cv
c      Print *, T(jj)*cinf**2/Cv, T(jj)
c      Print *,'Tem and InterE', T(0),ie(0)/rho(0),T(jj),ie(jj)/rho(jj)
      Print *,'Velocity    ratio     ',u(jj)/u(0)
      print *,'Temperature ratio    ',T(jj)/T(0)
      print *,'Density     ratio    ',rho(jj)/rho(0)
      Print *,'Pressure    ratio    ',p(jj)/p(0)
      print *,'Velocity Dimensional ',u(0)*cinf,u(jj)*cinf
      Pause

c.........update heat conductivity to calculate entropy (GAS Only)....
      if (eos .ne. 3) then
      kappa0 = gama*R*eta(0)/((gama-1.d0)*Pr*Cv)
      kappa = kappa0*(Cv*rhoinf*cinf*lambda)
      endif

c-------------------------theoretical entropy-------------------

c     e(jj) =(rho(0)*u(0)*(ie(0)/rho(0)+u(0)** print *,log(2.d0)
      dsa = Cv*dlog(T(jj)/T(0))+ R*dlog(rho(0)/rho(jj))  ! J/kg.k
theoretical Entropy
c      print *,Cv,Cv*dlog(T(jj)/T(0))
c      print *,R,R*dlog(rho(0)/rho(jj))
c      print *,dsa
c      pause

c       omegau = 1.16145/(vari0*T(jj))**0.14874
```

```
c     &       + 0.52487/(exp(0.77320*vari0*T(jj)))
c     &       + 2.16178/(exp(2.43787*vari0*T(jj)))
c       x = T(jj)*Tinf/vari0
c       omegau = -0.0799*x**5 + 0.743*x**4 - 2.7973*x**3 + 5.485*x**2
c     &         -5.9588*x + 4.1996
c      eta(jj) = 5.0*sqrt(acos(-1.0)*m*k*T(jj))/
c     &        ((16.0*acos(-1.0)*sigma**2*omegau)*(rhoinf*cinf*lambda))
c       etad(jj) = 5.0*sqrt(acos(-1.0)*m*k*T(jj))/
c     &         (16.0*acos(-1.0)*sigma**2*omegau)
c      eta(jj) = etad(jj)/(rhoinf*cinf*lambda)
c       eta(jj) = etainf*(T(jj))**w


c--------------------------set initial condidion----------------------
c..........................Left Half..........................
      do i = 1,jj/2
        rho(i) = rho(0)
        T(i)   = T(0)
        u(i)   = u(0)
        p(i)   = p(0)
        j(i)   = u(i)*rho(i)
        e(i)   = e(0)
        eta(i) = eta(0)
       ! write (*,'(i4,f12.5)') i,T(i)
      enddo

c..........................Right Half.........................
      do i=jj/2,jj-1
        rho(i) = rho(jj)
        T(i)   = T(jj)
        u(i)   = u(jj)
        p(i)   = p(jj)
        j(i)   = rho(i)*u(i)
        e(i)   = e(jj)
        eta(i) = eta(jj)
       ! write (*,'(i4,f12.5)') i,t(i)
      enddo
c

      print *,'RH',p(jj)/p(0),rho(jj)/rho(0),u(jj)/u(0),T(jj)/T(0)
      print *,eta(0)
c      pause
c
      do i=0,jj
      write(*,'(A2,i5,10f10.3)')
     &  'IC',i,rho(i),u(i),p(i),t(i),e(i),p(i)
      enddo

c-----------------------------------------------------------------------
c read restart file
c-----------------------------------------------------------------------
      nstep = 0
      ans='n'
      if (ans .eq. 'y' .or. ans .eq. 'Y') then
```

```
c      read in old file
       open (unit=44, file='restart.dat',form='formatted',
     &  status='old',err=135)
        do i = 0,jj,1 !Zero out variables
        read(44,'(i15,6e25.18)')  nstep,rho(i),U(i),P(i),e(i),T(i),eta(i)
c      write(*,'(6e25.18)') rho(i),U(i),P(i),e(i),T(i),eta(i)
       rhohat(i) = rho(i)
       uhat(i) =   u(i)
       phat(i) =   p(i)
       ehat(i) =   e(i)
       enddo
       print *,'ShockLeft:',rho(0),u(0),p(0),e(0),T(0)
       print *,'ShockRigh: ',rho(jj),u(jj),p(jj),e(jj),T(jj)
       print *,'ShockJump: ',
     & rho(jj)/rho(0),u(jj)/u(0),p(jj)/p(0),e(jj)/e(0),t(jj)/t(0)
       print *,'*'
       goto 137
 135   Print *,'No *restart file...'
 137   close(44)
      if (1 .eq. 1) then ! average out a restart
      do i=0,jj-2,2
        rho(i+1) = (rho(i+2) + rho(i))/2.d0
        u(i+1)   = (u(i+2)   + u(i))/2.d0
        p(i+1)   = (p(i+2)   + p(i))/2.d0
        e(i+1)   = (e(i+2)   + e(i))/2.d0
        eta(i+1) = (eta(i+2) + eta(i))/2.d0
        enddo
c       flag = .false.
        endif
       goto  456
       endif


c---------------------- state time steps  -----------------------
 456  do n = nstep+1,nstep+1000000  !time
c------------------------shock move---------------------------
      if (eos .eq. 0) then ! ideal gas EOS
      umax = 0.
      do i = 0,jj
      uloc = abs(u(i))+dsqrt(T(i))
      if (uloc .gt. umax) umax=uloc
      enddo
      ht = 0.001*hx/uloc
c      print *,'ht=',ht
c      print *,'integer time ',n
      do i= 0,jj-2,2    !space
c calculate average
      j(i+1) =    (rho(i+2)*u(i+2) + rho(i)*u(i))/2.d0
      PI(i+1) = 4.d0*(eta(i+2)+ eta(i))*(u(i+2)-u(i))/hx/6.d0
      q(i+1) = -1.d0*(eta(i+2)+ eta(i))*(T(i+2)-T(i))/hx
     &        /((gama-1.d0)*Pr)/2.d0
c     T(i+1) = gama*p(i+1)/rho(i+1)
      E(i) = rho(i)*u(i)*u(i)/2.d0+p(i)/(gama-1.d0)
      rho(i+1) = (rho(i+2)+ rho(i))/2.d0
      u(i+1) = (u(i+2)+u(i))/2.d0
      p(i+1) = (p(i+2)+ p(i))/2.d0
      E(i+1) = (E(i+2)+ E(i))/2.d0
      H(i+1) = (E(i+1)+ p(i+1))/rho(i+1)
c
      enddo
c      do i=0,jj
c      write(*,'(i4,10f10.3)')
c     &  i,j(i),PI(i),q(i),T(i),E(i),rho(i),u(i),p(i),e(i),h(i)
c      enddo
c
c      print *,'1',n,rho(24)
      do i=2,jj-2,2
c
      q(i) = (q(i-1)+q(i+1))/2.d0
      rhohat(i)=rho(i)-ht*(j(i+1)-j(i-1))/hx
      jhat(i) = rho(i)*u(i) + ht*(  (PI(i+1)    -  PI(i-1))
     &                          -(j(i+1)*u(i+1) - j(i-1)*u(i-1))
     &                                 -(p(i+1) - p(i-1)))/hx
      Ehat(i) = E(i)+ ht*(((PI(i+1)*u(i+1)) - PI(i-1)*u(i-1))-
     &                     (j(i+1)*H(i+1)   -  j(i-1)*H(i-1))-
     &                            (q(i+1)   -   q(i-1)))/hx
      uhat(i) = jhat(i)/rhohat(i)
      phat(i)= (gama -1.d0)*(Ehat(i)-rhohat(i)*uhat(i)*uhat(i)/2.d0)
c

c
      enddo
c      do i=0,jj
c      write(*,'(i4,10f10.3)')
c     &  i,rhohat(i),jhat(i),ehat(i),phat(i),uhat(i)
c      enddo
c
c      print *,rho(24),ht,j(28+1),j(28-1),hx
c      print *,'2',n,rhohat(28)

      do i=2,jj-2,2
      rho(i) = rhohat(i)
      u(i) = uhat(i)
      j(i) = jhat(i)
      E(i) = Ehat(i)
      p(i) = phat(i)
      T(i) = gama*p(i)/rho(i)
      omegau(i) = 1.16145d0/((Tinter*T(i))**0.14874d0) ! Chapman-Enskog
     &           + 0.52487d0/(exp(0.77320d0*Tinter*T(i)))
     &           + 2.16178d0/(exp(2.43787d0*Tinter*T(i)))
      eta(i) = 5.d0*dsqrt(m*k*T(i)/acos(-1.d0))
     &        /(16.d0*omegau(i)*sigma**2)

c      eta(i) = 2.d0*dsqrt(m*k*T(i)/acos(-1.d0))
c     &        /(3.d0*acos(-1.d0)*sigma**2)               ! Maxwell

c      omegau = 1.16145/(vari0*T(i))**0.14874
c     &       + 0.52487/(exp(0.77320*vari0*T(i)))
```

```
c     &       + 2.16178/(exp(2.43787*vari0*T(i)))
c      eta(i) = 5.0*sqrt(acos(-1.0)*m*k*T(i))/
c     &        ((16.0*acos(-1.0)*sigma**2*omegau)*(rhoinf*cinf*lambda))

c      x = T(i)*Tinf/vari0
c      omegau = -0.0799*x**5 + 0.743*x**4 - 2.7973*x**3 + 5.485*x**2
c     &         -5.9588*x + 4.1996
c      etad(i) = 5.d0*dsqrt(acos(-1.0)*m*k*T(i))/
c     &         ((16.d0*acos(-1.0)*sigma**2*omegau)*(rhoinf*cinf*lambda))
c      eta(i) = etad(i)/(rhoinf*cinf*lambda)


c      etad(i) = 5.0*sqrt(acos(-1.0)*m*k*T(i))/
c     &         (16.0*acos(-1.0)*sigma**2*omegau)
c      eta(i) = etad(i)/(rhoinf*cinf*lambda)

c      eta(i) = etainf*(T(i))**w                          ! Power Law
      enddo
c
c      print *,'3',n,rho(28)

c      do i=0,jj-2,2
c      rho(i+1) = (rho(i+2)+rho(i))/2.0
c      u(i+1) = (u(i+2)+u(i))/2.0
c      p(i+1) = (p(i+2)+ p(i))/2.000
c      E(i+1) = (E(i+2)+ E(i))/2.0
c      H(i+1) = (E(i+1)+ p(i+1))/rho(i+1)
c      enddo


c      do i=0,jj
c      write(*,'(A1,i4,10f10.3)')
c     &  'e',i,rho(i),j(i),e(i),p(i),u(i)
c      enddo
c
c      print *,'4',n,rho(28)

      elseif (eos .eq. 4) then ! ideal gas EOS with general Dless
scales
      umax = 0.
      do i = 0,jj
      uloc = abs(u(i))+dsqrt(gama*(gama-1.d0)*T(i))
      if (uloc .gt. umax) umax=uloc
      enddo
      ht = 0.001*hx/uloc
c      print *,'ht=',ht
c      print *,'integer time ',n
       do i= 0,jj-2,2    !space
c calculate average
      j(i+1) =    (rho(i+2)*u(i+2) + rho(i)*u(i))/2.d0
      PI(i+1) = 4.d0*(eta(i+2)+ eta(i))*(u(i+2)-u(i))/hx/6.d0
      q(i+1) = -1.d0*gama*(eta(i+2)+ eta(i))*(T(i+2)-T(i))/hx
     &        /(Pr)/2.d0
c     T(i+1) = gama*p(i+1)/rho(i+1)
```

```
      E(i) = rho(i)*u(i)*u(i)/2.d0+p(i)/(gama-1.d0)
      rho(i+1) = (rho(i+2)+ rho(i))/2.d0
      u(i+1) = (u(i+2)+u(i))/2.d0
      p(i+1) = (p(i+2)+ p(i))/2.d0
      E(i+1) = (E(i+2)+ E(i))/2.d0
      H(i+1) = (E(i+1)+ p(i+1))/rho(i+1)
c
      enddo
c       do i=0,jj
c       write(*,'(i4,10f10.3)')
c     &  i,j(i),PI(i),q(i),T(i),E(i),rho(i),u(i),p(i),e(i),h(i)
c       enddo
c
c       print *,'1',n,rho(24)
      do i=2,jj-2,2
c
      q(i) = (q(i-1)+q(i+1))/2.d0
      rhohat(i)=rho(i)-ht*(j(i+1)-j(i-1))/hx
      jhat(i) = rho(i)*u(i) + ht*(  (PI(i+1)    -  PI(i-1))
     &                          -(j(i+1)*u(i+1) - j(i-1)*u(i-1))
     &                                 -(p(i+1) - p(i-1)))/hx
      Ehat(i) = E(i)+ ht*(((PI(i+1)*u(i+1)) - PI(i-1)*u(i-1))-
     &                     (j(i+1)*H(i+1)   -  j(i-1)*H(i-1))-
     &                            (q(i+1)   -   q(i-1)))/hx
      uhat(i) = jhat(i)/rhohat(i)
      phat(i)= (gama -1.d0)*(Ehat(i)-rhohat(i)*uhat(i)*uhat(i)/2.d0)
c

c
      enddo
c       do i=0,jj
c       write(*,'(i4,10f10.3)')
c     &  i,rhohat(i),jhat(i),ehat(i),phat(i),uhat(i)
c       enddo
c
c       print *,rho(24),ht,j(28+1),j(28-1),hx
c       print *,'2',n,rhohat(28)

      do i=2,jj-2,2
      rho(i) = rhohat(i)
      u(i) = uhat(i)
      j(i) = jhat(i)
      E(i) = Ehat(i)
      p(i) = phat(i)
      T(i) = p(i)/rho(i)/(gama-1.d0)
c       omegau(i) = 1.16145d0/((Tinter*T(i))**0.14874d0)          !
Chapman-Enskog
c     &          + 0.52487d0/(exp(0.77320d0*Tinter*T(i)))
c     &          + 2.16178d0/(exp(2.43787d0*Tinter*T(i)))
c       eta(i) = 5.d0*dsqrt(m*k*T(i)/acos(-1.d0))
c     &        /(16.d0*omegau(i)*sigma**2)

c       eta(i) = 2.d0*dsqrt(m*k*T(i)/acos(-1.d0))
c     &        /(3.d0*acos(-1.d0)*sigma**2)      ! Maxwell
```

```
c      omegau = 1.16145/(vari0*T(i))**0.14874
c     &       + 0.52487/(exp(0.77320*vari0*T(i)))
c     &       + 2.16178/(exp(2.43787*vari0*T(i)))
c      eta(i) = 5.0*sqrt(acos(-1.0)*m*k*T(i))/
c     &        ((16.0*acos(-1.0)*sigma**2*omegau)*(rhoinf*cinf*lambda))

c      x = T(i)*Tinf/vari0
c      omegau = -0.0799*x**5 + 0.743*x**4 - 2.7973*x**3 + 5.485*x**2
c     &         -5.9588*x + 4.1996
c      etad(i) = 5.d0*dsqrt(acos(-1.0)*m*k*T(i))/
c     &         ((16.d0*acos(-1.0)*sigma**2*omegau)*(rhoinf*cinf*lambda))
c      eta(i) = etad(i)/(rhoinf*cinf*lambda)


c      etad(i) = 5.0*sqrt(acos(-1.0)*m*k*T(i))/
c     &         (16.0*acos(-1.0)*sigma**2*omegau)
c      eta(i) = etad(i)/(rhoinf*cinf*lambda)

c      eta(i) = etainf*(T(i))**w                 ! Power Law
      enddo
c
c
c      print *,'3',n,rho(28)

c      do i=0,jj-2,2
c      rho(i+1) = (rho(i+2)+rho(i))/2.0
c      u(i+1) = (u(i+2)+u(i))/2.0
c      p(i+1) = (p(i+2)+ p(i))/2.000
c      E(i+1) = (E(i+2)+ E(i))/2.0
c      H(i+1) = (E(i+1)+ p(i+1))/rho(i+1)
c      enddo


c      do i=0,jj
c      write(*,'(A1,i4,10f10.3)')
c     &  'e',i,rho(i),j(i),e(i),p(i),u(i)
c      enddo
c
c      print *,'4',n,rho(28)

      elseif (eos .eq. 1) then ! M-G EOS
      umax = 0.d0
      do i = 0,jj
      uloc = abs(u(i))+dsqrt((gama0+1.d0)*gama0*T(i))
      if (uloc .gt. umax) umax=uloc
      enddo
      ht = 0.001*hx/uloc
c      print *,'ht=',ht

c      print *,'integer time ',n
      do i= 0,jj-2,2    !space
c calculate average
      j(i+1) =    (rho(i+2)*u(i+2) + rho(i)*u(i))/2.d0
      PI(i+1) = 4.d0*(eta(i+2)+ eta(i))*(u(i+2)-u(i))/hx/6.d0
      q(i+1) = -1.d0*(gama0+1.d0)*(eta(i+2)+ eta(i))*
     &         (T(i+2)-T(i))/(hx*Pr*2.d0)
c      q(i+1) = - kappa0*(T(i+2)-T(i))/hx
c      q(i+1) = - (eta(i+2)+ eta(i))*(T(i+2)-T(i))/hx/((gama0)*Pr)/2.d0
c     T(i+1) = gama*p(i+1)/rho(i+1)
c      E(i) = rho(i)*u(i)*u(i)/2.d0+p(i)/gama0
      rho(i+1) = (rho(i+2)+ rho(i))/2.d0
      u(i+1) = (u(i+2)+ u(i))/2.d0
      p(i+1) = (p(i+2)+ p(i))/2.d0
      E(i+1) = (E(i+2)+ E(i))/2.d0
      H(i+1) = (E(i+1)+ p(i+1))/rho(i+1)
      enddo

c      do i=0,jj
c      write(*,'(i4,10f10.3)')
c     &  i,j(i),PI(i),q(i),T(i),E(i),rho(i),u(i),p(i),e(i),h(i)
c      enddo
c
c      print *,'1',n,rho(24)

      do i=2,jj-2,2
c
      q(i) = (q(i-1)+q(i+1))/2.d0
      rhohat(i)=rho(i)-ht*(j(i+1)-j(i-1))/hx
      jhat(i) = rho(i)*u(i) + ht*(  (PI(i+1)    -  PI(i-1))
     &                          -(j(i+1)*u(i+1) - j(i-1)*u(i-1))
     &                                 -(p(i+1) - p(i-1)))/hx
      Ehat(i) = E(i)+ ht*(((PI(i+1)*u(i+1)) - PI(i-1)*u(i-1))-
     &                     (j(i+1)*H(i+1)   -  j(i-1)*H(i-1))-
     &                            (q(i+1)   -   q(i-1)))/hx
      uhat(i) = jhat(i)/rhohat(i)
      x = 1.d0 - rho(0)/rhohat(i)
c      phat(i) = -
gama0**5*x**4*e00/(24.d0*cinf**2)+gama0**2*x**4*s/12.d0
c     &       - 3.d0*gama0*x**4*s**2/4.d0
c     &       + (x + (2.d0*s- gama0/2.d0)*x**2 + s*(3.d0*s -
gama0)*x**3)
c     &       + gama0*rho(0)*(Ehat(i)-rhohat(i)*uhat(i)**2/2.d0)
c     &        /rhohat(i) ! 4th order eos
      phat(i) = (x + (2.d0*s - gama0/2.d0)*x**2
     &          + s*(3.d0*s - gama0)*x**3)*C0**2/cinf**2
     &        + gama0*rho(0)*(Ehat(i)-rhohat(i)*uhat(i)**2/2.d0)
     &          /rhohat(i) ! 3rd order eos

c      print *,'values to find pressure'
c      print *, 'rho(0)',rho(0),'x',x,'s',s,'gama0',gama0
c      print *, 'rhohat',rhohat(i),'Ehat',Ehat(i),'uhat(i)',uhat(i)
c      print *, 'phat(i)', phat(i)
c      pause
c      phat(i) = gama0*((Ehat(i)-rhohat(i)*uhat(i)**2/2.d0)
c      phat(i) = p(0)
c     &        + gama0*rho(0)*((Ehat(i)-rhohat(i)*uhat(i)**2/2.d0)
c     &          /rhohat(i)
c     &        - ie(0)/rho(0))
c      phat(i) = gama0*rhoref*(Ehat(i)-rhohat(i)*uhat(i)*uhat(i)/2.0
c     &          - eref) + Pref
c      phat(i)= gama0*(Ehat(i)-rhohat(i)*uhat(i)*uhat(i)/2.0)
c
      enddo

c      if (n/100 .eq. float(n)/100.) then
c      do bug = 0,jj,10   !debug
c      write (*,'(2i5,3f16.5)')n,bug,phat(bug),Ehat(bug),rhohat(bug)
c      enddo
c      pause
c      endif
c      do i=0,jj
c      write(*,'(i4,10f10.3)')
c     &  i,rhohat(i),jhat(i),ehat(i),phat(i),uhat(i)
c      enddo
c
c      print *,rho(24),ht,j(28+1),j(28-1),hx
c      print *,'2',n,rhohat(28)

      do i=2,jj-2,2
      rho(i) = rhohat(i)
      u(i) = uhat(i)
      j(i) = jhat(i)
      E(i) = Ehat(i)
      p(i) = phat(i)
      T(i) = T(0)+ (E(i)-rho(i)*u(i)**2/2.d0)/rho(i)
     &          - (E(0)-rho(0)*u(0)**2/2.d0)/rho(0)
      PI(i) = 4.d0*eta(i)*(u(i+2)-u(i))/2.d0/hx/3.d0

c      omegau(i) =
1.16145/((Tinter*T(i))**0.14874)                     ! Chapman-Enskog
c     &           + 0.52487/(exp(0.77320*Tinter*T(i)))
c     &           + 2.16178/(exp(2.43787*Tinter*T(i)))
c      eta(i) = 5.0*sqrt(m*k*T(i)/acos(-1.0))/(16.0*omegau(i)*sigma**2)
c      eta(i) = 2.0*sqrt(m*k*T(i)/acos(-1.0))/(3.0*acos(-
1.0)*sigma**2) ! Maxwell

c      omegau = 1.16145/(vari0*T(i))**0.14874
c     &       + 0.52487/(exp(0.77320*vari0*T(i)))
c     &       + 2.16178/(exp(2.43787*vari0*T(i)))
c      eta(i) = 5.0*sqrt(acos(-1.0)*m*k*T(i))/
c     &        ((16.0*acos(-1.0)*sigma**2*omegau)*(rhoinf*cinf*lambda))
c      x = T(i)*Tinf/vari0
c      omegau = -0.0799*x**5 + 0.743*x**4 - 2.7973*x**3 + 5.485*x**2
c     &         -5.9588*x + 4.1996
c      etad(i) = 5.0*sqrt(acos(-1.0)*m*k*T(i))/
c     &         (16.0*acos(-1.0)*sigma**2*omegau)
c      eta(i) = etad(i)/(rhoinf*cinf*lambda)

c      eta(i) =
etainf*(T(i))**w                                        ! Power Law
      enddo
```

```
c
c      print *,'3',n,rho(28)

c      do i=0,jj-2,2
c      rho(i+1) = (rho(i+2)+rho(i))/2.0
c      u(i+1) = (u(i+2)+u(i))/2.0
c      p(i+1) = (p(i+2)+ p(i))/2.0
c      E(i+1) = (E(i+2)+ E(i))/2.0
c      H(i+1) = (E(i+1)+ p(i+1))/rho(i+1)
c      enddo

c      do i=0,jj
c      write(*,'(A1,i4,10f10.3)')
c     &  'e',i,rho(i),j(i),e(i),p(i),u(i)
c      enddo
c
c      print *,'4',n,rho(28)
      elseif (eos .eq. 2) then ! ideal gas EOS general dimensionless
      umax = 0.
      do i = 0,jj
      uloc = abs(u(i))+dsqrt(gama*R*T(i)/Cv)
      if (uloc .gt. umax) umax=uloc
      enddo
      ht = 0.001*hx/uloc

c      print *,'ht=',ht

c      print *,'integer time ',n
      do i= 0,jj-2,2    !space
c calculate average
      j(i+1) =    (rho(i+2)*u(i+2) + rho(i)*u(i))/2.d0
      PI(i+1) = 4.d0*(eta(i+2)+ eta(i))*(u(i+2)-u(i))/hx/6.d0
      q(i+1) = - kappa0*(T(i+2)-T(i))/hx
c      q(i+1) = - (eta(i+2)+ eta(i))*(T(i+2)-T(i))/hx/((gama0)*Pr)/2.0
c     T(i+1) = gama*p(i+1)/rho(i+1)
      E(i) = rho(i)*u(i)*u(i)/2.d0+p(i)/gama0
      rho(i+1) = (rho(i+2)+ rho(i))/2.d0
      u(i+1) = (u(i+2)+ u(i))/2.d0
      p(i+1) = (p(i+2)+ p(i))/2.d0
      E(i+1) = (E(i+2)+ E(i))/2.d0
      H(i+1) = (E(i+1)+ p(i+1))/rho(i+1)

      enddo
c      do i=0,jj
c      write(*,'(i4,10f10.3)')
c     &  i,j(i),PI(i),q(i),T(i),E(i),rho(i),u(i),p(i),e(i),h(i)
c      enddo
c
c      print *,'1',n,rho(24)
      do i=2,jj-2,2
c
      q(i) = (q(i-1)+q(i+1))/2.d0
      rhohat(i)=rho(i)-ht*(j(i+1)-j(i-1))/hx
      jhat(i) = rho(i)*u(i) + ht*(  (PI(i+1)    -  PI(i-1))
     &                          -(j(i+1)*u(i+1) - j(i-1)*u(i-1))
     &                                 -(p(i+1) - p(i-1)))/hx
      Ehat(i) = E(i)+ ht*(((PI(i+1)*u(i+1)) - PI(i-1)*u(i-1))-
     &                     (j(i+1)*H(i+1)   -  j(i-1)*H(i-1))-
     &                            (q(i+1)   -   q(i-1)))/hx
      uhat(i) = jhat(i)/rhohat(i)
      phat(i)= (gama -1.d0)*(Ehat(i)-rhohat(i)*uhat(i)*uhat(i)/2.d0)
c      phat(i) = gama0*rhoref*(Ehat(i)-rhohat(i)*uhat(i)*uhat(i)/2.0
c     &         - eref) + Pref
c      phat(i)= gama0*(Ehat(i)-rhohat(i)*uhat(i)*uhat(i)/2.0)
c
c      if (n/100 .eq. float(n)/100.) then
c      do bug = 0,jj,10   !debug
c      write (*,'(2i4,f12.5)')n,bug,phat(bug)
c      enddo
c      pause
c      endif
c
      enddo
c      do i=0,jj
c      write(*,'(i4,10f10.3)')
c     &  i,rhohat(i),jhat(i),ehat(i),phat(i),uhat(i)
c      enddo
c
c      print *,rho(24),ht,j(28+1),j(28-1),hx
c      print *,'2',n,rhohat(28)

      do i=2,jj-2,2
      rho(i) = rhohat(i)
      u(i) = uhat(i)
      j(i) = jhat(i)
      E(i) = Ehat(i)
      p(i) = phat(i)
c      T(i) = p(i)/rho(i)/(gama-1.d0)
      T(i) = T(0)+ (e(i)-rho(i)*u(i)**2/2.d0)/rho(i)
     &           - (e(0)-rho(0)*u(0)**2/2.d0)/rho(0)
c      omegau(i) =
1.16145/((Tinter*T(i))**0.14874)                      ! Chapman-Enskog
c     &           + 0.52487/(exp(0.77320*Tinter*T(i)))
c     &           + 2.16178/(exp(2.43787*Tinter*T(i)))
c      eta(i) = 5.0*sqrt(m*k*T(i)/acos(-1.0))/(16.0*omegau(i)*sigma**2)
c      eta(i) = 2.0*sqrt(m*k*T(i)/acos(-1.0))/(3.0*acos(-
1.0)*sigma**2)  ! Maxwell

c      omegau = 1.16145/(vari0*T(i))**0.14874
c     &       + 0.52487/(exp(0.77320*vari0*T(i)))
c     &       + 2.16178/(exp(2.43787*vari0*T(i)))
c      eta(i) = 5.0*sqrt(acos(-1.0)*m*k*T(i))/
c     &        ((16.0*acos(-1.0)*sigma**2*omegau)*(rhoinf*cinf*lambda))
c      x = T(i)*Tinf/vari0
c      omegau = -0.0799*x**5 + 0.743*x**4 - 2.7973*x**3 + 5.485*x**2
c     &         -5.9588*x + 4.1996
c      etad(i) = 5.0*sqrt(acos(-1.0)*m*k*T(i))/
c     &         (16.0*acos(-1.0)*sigma**2*omegau)
```

```
c      eta(i) = etad(i)/(rhoinf*cinf*lambda)

c      eta(i) =
etainf*(T(i))**w                                         ! Power Law
      enddo
c
c      print *,'3',n,rho(28)

c      do i=0,jj-2,2
c      rho(i+1) = (rho(i+2)+rho(i))/2.0
c      u(i+1) = (u(i+2)+u(i))/2.0
c      p(i+1) = (p(i+2)+ p(i))/2.0
c      E(i+1) = (E(i+2)+ E(i))/2.0
c      H(i+1) = (E(i+1)+ p(i+1))/rho(i+1)
c      enddo


c      do i=0,jj
c      write(*,'(A1,i4,10f10.3)')
c     &  'e',i,rho(i),j(i),e(i),p(i),u(i)
c      enddo
c
c      print *,'4',n,rho(28)

c------------------------------liquid------------------------------
      elseif (eos .eq. 3) then !liquid M-G
      umax = 0.
      do i = 0,jj
c      uloc = abs(u(i))+dsqrt(gama*R*T(i)/Cv)
      uloc = abs(u(i))+1.d0
      if (uloc .gt. umax) umax=uloc
      enddo
      ht = 0.001*hx/uloc

c      print *,'ht=',ht
c      print *,'integer time ',n


      do i= 0,jj-2,2    !space
c calculate average
      j(i+1) =  (rho(i+2)*u(i+2) + rho(i)*u(i))/2.d0
      PI(i+1) = (4.d0/3.d0+uv)*((eta(i+2)+ eta(i))/2.d0)
     &         *(u(i+2)-u(i))/hx
      q(i+1) = -kappa0*(T(i+2)-T(i))/hx
      rho(i+1) = (rho(i+2)+ rho(i))/2.d0
      u(i+1) = (u(i+2)+ u(i))/2.d0
      p(i+1) = (p(i+2)+ p(i))/2.d0
      E(i+1) = (E(i+2)+ E(i))/2.d0
      H(i+1) = (E(i+1)+ p(i+1))/rho(i+1)

c      if (n/1000 .eq. float(n)/1000.) then   !debug
c      print *,'debug',n,E(i)
c      endif
      enddo
```

```
c       print *,Pr, gama, mu, R, Tinf, rhoinf, cinf,lambda,eta(0)
c       print *, kappa0,eta(0)/((gama-1.d0)*Pr)
c       pause

      do i=2,jj-2,2
c
      q(i) = (q(i-1)+q(i+1))/2.d0
      rhohat(i)=rho(i)-ht*(j(i+1)-j(i-1))/hx
      jhat(i) = rho(i)*u(i) + ht*(  (PI(i+1)    -  PI(i-1))
     &                          -(j(i+1)*u(i+1) - j(i-1)*u(i-1))
     &                                 -(p(i+1) - p(i-1)))/hx
      Ehat(i) = E(i)+ ht*(((PI(i+1)*u(i+1)) - PI(i-1)*u(i-1))-
     &                     (j(i+1)*H(i+1)   -  j(i-1)*H(i-1))-
     &                            (q(i+1)   -   q(i-1)))/hx
      uhat(i) = jhat(i)/rhohat(i)
      x = 1.d0 - rho(0)/rhohat(i)
c      phat(i) = -
gama0**5*x**4*e00/(24.d0*cinf**2)+gama0**2*x**4*s/12.d0
c     &       - 3.d0*gama0*x**4*s**2/4.d0
c     &       + (x + (2.d0*s- gama0/2.d0)*x**2 + s*(3.d0*s -
gama0)*x**3)
c     &       + gama0*rho(0)*(Ehat(i)-rhohat(i)*uhat(i)**2/2.d0)
c     &        /rhohat(i) ! 4th order eos
      phat(i) = (x + (2.d0*s - gama0/2.d0)*x**2 + s*(3.d0*s-gama0)*x**3)
     &        + gama0*rho(0)*(Ehat(i)-rhohat(i)*uhat(i)**2/2.d0)
     &        /rhohat(i) ! 3rd order eos
c      print *,'values to find pressure'
c      print *, 'rho(0)',rho(0),'x',x,'s',s,'gama0',gama0
c      print *, 'rhohat',rhohat(i),'Ehat',Ehat(i),'uhat(i)',uhat(i)
c      print *, 'phat(i)', phat(i)
c      pause
c      phat(i) = gama0*((Ehat(i)-rhohat(i)*uhat(i)**2/2.d0)
c      phat(i) = p(0)
c     &        + gama0*rho(0)*((Ehat(i)-rhohat(i)*uhat(i)**2/2.d0)
c     &          /rhohat(i)
c     &        - ie(0)/rho(0))
c      phat(i) = gama0*rhoref*(Ehat(i)-rhohat(i)*uhat(i)*uhat(i)/2.0
c     &          - eref) + Pref
c      phat(i)= gama0*(Ehat(i)-rhohat(i)*uhat(i)*uhat(i)/2.0)
c
c      if (n/100 .eq. float(n)/100.) then
c      do bug = 0,jj,10   !debug
c      write (*,'(2i4,f12.5)')n,bug,phat(bug)
c      enddo
c      pause
c      endif
c
      enddo
c      do i=0,jj
c      write(*,'(i4,10f10.3)')
c     &  i,rhohat(i),jhat(i),ehat(i),phat(i),uhat(i)
c      enddo
c
```

```
c      print *,rho(24),ht,j(28+1),j(28-1),hx
c      print *,'2',n,rhohat(28)

      do i=2,jj-2,2
      rho(i) = rhohat(i)
      u(i) = uhat(i)
      j(i) = jhat(i)
      E(i) = Ehat(i)
      p(i) = phat(i)
      T(i) = T(0)+ (e(i)-rho(i)*u(i)**2/2.d0)/rho(i)
     &           - (e(0)-rho(0)*u(0)**2/2.d0)/rho(0)
      enddo

c      do i=0,jj
c      write(*,'(A1,i4,10f10.3)')
c     &  'e',i,rho(i),j(i),e(i),p(i),u(i)
c      enddo

      endif

      if (n/10000 .eq. float(n)/10000.) then
      print *,'Time step n=',n
      if     (n .gt. 99 .and. n .le. 999) then
      write(filen(8:10),'(i3)') n
c      elseif (n .gt. 999 .and. n .le. 9999) then
c      write(filen(7:10),'(i4)') n
c      elseif (n .gt. 9999 .and. n .le. 99999) then
c      write(filen(6:10),'(i5)') n
c      elseif (n .gt. 99999 .and. n .le. 999999) then
c      write(filen(5:10),'(i6)') n
      elseif (n .gt. 999999 .and. n .le. 9999999) then
      write(filen(4:10),'(i7)') n
      open(unit=22,file=filen,form='formatted',status='unknown')
      do i=0,jj
      write (22,'(81f12.5)') (hx/2.d0)*(float(i)-float(jj)/2.d0),
     & p(i),rho(i),(rho(i)-rho(0))/(rho(jj)-rho(0))
      enddo
      close(22)
c      write(entropy(4:10),'(i7)') n
      elseif (n .gt. 9999999 .and. n .le. 99999999) then
      write(filen(3:10),'(i8)') n
      open(unit=22,file=filen,form='formatted',status='unknown')
      do i=0,jj
      write (22,'(81f12.5)') (hx/2.d0)*(float(i)-float(jj)/2.d0),
     & p(i),rho(i),(rho(i)-rho(0))/(rho(jj)-rho(0))
      enddo
      close(22)
c      write(entropy(3:10),'(i8)') n
      endif

c      open(unit=22,file=filen,form='formatted',status='unknown')
c      print *,'Time step n=',n
c      do i=0,jj
c      write (22,'(81f12.5)') (hx/2.d0)*(float(i)-float(jj)/2.d0),
c     & p(i),rho(i),(rho(i)-rho(0))/(rho(jj)-rho(0))
c      enddo
c      close(22)

      if(eos .eq. 0) then
c      open(unit=21,file=entropy,form='formatted',status='unknown')
      dsus = 0
      dsTs = 0
      do i = 2,jj-2,2
      dsu(i) = 4.d0*eta(i)*((u(i+2)-u(i-2))/(2.d0*deltax))**2*deltax
     &         /(3.d0*T(i))/(rho(i)*u(i))
      dsus = dsus + dsu(i)
      dsT(i) =-kappa*((T(i+2)-2.d0*T(i)+T(i-2))/(deltax**2))*deltax
     &   /T(i)/(rho(i)*u(i))
      dsT(i) =((q(i+2)-q(i-2))/(2.d0*hx))*hx/T(i)/(rho(i)*u(i))
      dsTs = dsTs + dsT(i)
      enddo
      ds = (dsus -dsTs)*R*gama
      entropy(n) = ds
c      write (21,'(4f12.5)') dsa, dsus*R*gama,dsTs*R*gama, ds
c      close(21)
c
c      do i = 0,jj
c      write (* ,'(i4,f12.5)') i,rho(i)
c      write (23,'(f12.6)') hx*(float(i)-float(jj)/2.0)
c      enddo
c         pause
c      close(23)

      elseif(eos .ne. 0) then
      dsus = 0
      dsTs = 0
      do i = 2,jj-2,2
      dsu(i) = 4.d0*eta(i)*((u(i+2)-u(i-2))/(2.d0*deltax))**2*deltax
     &         /(3.d0*T(i))/(rho(i)*u(i))
      dsus = dsus + dsu(i)
      dsT(i) =-kappa*((T(i+2)-2.d0*T(i)+T(i-2))/(deltax**2))*deltax
     &   /T(i)/(rho(i)*u(i))
      dsT(i) =((q(i+2)-q(i-2))/(2.d0*hx))*hx/T(i)/(rho(i)*u(i))
      dsTs = dsTs + dsT(i)
      enddo
c      do i = 2,jj-2,2
c      dsu(i) = 4.d0*eta(i)*((u(i+2)-u(i))/deltax)**2*(deltax)
c     &         /(3.d0*T(i))/(rho(i)*u(i))
c      dsus = dsus + dsu(i)
c      dsT(i) =-gama*eta(i)/Pr*((T(i+2)-2*T(i)+T(i-2))/((deltax)**2))
c     &        *(deltax)/T(i)/(rho(i)*u(i))
c      dsT(i) =((q(i+2)-q(i))/(deltax))*(deltax)/T(i)/(rho(i)*u(i))
c      dsTs = dsTs + dsT(i)
c     enddo
      ds = (dsus -dsTs)*Cv
      entropy(n) = ds
      endif
```

```
c output restart
      print *,'Write restart file...'
      open (unit=44, file='restart.dat', form='formatted',
     &      status='unknown')
       do i = 0,jj,1 !Zero out variables
      write (44,'(i15,6e25.18)') n,rho(i),u(i),P(i),e(i),T(i),eta(i)
       enddo
       close(44)
c       endif

      endif

      enddo !j-loop

c---------------analytic solution is only available for gas------------
c-------------------------analytic solution---------------------------
      print *,'Analytic solution'
      open (unit=41,file='shock.dat',form='formatted',status='unknown')
      open(unit=21,file='entropy.dat',form='formatted',status='unknown')

      if (eos .ne. 3) then
c caculate parameters

c      print *,'alpha2=',alpha,' beta=',beta
      c1= sqrt(gama*R*Tinf)                            !sound speed of
inflow [cm/us]
c      print *,'alpha2=',alpha,' beta=',beta
      x0=0.*t(1)*((gama+1.0)*u(0)/4.0 + !shock position from landau and
liftshitz  pg 358. Why there is a 0. before t(n),besides, is t(n) the
same mean as T(n)?
     &  sqrt((gama+1.0)**2*u(0)**2/16.0+ c1**2 ))
c      print *,'alpha2=',alpha,' beta=',beta
      x0=0.*x0/lambda
c      print *,'alpha2=',alpha,' beta=',beta
      M1= sqrt(((p(jj)/p(0))*(gama+1.0)+gama-1.0)/(2.d0*gama) )! mach#
for the stationary shock solution
      v1=M1*c1                                                 !
upstream velocity
      alpha=(gama-1.d0)/(gama+1.d0)+2.d0/((gama+1.d0)*M1**2)   ! alpha,
asymptote of analytic solution see BSL notes
      beta=9.0*(gama+1.0)*sqrt(acos(-1.0)/8.0/gama)/8.0        ! beta,
from transport phonomena book P352
c     lambda=3.d0*mu/rho(0)*       ! mean free path [cm], use value in
numerical method
c    &   dsqrt(dacos(-1.d0)*M1/8.d0/R/T(0))

c bisection method

      do i=0,jj-2,2                    ! space step through domain and
find the analytic solution psi
      xx =-5.0+(5.0+5.0)*float(i)/(float(jj)-1.0)       !dimensionless
position
```

```
      xa = 1.0                         ! first guess, psi must be
between xa and xb
      xb = alpha                       ! xb=alpha will always make f>0
c     print *,'alpha2=',alpha,' beta=',beta
c     print *,'psialpha3',alpha
c     pause

      fa   = 1.0-xa-(xa-alpha)**alpha                        ! analytic
solution at xa
     &      *exp(beta*M1*(1.0-alpha)*(xx-x0))
      fb   = 1.0-xb-(xb-alpha)**alpha                        ! analytic
solution at xb
     &      *exp(beta*M1*(1.0-alpha)*(xx-x0))
      print *,'xa and xb',xx,xa,xb,fa,fb

       if (fa .eq. 0.0) then                                 ! xa first
guess was right!
       psi(i) = xa
       goto 99
       elseif (fb .eq. 0.0) then                             ! xb first
guess was right!
       psi(i) = xb
       goto 99
       endif

       if (fa*fb .ge. 0) pause'root must be bracketed'!determine
interval limits
       if (fa .lt. 0) then                       !orient the search to
keep fa<0
       dx=xb-xa
       xa=xa
       else
       dx=xa-xb
       xa=xb
       endif
      print *,'alpha5=',alpha,' beta=',beta
      print *,xa,xb,fa,fb
      do iter=1,1000            ! iterate bisection method to find psi
at each r position
c      print *,iter,'alpha6=',alpha,' beta=',beta
       dx=dx*0.5
       xb=xa+dx
       fb=1.0-xb-(xb-alpha)**alpha
     &      *exp(beta*M1*(1.0-alpha)*(xx-x0))
c      print *,iter,xx,xa,xb,dx,fb
       if (fb .le. 0) xa=xb
c      print *,iter,'alpha6=',alpha,' beta=',beta
c      if(fb .eq. 0) return
       enddo
c     print *,'out of loop',j
      psi(i)=xb

 99   continue      ! solution converged
c      pause
```

```
      write(41,'(I5,10e13.4)')
     &  i,xx,psi(i),
     &  (1.0/psi(i)-rho(0))/(rho(jj)-rho(0)),
     &  hx*(dfloat(i)-dfloat(jj)/2.d0)/2.0d0,
     &   u(i)*c1/v1,v1/u(i)/c1,
     &  (rho(i)-rho(0))/(rho(jj)-rho(0)),
     &  q(i), PI(i)

      enddo
c       print *,Pr, gama, mu, R, Tinf, rhoinf, cinf,lambda,eta(0)
c       print *, kappa0,eta(0)/((gama-1.d0)*Pr)
c       pause

      else
      do i = 0,jj
      write(41,'(I5,4e13.4)')
     &  i,
     &  hx*(dfloat(i)-dfloat(jj)/2.d0)/2.0d0,
     &  (rho(i)-rho(0))/(rho(jj)-rho(0)),
     &  p(i)*Pless/(1.d9)
      enddo
      endif

      write(21,'(f12.5)') dsa
      do counter = 0, n, 10000
      write(21,'(f12.5)') entropy(counter)
      enddo
      end
```