Shock Instability in Gases Characterized by Inelastic Collisions

by

Nick Sirmas

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the M.A.Sc. degree in
Mechanical Engineering

The Ottawa-Carleton Institute for Mechanical and Aerospace Engineering
Faculty of Engineering
University of Ottawa

Abstract

The current study addresses the stability of shock waves propagating through dissipative media, analogous to both granular media and molecular gases undergoing endothermic reactions. In order to investigate the stability, a simple molecular dynamics model was developed to observe shock waves and their structures with the inclusion of energy dissipation. For this, an Event Driven Molecular Dynamics model was implemented in a 2D environment, where a molecule is represented by a disk. The simulations addressed the formation of a shock wave in a gas by the sudden acceleration of a piston. Inelastic collisions were assumed to occur only if an impact velocity threshold is surpassed, representing the activation energy of the dissipative reactions.

Parametric studies were conducted for this molecular model, by varying the strength of the shock wave, the activation threshold and the degree of inelasticity in the collisions. The resulting simulations showed that a shock structure does indeed become unstable with the presence of dissipative collisions. This instability manifests itself in the form of distinctive high density non-uniformities behind the shock wave, which take the form of convective rolls. The spacing and size of this "finger-like" unstable pattern was shown to be dependent on the degree of inelasticity, the activation energy, and the strength of the driving piston.

The mechanism responsible for the instability was addressed by studying the time evolution of the material undergoing the shock wave compression and further relaxation. It is found that the gas develops the instability on the same time scales as the clustering instability in homogeneous gases, first observed by Goldhirsch and Zanetti in granular gases. This confirmed that the clustering instability is the dominant mechanism.

Acknowledgements

I would like to thank my supervisor, Dr. Matei Radulescu, for his guidance and support throughout my studies. His mentoring has played a significant role in the completion of the current work and in my development within the field.

I would also like to thank my colleagues from the Detonation & Reactive Dynamics Laboratory at the University of Ottawa for their input and help.

Contents

List of Figures

List of Tables

Nomenclature

Roman Symbols

A	Area of domain, $A = L_x \times L_y$
c	Speed of sound
D	Velocity of shock wave
e	Internal energy
k	Boltzmann constant
l_r	Relaxation length scale
L_x	Length of domain in x-direction
L_y	Length of domain in y-direction
M	Mach number
m	Mass of particle
N	Number of hard disks/particles
n	Number density of hard disks/particles ($n = N/A$)
R	Radius of hard particle/disk
T	Temperature
t_{clust}	Time to clustering instability in homogeonous cooling granular gas
u	Velocity of entity
u_{max}	Activation impact velocity threshold
u_{rms}	Root mean squared velocity
v	Specific volume
x	x-coordinate of entity

y y-coordinate of entity

Z Compressibility factor

Greek Symbols

β Ratio between kinetic energy imparted by piston and the initial energy of particles

δ Width of non-uniformities

ε Coefficient of restitution

γ Isentropic exponent of system of hard disks

η Packing factor, $\eta = \pi R^2 n$

λ Mean free path

π Ratio of pressure in equilibrated shocked state to initial state

ρ Density

σ Ratio of specific volume in equilibrated shocked state to initial state

τ Mean free time

τ_R Exponential decay time constant

ψ Shock strength parameter

Subscripts

$(\,)_1$ Initial state

$(\,)_2$ Equilibrated shocked state

$(\,)_i$ Property of ith particle

$(\,)_p$ Property of piston

Chapter 1

Introduction

Shock waves are strong compression waves giving rise to a sharp change in thermodynamic properties. When shock waves propagate in a reactive medium, the induced exothermicity can trigger instability in the shock structure. Detonation waves in gases have been long known to have a cellular structure and pulsating velocities (Fickett & Davis, 2000). On the other hand, shock waves propagating through inert gases are usually stable, leading to a uniform compressed state behind the shock wave. However, there are also indications from experiments that shock waves through inert gases can become unstable. The goal of the present study is to investigate this instability, which appears in some shock waves propagating in inert gases, and investigate the physical mechanism that controls this phenomenon.

1.1 Shock instability

The stability of shock waves has been reviewed by Zeldovich & Raizer (1966) and Landau & Lifshitz (1987). Typically, most shock waves are stable. Figure 1.1a shows the structure of a piston supported shock wave. By stability we refer to the stability of the planar travelling wave solution (Whitham, 1974). Shocks with departures of the planarity of the shock front and with spatial and/or temporal oscillations from the uniform conditions are labelled unstable. One such example is shown in Figure 1.1b, where an unstable shock wave may form ripples or corrugations on its front.

Unstable shock structures have been observed experimentally in strong shocks in noble gases, like argon. The experiments suggest that these shock waves become unstable with the presence of ionization (Grun *et al.*, 1991; Glass & Liu, 1978; Griffiths *et al.*, 1976). Instability in shock waves propagating in argon has also been observed in numerical sim-

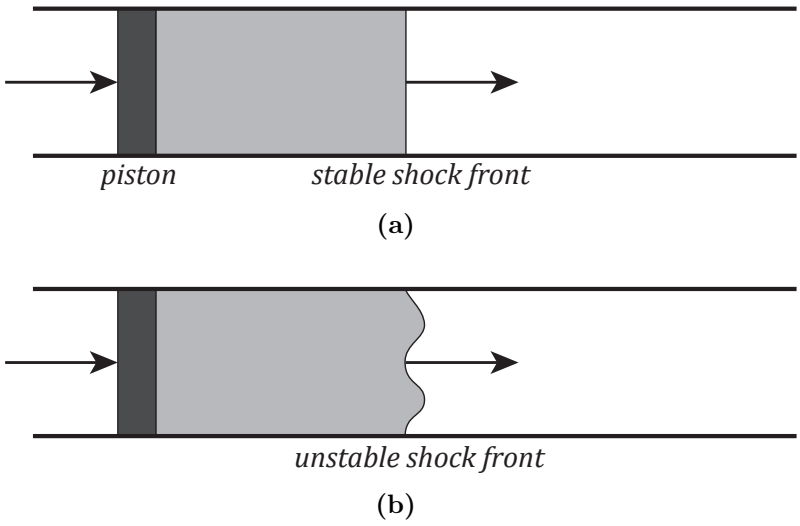


Figure 1.1: Schematic of a piston propagated shock wave generating a) stable and b) unstable structures

ulations by Kapper & Cambier (2011) and Mond *et al.* (1997). Similar instabilities have also been observed in sufficiently strong shock waves when molecular dissociation occurs (Griffiths *et al.*, 1976). Further experiments have revealed that shock waves can also become unstable when neither ionization or dissociation is expected. This has been observed in shock waves through gases composed of heavy molecules, such as carbon dioxide (Griffiths *et al.*, 1976; Mishin *et al.*, 1981; Hornung & Lemieux, 2001), propane (Hornung & Lemieux, 2001), and freon (Semenov *et al.*, 2012). Figure 1.2 shows experimental results obtained by Hornung & Lemieux (2001) for stable and unstable shock structures around a projectile travelling through propane, with similar results obtained for carbon dioxide. These results demonstrate that there exists a critical Mach number above which the instability is manifested. The shock instability observed with these heavier gases is puzzling, as the molecular structure does not change through dissociation or ionization.

Currently, these instabilities are not well understood. The ability to predict when these shock waves become unstable is lacking. Current models for the instability only consider the equilibrium states before and after the shock. The locus of the post shock states, usually represented in a pressure-specific volume diagram, is referred to as the Hugoniot curve (Zeldovich & Raizer, 1966); an example is given in Figure 1.3. The models relate the instability of inert shock waves with the shape of the shock Hugoniot (Fickett & Davis, 2000). For reference, the Hugoniot of a typical substance is a convex curve, such as that shown in Figure 1.3. For a given upstream state (A), the possible post shock states

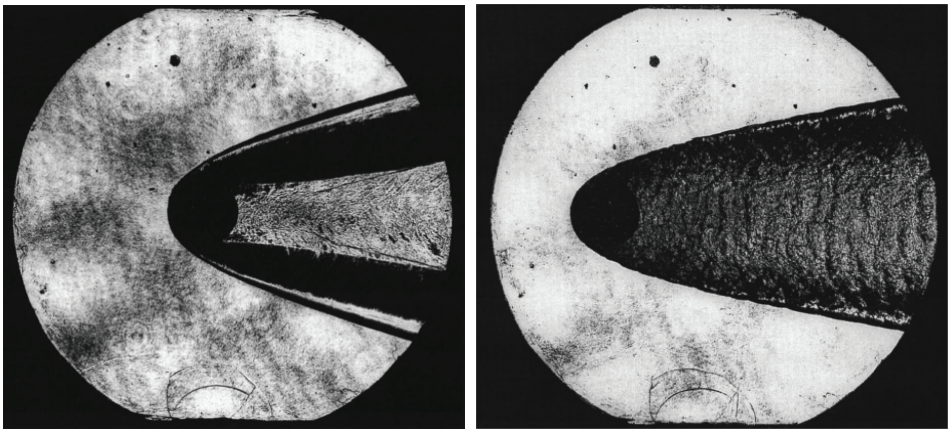


Figure 1.2: Experimental images of shock formation over a projectile propagating through propane. The stable shock structure (left) is shown from the projectile travelling at 2.26 km/s, and the unstable structure (right) at 2.7 km/s (Hornung & Lemieux, 2001)

(B) lie on the Hugoniot curve. Depending on the mass flux across the shock wave, or alternatively on the shock strength, different solutions are accessible along the Hugoniot curve. The Rayleigh line, also shown in Figure 1.3, shows the locus of non-equilibrium states that satisfy the one-dimensional inviscid mass and momentum conservation laws; steeper Rayleigh lines correspond to higher mass fluxes across the wave. The intersection of the Rayleigh and Hugoniot curves gives the post shock state.

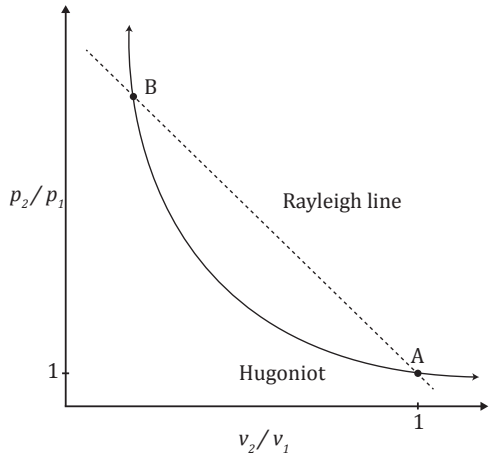


Figure 1.3: Hugoniot structure in which stable shock waves are predicted, where the post shock state B is found from the intersection of the Rayleigh line extending from the initial state A

For unstable shock waves, the Hugoniot is expected to take on a more complex shape,

as shown in the Figure 1.4 (Fickett & Davis, 2000; Zeldovich & Raizer, 1966). Figure 1.4a shows the Hugoniot characteristic of a material undergoing a phase transition, which is shown to become unstable (Fickett & Davis, 2000) in the vicinity of the phase transition point (C). Due to the multiple states characterized by the intersection of the Hugoniot and Rayleigh lines, the shock wave is no longer a single shock structure, but can split into two sequential shock waves.

Media characterized by the Hugoniot shown in Figure 1.4b are also expected to sustain unstable shock waves. This Hugoniot is characteristic of dissociating gases. Instability is expected when the Rayleigh line intersects a segment of the Hugoniot with a positive slope (B). Instability caused by the positive slope is associated with the D'yakov-Kontorovich (DK) instability. The DK criterion based on the slope of the Hugoniot, identifies whether the shock wave will be stable or unstable (Bates & Montgomery, 2000; Landau & Lifshitz, 1987).

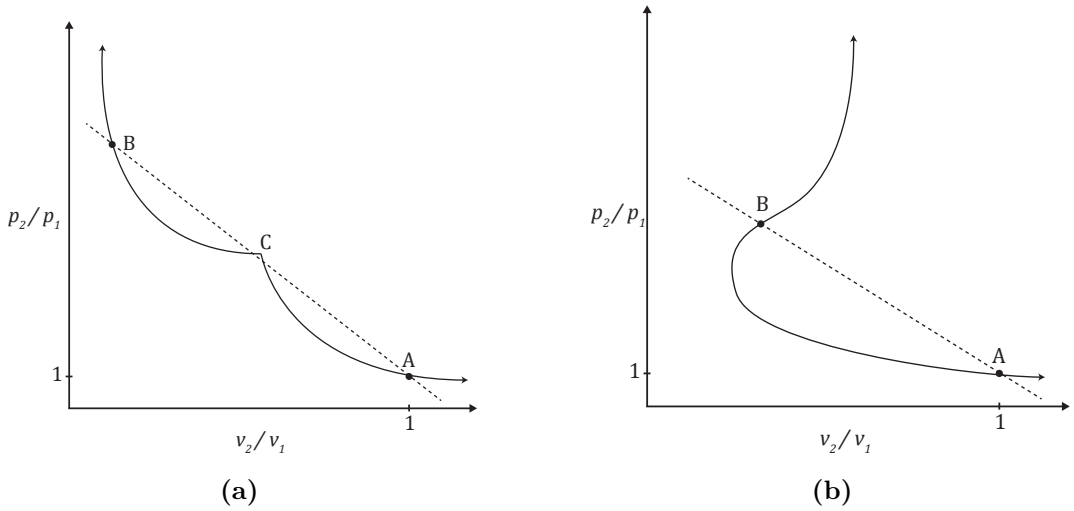


Figure 1.4: Expected Hugoniot structures where unstable shock waves are expected in a) media undergoing a phase transition and b) gas undergoing dissociation

The shape of the Hugoniots give theoretical predictions for when shock waves are expected to become unstable. However, they have not been able to predict instability for all cases that instability has been observed experimentally. For instance, results by Hornung & Lemieux (2001) and Griffiths *et al.* (1976) have shown shock instability at conditions predicted to be stable. This suggests that models based on the equilibrium states across the shock are insufficient to predict instability. The present study attempts to address this shortcoming by identifying the physical mechanism leading to the instability

of such shock waves.

1.2 Relaxing shock wave

The media for which instability has been observed share the common feature that energy relaxation occurs *within* the non-equilibrium shock wave structure. Indeed, for ionizing gases, such anomalous instability has only been reproduced numerically by simulating the ionization relaxation process within the non-equilibrium shock structure (Kapper & Cambier, 2011). The starting hypothesis of the present thesis is that one needs to model the relaxation effects in order to reproduce the instability, akin to the treatment of exothermic shock waves (Fickett & Davis, 2000). The current study will thus focus on these non-equilibrium relaxation effects and their potential role in the stability of the shock wave.

Ionization, dissociation or vibrational relaxation are endothermic processes that occur within the shock structure (Zeldovich & Raizer, 1966). These relaxation processes are illustrated in Figure 1.5 in terms of the variation of the translational temperature within the shock structure. In strong shock waves, the first internal modes to equilibrate within the shock structure are the translational and rotational degrees of freedom, leading to the temperature rising significantly within a few mean free paths. If a shock wave is strong enough, the vibrational modes of freedom also become excited. However, this energization of the vibrational modes is much slower than the translational equilibration. As a result, the vibrational modes energize by extracting energy from the translational and rotational modes. The gas is thus undergoing an effectively endothermic process. Ionization and dissociation follow a similar sequence. If the temperature jump caused by the shock is sufficiently high to cause ionization and dissociation, then the energy available in the translational and rotational modes is transferred to modify the structure of the molecules. All of these relaxation processes of ionization, dissociation and the excitation of vibrational degrees of freedom correspond to an interval during which the translational and rotational kinetic energies of molecules are lost through inelastic collisions. It is these inelastic collisions that we wish to investigate and determine if they have any role in the shock wave instability.

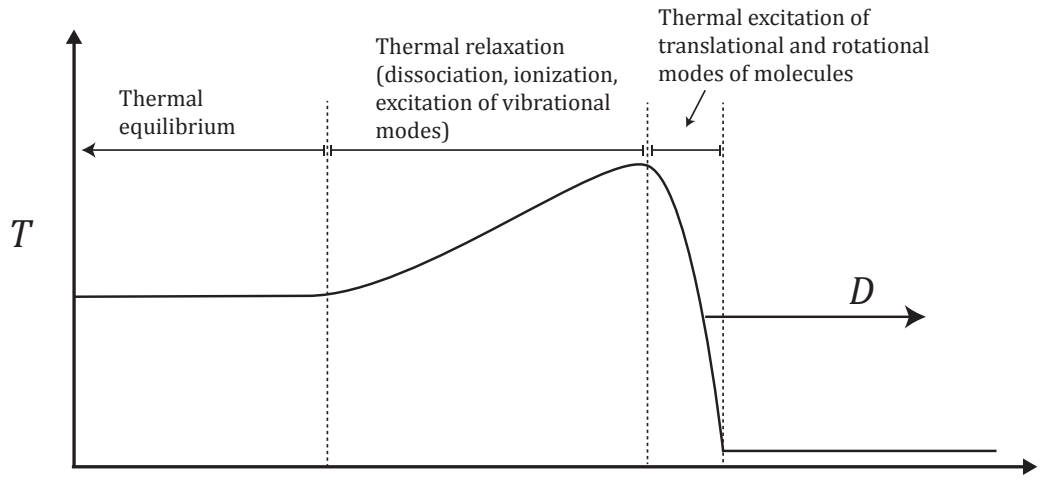


Figure 1.5: Temperature distribution for a thermally relaxing shock wave travelling at velocity D

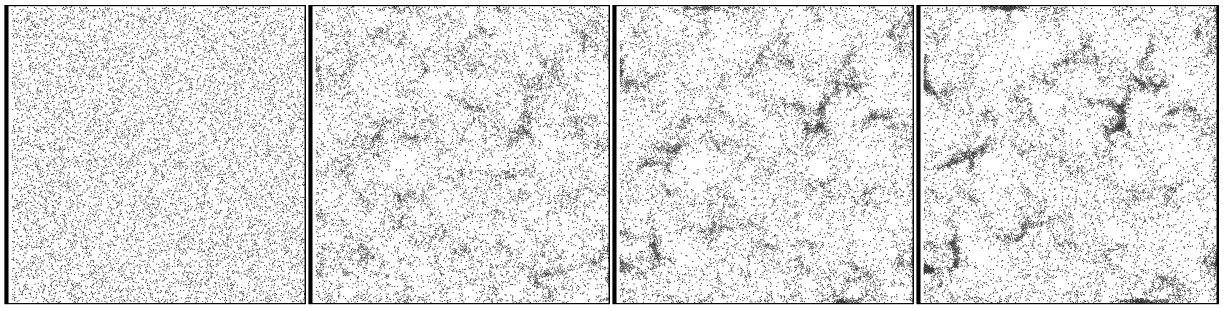
1.3 Granular media

Inelastic collisions are known to induce hydrodynamic instabilities in granular media. In granular media, granular particles dissipate energy due to the inelastic nature of the collisions. The study of granular media may offer additional insight into possible instability mechanisms within relaxing gases.

Granular media can have a gaseous state for which the density is fairly low and there is motion between the particles, for example, in agitated containers. The granular temperature is similar to that of molecular gases, where it is related to the translational kinetic energy of the particles. Granular gases continuously dissipate energy from the inelastic collisions, differing from molecular gases that retain motion of molecules at equilibrium.

Granular gases have shown the presence of instabilities due to inelastic collisions between particles. Goldhirsch & Zanetti (1993) demonstrated a clustering instability caused during the cooling of granular gases. This study was performed using molecular dynamics, where a homogeneous mixture of granular particles is represented in a 2D space by smooth inelastic disks. These particles were prescribed an initial kinetic energy, slowing down over time due to the dissipative collisions. The inelasticity was controlled by defining the coefficient of restitution ε of the collisions. The results demonstrate that once sufficient energy is lost, particles would cluster together, diverging from the initially homogeneous mixture (see Figure 1.6). The time until clustering is found to depend on ε and on the initial fraction of the volume occupied by the particles. It was observed that decreasing

ε (representing greater dissipation per collision) leads to the clustering occurring after a shorter time. Increasing the volume fraction of the particles also decreases the time until the clustering instability. The size of these clusters was found to be dependent on ε, with smaller, denser clusters being associated with lower values of ε.



(a) $t^* = 0$ **(b)** $t^* = 50$ **(c)** $t^* = 100$ **(d)** $t^* = 140$

Figure 1.6: Evolution of clustering in a system of 10,000 hard disks, with $\varepsilon = 0.5$, where t is normalized by the initial mean free time, τ_1

These observations of hydrodynamic instabilities in granular gases are very interesting and may have a direct bearing in the endothermic relaxation processes described above in relaxing shock waves. To the best of our knowledge, this link has never been made before.

Shock wave propagation in granular media has been studied before, although instabilities have not been reported. Goldshtein *et al.* (1996) determined the long-time asymptotic similarity solution for one-dimensional shock waves in granular gases. This structure was reproduced numerically in the continuum Euler description of granular flow by Kamenetsky *et al.* (2000). Both studies identified a unique stable structure for shock waves through one-dimensional granular media, as shown in Figure 1.7 for a piston propagated shock wave. The shock wave in granular gases shown in Figure 1.7 is somewhat artificial, in that the undisturbed region corresponds to a gas-like state with low translational energy. In the example of an agitated container, it corresponds to the stage in which the particles are in free-fall in the container, and the bottom wall strikes the cloud of particles (Swinney & Rericha, 2004). The shock wave causes the granular temperature to rise, forming a denser "fluidized" region behind the shock front. Within the fluidized region, inelastic collisions between moving particles lead to an increase in density and decrease in granular temperature. Eventually, the density is high enough that there is no more movement between particles, characterized as the "frozen" region. These regions are analogous to those for relaxing shock waves in molecular gases (from Figure 1.5), where the "fluidized"

region represents the region containing both the thermal excitation of translational and rotational modes, and the relaxation processes. The "frozen" region, where the granular temperature returns to zero, is similar to the region of thermal equilibrium for relaxing shock waves, although this region retains kinetic energy for molecular gases. This suggests that modelling of relaxing shock waves can be done through a similar approach, adjusting collision properties to account for the thermalized equilibrium region.

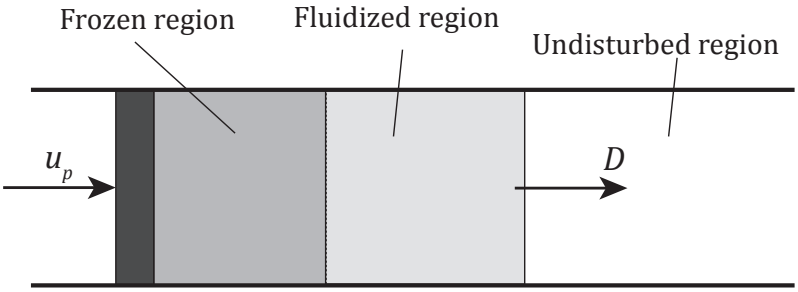


Figure 1.7: Schematic of granular shock structure which can be separated into three sections (undisturbed, fluidized, and frozen regions)

1.4 Present study

Recognizing that inelastic collisions play a dominant role in the relaxing structure of shock waves in relaxing real gases and granular gases alike, we wish to study the effect of such dissipative collisions on the stability of the shock wave structure and determine whether it is linked to the instability of the homogeneous cooling state of granular gases (Goldhirsch & Zanetti, 1993).

The present study thus considers the classical problem of shock wave driven by a moving piston. The approach we use is a hard particle dynamics approach, akin to the Direct Simulation Monte Carlo (DSMC) technique (Bird, 1999). In DSMC, molecules are represented as rigid particles where there is no attractive or repulsive forces acting between the particles during their free flight. Only the collision rules are prescribed in order to capture a physical phenomenon (granular gases, relaxation, chemical reactions, etc...). Much of the kinetic theory of dilute, idealized gases can be obtained by treating molecules as hard spheres with no internal structure, as demonstrated by Chapman & Cowling (1970) and Hirschfelder *et al.* (1964). Some of the first investigations employing hard particle molecular dynamics were completed by Alder & Wainwright (1957, 1962) in order to look at phase transition phenomena.

In order to model the inelastic collisions of relaxing and granular gases, we adopt the simple formalism adopted in granular flows, namely that collisions between particles occur with a constant coefficient of restitution. This is the simplest possible kinetic model that can mimic relaxation. In order to capture the structure of relaxing gases more closely, we also assume that the collisions are activated by an energy threshold, which corresponds to the equilibrium temperature in the gas. In other words, collisions occurring below this threshold will be elastic, and the system stops dissipating energy and stabilizes at that temperature.

The thesis is organized as follows. Chapter 2 presents the details of the numerical technique. The implementation of the numerical algorithm of the hard particle molecular dynamics of Alder & Wainwright (1957) is due to Pöschel & Schwager (2005). Modifications are presented, to allow for moving boundaries and activated collisions.

Chapter 3 considers the shock jump conditions in non-ideal hard particle gases in which the particle co-volumes become important. This work has now been published in Sirmas *et al.* (2012). This chapter is aimed to identify an equation of state for such dense hard disk gases accurate at high densities, but sufficiently simple so it can allow analytical predictions.

In Chapter 4, the structure of relaxing shock waves is considered though molecular dynamic simulations. The effect that the controlling parameters have on the shock instability is investigated. The results for the numerical and theoretical jump conditions are used in order to construct the shock Hugoniot, which is analyzed in order to determine if the modelled instability can be explained from shock theory.

In Chapter 5, the evolution of the simulated shock waves are compared with the evolution of the clustering instability. This is done by calculating the time scales for decay across the simulated unstable shock waves, and comparing with the time for clustering instability to occur in cooling granular gases.

Chapter 2

Model description

This chapter looks at the molecular modelling method that is implemented to simulate piston propagated shock waves. A sketch of the problem is shown in Figure 2.1. As discussed in the Introduction, a hard particle molecular dynamics method in 2D is used. This code was previously modified by Tudorache (2008) to investigate piston propagated shock waves with modifications made in the present study for the collision properties between hard disks. In this chapter, only a brief overview is given for the hard particle model and the algorithm that is employed, with specifics outlined to account for a moving piston and for the activated inelastic collisions. For the complete code in the C++ programming language, see Appendix A.

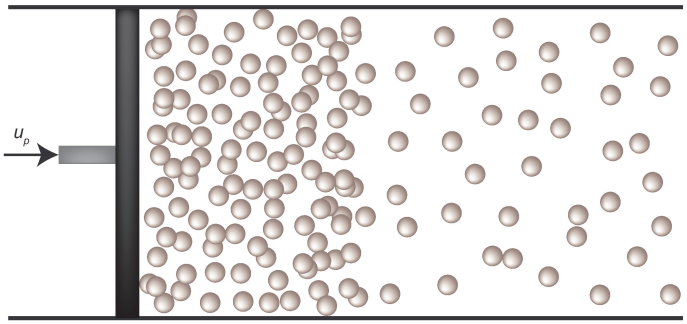


Figure 2.1: Sketch of simulations of piston propagated compression shock through system of particles

2.1 Hard Particle Molecular Dynamics model

In the hard particle molecular dynamics model that is used, molecules are represented as hard disks in free flight within a bounded volume. The hard disks are chosen since there are no force potentials between the molecules, which enables easy calculation of the kinematics of the system. Since there are no external forces or inter-molecular forces, the velocity of the hard particles remain constant unless in contact with a boundary, or another particle. This section looks at the dynamics of collisions for the hard particle molecular dynamics model, where the collision properties are explained for activated, inelastic collisions, which we are interested in modelling in the current study.

2.1.1 Mechanics of collisions

The post-collision velocities $\vec{u_i'}$ are calculated for colliding i particles travelling at velocity $\vec{u_i}$. These collisions are separated into either *boundary collisions* or *pairwise collisions*.

Boundary collisions

Collisions between hard disks and boundaries are elastic, whereby the disks collide specularly with the walls. The post-collision velocities for disks are separated into the tangential velocity component (u_{Ti}), which remains constant, and the normal component (u_{Ni}), which reflects in the opposite direction, i.e.,

$$\begin{aligned} u'_{Ti} =& u_{Ti} \\ u'_{Ni} =& - u_{Ni} \end{aligned} \tag{2.1}$$

Collisions with the piston are also elastic, where the piston is travelling with a velocity u_p and assumed to have an infinite mass. The post-collision velocities are separated into the tangential and normal components, with the piston is propagating into the domain from the negative x-direction with speed u_p. The post-collision velocities are given by:

$$\begin{aligned} u'_{T(i)} =& u'_{yi} = u_{yi} \\ u'_{N(i)} =& u'_{xi} = 2u_p - u_{xi} \end{aligned} \tag{2.2}$$

Pairwise collisions

Figure 2.2 shows a pairwise collision at the point of impact. The pre- and post-collision velocities are broken up into normal and tangential components at the point of impact.

The normal components lie along the *line of action*, connecting the centers of mass of the colliding particles.

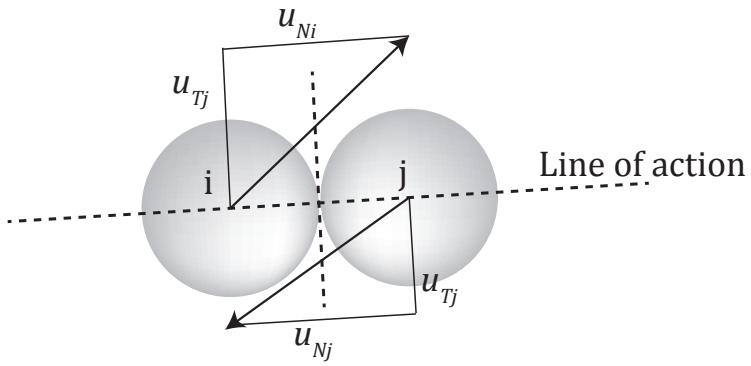


Figure 2.2: Sketch of a pairwise collision between two particle demonstrating the normal and tangential components of velocity with respect to the line of action

The particles are smooth disks, which mean that there is no dissipation due to friction during collisions. This translates to no change in the tangential components during the collision, i.e.,

$$\begin{aligned} u'_{Ti} &= u_{Ti} \\ u'_{Tj} &= u_{Tj} \end{aligned} \tag{2.3}$$

The velocities in the normal direction are found by manipulating the conservation of momentum equation for the collision in the normal direction. To account for the inelasticity, the coefficient of restitution ε^* is used, where $0 \leq \varepsilon^* \leq 1$. This represents the ratio of pre- and post-collision normal velocities:

$$\varepsilon^* = -\frac{\left(u'_{Ni} - u'_{Nj}\right)}{\left(u_{Ni} - u_{Nj}\right)} \quad \text{where} \quad 0 \leq \varepsilon^* \leq 1 \tag{2.4}$$

This gives the following post-collision velocities along the normal direction for disks i and j, which take the form:

$$\begin{aligned} u'_{Ni} &= u_{Ni} - \frac{(1+\varepsilon^*)}{2}\left(u_{Ni} - u_{Nj}\right) \\ u'_{Nj} &= u_{Nj} + \frac{(1+\varepsilon^*)}{2}\left(u_{Ni} - u_{Nj}\right) \end{aligned} \tag{2.5}$$

In order to emulate activated endothermic reactions for the current study, inelastic collisions with $\varepsilon^* < 1$ only occur when the relative impact velocity exceeds a threshold, u_{max}. All other pairwise collisions are elastic, i.e.,

$$\varepsilon^* = \begin{cases} 1 & \text{if} \quad |u_{Ni} - u_{Nj}| < u_{max} \\ \varepsilon & \text{if} \quad |u_{Ni} - u_{Nj}| \geq u_{max} \end{cases}$$

where the predefined u_{max} and ε remain constant during each simulation.

2.2 Model implementation

The hard particle molecular dynamics model using the collision properties discussed in the previous section was implemented by using Event Driven Molecular Dynamics (EDMD) method (Alder & Wainwright, 1957). The model used in the current study follows the algorithm as presented by Pöschel & Schwager (2005). For the *event-driven* algorithm, the simulation propagates to the next time step where an event occurs, which is either a collision with a boundary, or a pairwise collision between disks. In contrast, if there were forces acting on the molecules (force-based algorithms), then small time steps would have to be taken to account for the change in acceleration of molecules, resulting in significantly longer computational times and EDMD could not be easily implemented.

2.2.1 EDMD algorithm

The EDMD algorithm follows a simple set of steps to advance the system of particles in time. More efficient algorithms to what is presented here can be seen in Pöschel & Schwager (2005) and are implemented in the code used for this study. A simplified EDMD algorithm to investigate piston propagated shock waves through hard particles is demonstrated by the following set of steps:

1. The system is initialized ($t = 0$) with N hard disks at their respective positions (x_{io}, y_{io}) and velocities (u_{xi}, u_{yi})

2. Schedule of events are computed by calculating the collision times for all pairwise collisions and when each hard disk will collide with a boundary, with the length of time to the nearest collision time denoted as t^* (see Figure 2.3 for a simple schematic of this step)

3. The $i = 1, 2...N$ particles are propagated to their new positions (x_i, y_i) from position (x_{io}, y_{io}):

$$\begin{aligned} x_i &= x_{io} + u_{xi}t^* \\ y_i &= y_{io} + u_{yi}t^* \end{aligned} \tag{2.6}$$

   Piston travelling at velocity u_p is propagated to position x_p:

$$x_p = x_{po} + u_p t^* \tag{2.7}$$

4. Post collision velocities u'_{xi} and u'_{yi} calculated for boundary collisions, and u'_{xi}, u'_{yi}, u'_{xj}, u'_{yj} calculated for pairwise collisions between i and j. These velocities are updated in the algorithm.

5. Time is set to $t = t + t^*$ and positions set to $x_{io} = x_i$, $y_{io} = y_i$ for $i = 1, 2...N$

6. Return to Step 2.

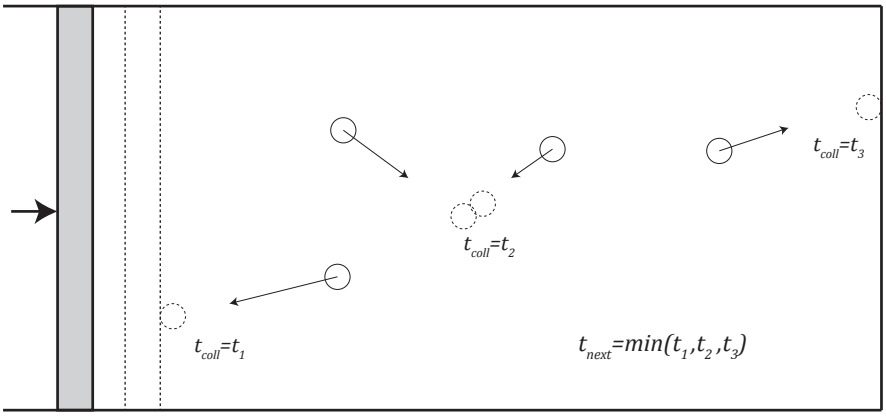


Figure 2.3: Sketch demonstrating the scheduling used for the Event Driven Molecular Dynamics algorithm, calculating the minimum time to a collision within a set of disks

In order to follow this algorithm we must calculate the time between collisions and the properties of collisions, which are explained in the following sections.

2.2.2 Kinematics of particles and boundaries

The novelty of the EDMD algorithm is the scheduling of collision times for all possible collisions. This allows for the propagation of the system to the next collision, thus avoiding the need to integrate over discrete, small time steps.

Since there are no forces acting on the molecules, the equations of motion are easily expressed for the hard disks and moving boundaries (piston). This allows for the time between events to be easily calculated. The trajectories for the centre of each hard disk can be expressed in terms of the velocities (u_y, u_x) and the position at the time of calculation (x_o, y_o):

$$\begin{aligned} x_i(t) &= x_{io} + u_{xi}t \\ y_i(t) &= y_{io} + u_{yi}t \end{aligned} \tag{2.8}$$

Pairwise collisions between two hard disks with radius R occur when their centres are located at a distance of $2R$ from each other. A collision occurs between an i and j hard disk at the time calculated by the following:

$$d_{ij} = \sqrt{(x_i(t^*_{ij}) - x_j(t^*_{ij}))^2 + (y_i(t^*_{ij}) - y_j(t^*_{ij}))^2} = 2R \tag{2.9}$$

Since the velocities and positions are known for each hard disk at the time of calculations, the time of collision t^*_{ij} can be calculated. For a calculated $t^*_{ij} < 0$, no collision between hard disk i and j will occur without other collisions occurring first. For $i, j = 1, 2.....N$, the time for all possible pairwise collisions is calculated in order to determine which event occurs first.

Similarly, collisions times between hard disks and the boundaries are calculated. In this case, the collisions can occur with the walls if the centres of the hard disks are a distance R away from the boundary. The boundaries of the system are three stationary walls at the upper (y_{upper}), lower (y_{lower}) and right (x_{right}) boundaries, and a moving boundary from the left representing a piston at the time dependent position $x_p(t)$. The piston propagates at constant velocity, u_p, such that the position of the piston at any time is:

$$x_p(t) = x_{po} + u_p t \tag{2.10}$$

The time until collisions with the boundaries are separated into the time to collide with the surfaces in the y-direction (t^*_{iy}) and time to collide with surfaces in the x-direction (t^*_{ix}). For $v_{yi} > 0$ the collision will occur with the upper wall at time $t^*_{upper(i)}$, and for $u_{yi} < 0$ the collision will occur with the lower wall at $t^*_{lower(i)}$,

$$\begin{aligned} t^*_{upper(i)} &= \frac{y_{upper} - y_{io} - R}{u_{yi}} \\ t^*_{lower(i)} &= \frac{y_{lower} - y_{io} + R}{u_{yi}} \end{aligned} \tag{2.11}$$

Collisions in the x-direction depend on the direction of a hard disk trajectories, and the relative velocity to the piston. If $u_{xi} < 0$, a collision with the piston will occur at

$t^*_{ix} = t^*_{piston(i)}$. If $u_{xi} > 0$ the time to collide with piston will be the minimum time between a collision with the right wall or with the piston, i.e., $t^*_{ix} = \min(t^*_{piston(i)}, t^*_{right(i)})$. These times are calculated as:

$$\begin{aligned} t^*_{piston(i)} &= \frac{x_{po} - x_{io} + R}{u_p - u_{xi}} \\ t^*_{right(i)} &= \frac{x_{right} - x_{io} - R}{u_{xi}} \end{aligned} \tag{2.12}$$

Therefore, hard disk i will collide with the boundaries at the minimum value between t^*_{ix}, t^*_{iy}, denoted as the time $t^*_{wall(i)}$.

Calculating t^*_{ij} and $t^*_{wall(i)}$ for $i, j = 1, 2...N$ allows for a schedule of all events to be compiled.

2.3 Initialization of domain and simulation

At the onset of each simulation, N identical hard disks are initialized with equal initial velocities ($u_{rms(1)}$) and randomized directions. The spacing between molecules is controlled by the packing factor η, which is the fraction of the volume occupied by the hard disks, i.e.,

$$\eta = \frac{N\pi R^2}{A} \tag{2.13}$$

By defining η and the initial domain area, $A = L_x \times L_y$, the radius R of the hard disks is fixed.

The hard disks collided completely elastically for a given time until the system reached an equilibrium state, defined by the Maxwellian velocity distribution. An interval of approximately 10 mean free times was found to be a sufficient duration for this (Tudorache, 2008). Once the gas reached its equilibrium state, the piston is rapidly accelerated from rest to a constant speed u_p.

2.4 Scaling of simulations

In the simulations, the macroscopic properties of temperature T, density ρ, hydrostatic pressure p, and particle velocity u, are scaled by the initial, undisturbed states of these

properties:

$$\begin{aligned} T^* &= \frac{T}{T_1} \\ \rho^* &= \frac{\rho}{\rho_1} = \frac{n}{n_1} \\ p^* &= \frac{p}{\rho_1 Z_1 \frac{k}{m} T_1} \\ u^* &= \frac{u}{u_{rms(1)}} = \frac{u}{\sqrt{2kT_1/m}} \end{aligned} \tag{2.14}$$

where n is the number density of molecules, Z is the compressibility factor, k is the Boltzmann constant, and m is the mass of a particle.

The lengths are scaled by the initial mean free path λ_1 (Brilliantov & Pöschel, 2004):

$$x^* = \frac{x}{\lambda_1} \tag{2.15}$$

where

$$\lambda_1 = \frac{1}{2\sqrt{2\pi} n_1 R g(\eta_1)} \tag{2.16}$$

$g(\eta)$ is the Enskog factor for 2D system of hard particles, dependent on the packing factor η,

$$g(\eta) = \frac{1 - \frac{7\eta}{16}}{(1-\eta)^2} \tag{2.17}$$

The time is normalized by $\tau_1 = \lambda_1 / u_{rms(1)}$, which is the initial mean free time between collisions:

$$\tau_1 = \frac{1}{2\sqrt{2\pi} n_1 R g(\eta_1) u_{rms(1)}} \tag{2.18}$$

2.5 Data collection

Throughout the simulations, the velocities and positions of every disk are tracked, capturing the macroscopic state of the system. Data is output after a specified time step, or distance travelled by the piston. At each of these times, the model outputs various files recording the state of the system.

First, a scalar vector graphic (.svg) image file is output, displaying the position of all the particles and the evolution of the shock waves. The background of Figure 2.4 shows an example of this output, where the distribution of particles is shown, along with the position of the piston at that time.

The macroscopic properties of the flow are found by coarse grain averaging the positions and velocities of the particles. This is done by separating the domain into vertical strips of width dx extending the height of the domain, or into boxes of size $dx \times dy$, as shown in Figure 2.4. The one-dimensional distribution of macroscopic properties are found by coarse grain averaging the microscopic properties within the strips, which are x- and y-directions are collected within the square boxes with sides approximately equal to $2\lambda_1$.

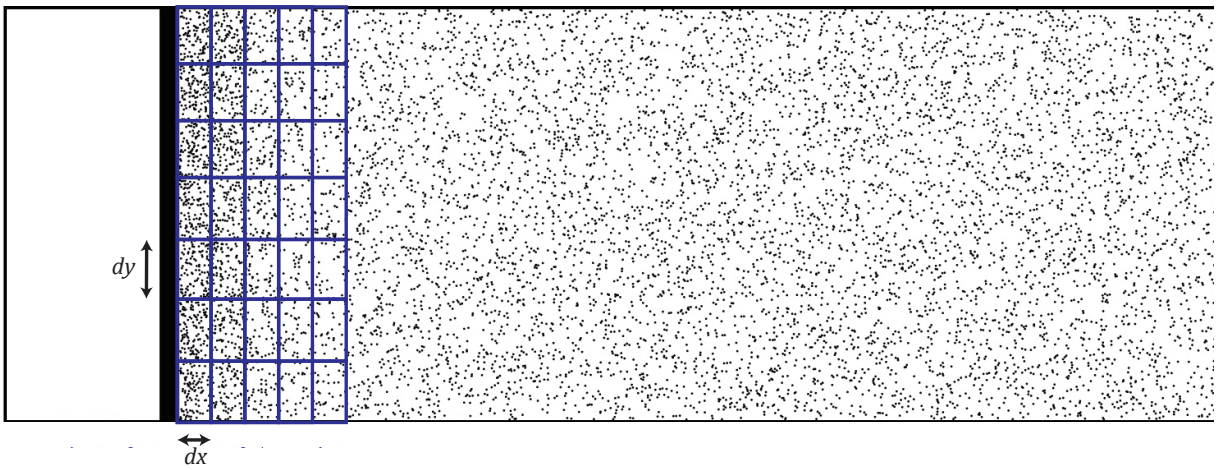


Figure 2.4: Example of particle distribution in EDMD simulation with a sample grid used for coarse-grain averaging

The local temperature $T(x)$ of each strip or box is found by averaging the internal energies in the x- and y-directions over all of the particles within the grid. In the x-direction, the mean velocity of the macroscopic motion is subtracted to account for the flow velocity of the moving piston. The local temperature at position x, bounded by the strip of width dx is written as:

$$\begin{aligned} T(x) =& \frac{T_x + T_y}{2} \\ T_y(x) =& \frac{1}{N(x)} \sum_{i=1}^{N(x)} \frac{\frac{1}{2} u_{yi}^2}{k/m} \\ T_x(x) =& \frac{1}{N(x)} \sum_{i=1}^{N(x)} \frac{\frac{1}{2} (u_{xi} - \langle u_x \rangle)^2}{k/m} \end{aligned} \tag{2.19}$$

Where $N(x)$ is the number of hard disks at position x, bounded by the strip of width dx, and $\langle u_x \rangle = \frac{1}{N(x)} \sum_{i=1}^{N(x)} u_{xi}$ is the local mean velocity in the x-direction.

The local number density in each strip, $n(x)$ is written as:

$$n(x) = \frac{N(x)}{dx \times L_y} \tag{2.20}$$

Ensemble averaging is used to obtain accurate measurements of the coarse-grained density and temperature. This is done by repeating the calculation with statistically different initial conditions, which is implemented by randomizing the initial positions and directions of motion for each simulation.

Chapter 3

Shock waves in elastic dense media

Before looking at inelastic collisions in shock waves, we first look at shock waves through completely elastic hard disks. We determine the shock jump conditions and compare them to the numerical results from the simulations. Observing the elastic shock structure allows for comparison of the stability based on whether the collisions are elastic or inelastic in the subsequent chapters. Additional details to the work presented in this chapter can be found in Sirmas *et al.* (2012).

3.1 Continuum description of elastic hard disks

3.1.1 Equation of state for a hard disk gas

In hard particle systems, there is no internal structure, therefore the internal energy e consists of only the translational kinetic energy. The energy for each translational degree of freedom is $\frac{1}{2}\frac{k}{m}T$. For a hard disk system, there are two translational degrees of freedom, giving the internal energy as,

$$e = \frac{k}{m}T \tag{3.1}$$

An equation of state is needed to relate the hydrostatic pressure p, the specific volume v and the temperature T, written as (Mulero *et al.*, 2008):

$$pv = Z\frac{k}{m}T \tag{3.2}$$

where Z is the compressibility factor, which tends to 1 for an ideal gas.

Combining (3.1) and (3.2), the internal energy e can be written in terms of p and v:

$$e = \frac{pv}{Z} \tag{3.3}$$

The compressibility factor Z is usually expressed in terms of the packing factor η. The caloric equation of state can then be expressed as:

$$e(p, v) = \frac{pv}{Z(\eta)} \tag{3.4}$$

For a hard-disk medium, the exact equation of state is not known in closed form, requiring an infinite virial expansion. Mulero *et al.* (2008) demonstrated the various equations of state proposed for a hard particle system. One equation of state proposed in hard disks is that by Helfand *et al.* (1961), which is accurate, physically meaningful and analytically simple. This simple expression takes the form

$$Z(\eta) = \frac{1}{(1-\eta)^2} \tag{3.5}$$

This equation of state shall be used to analytically investigate the effect that the initial compaction η_1 has on the medium's compressibility.

3.1.2 Isentropic exponent and sound speed

The change in the compressibility of medium is best described by the isentropic exponent γ, which describes the relation between changes in density and changes in pressure for isentropic processes. It is defined as

$$\gamma \equiv \left(\frac{\partial \ln p}{\partial \ln \rho}\right)_s \tag{3.6}$$

The sound speed in the medium is directly related to γ, i.e.:

$$c^2 \equiv \left(\frac{\partial p}{\partial \rho}\right)_s = \gamma \frac{p}{\rho} \tag{3.7}$$

By using the Helfand equation of state, we are able to express the sound speed in terms of the local state of the medium and the local packing factor η

$$c^2 = pv\left(1 + (1-\eta)^{-2} + 2\eta(1-\eta)^{-1}\right) \tag{3.8}$$

and from (3.7) we get the isentropic exponent.

$$\gamma = 1 + (1-\eta)^{-2} + 2\eta(1-\eta)^{-1} \tag{3.9}$$

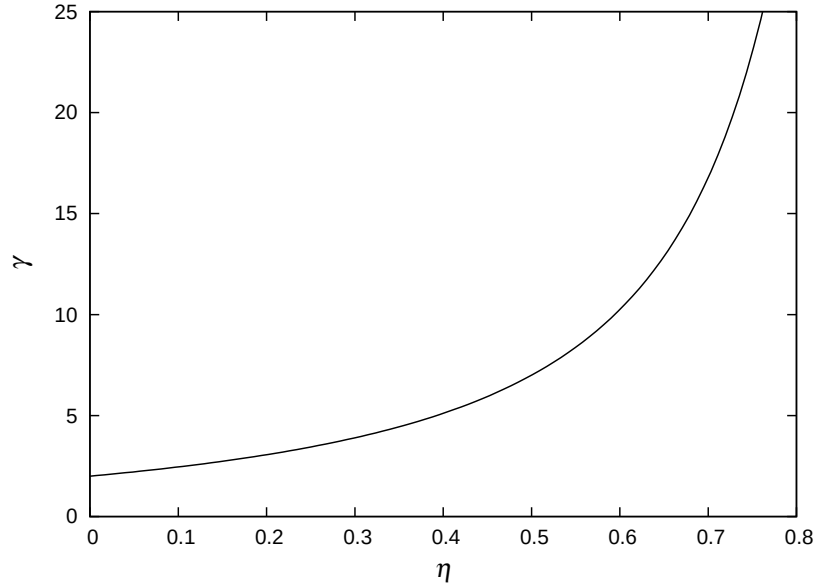


Figure 3.1: The variation of the isentropic exponent with the local packing factor η

Figure 3.1 shows the variation of the isentropic exponent with the local packing fraction. In the dilute limit of an ideal gas, i.e., $\eta \to 0$, we recover the isentropic exponent of a 2-dimensional hard disk gas of $\gamma = 2$. However, with increasing packing fraction, the isentropic exponent grows significantly, reflecting the incompressibility of the medium. A small change in density requires a very large change in pressure, a characteristic of nearly *incompressible* media like liquids and solids.

3.1.3 Shock Waves in a hard disk medium

To look at shock waves, we consider the classic problem where a piston is suddenly accelerated from zero velocity to a constant velocity u_p into a medium initially at rest, as shown in Figure 3.2. A shock wave is formed, which propagates with velocity D. If diffusive fluxes are negligible in the pre- and post-shock states, and letting the subscript 2 denote the uniform state of the medium behind the shock wave, the conservation of mass, linear momentum and energy, in the frame of reference of the shock (see Figure 3.2) yield (Zeldovich & Raizer, 1966):

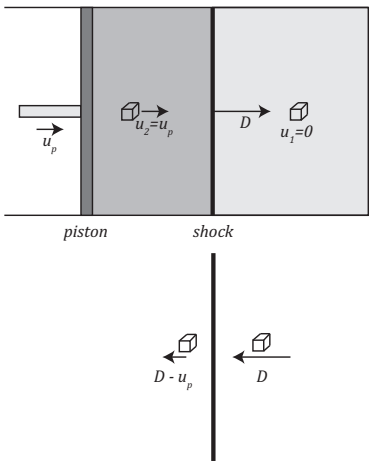


Figure 3.2: Propagation of a piston driven shock wave into a quiescent medium and transformation to the shock fixed frame of reference (lower)

$$\frac{D}{v_1} = \frac{(D-u_p)}{v_2} \tag{3.10}$$

$$p_1 + \frac{D^2}{v_1} = p_2 + \frac{(D-u_p)^2}{v_2} \tag{3.11}$$

$$e_1 + p_1 v_1 + \frac{1}{2}D^2 = e_2 + p_2 v_2 + \frac{1}{2}(D-u_p)^2 \tag{3.12}$$

Combining the equation of mass (3.10), momentum (3.11) and energy (3.12) to eliminate the two speeds, the accessible end states 2 across the shock wave can be represented by the shock Hugoniot relation in the (p, v) plane (Zeldovich & Raizer, 1966):

$$e_2(p_2, v_2) - e_1(p_1, v_1) = \frac{1}{2}(v_1 - v_2)(p_1 + p_2) \tag{3.13}$$

This shock description is completely general, and depends on the equation of state $e(p, v)$ characterizing the particular medium.

Using Helfand's equation of state (3.5) and relations (2.13) and (3.4), the Hugoniot relation (3.13) can be used to express in closed form the variation of pressure with specific volume across the shock wave, yielding

$$\pi = \frac{\frac{1}{2}(1-\sigma) + (1-\eta_1)^2}{\sigma(1-\frac{\eta_1}{\sigma})^2 - \frac{1}{2}(1-\sigma)} \tag{3.14}$$

where π, σ and η_1 are, respectively, the non-dimensional pressure ratio across the shock wave, non-dimensional specific volume ratio across the shock wave, and the initial packing

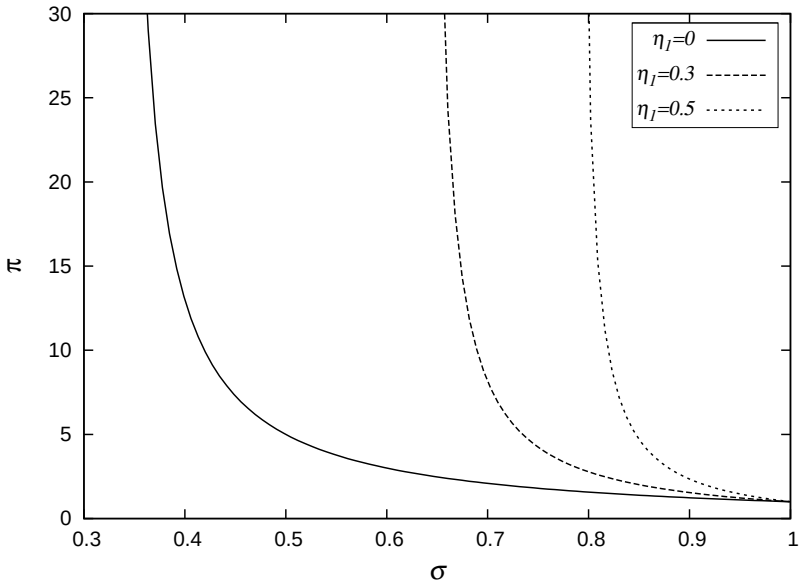


Figure 3.3: The relation between the shock overpressure and compression ratio represented on the shock Hugoniot for a hard disk medium in terms of the initial packing factor η_1

fraction of the medium.

$$\pi = \frac{p_2}{p_1}, \sigma = \frac{v_2}{v_1}, \eta_1 = \frac{(V_a/m)}{v_1} \tag{3.15}$$

Figure 3.3 displays the shock Hugoniot curves for three initial packing fractions. As the packing fraction is increased, a higher pressure is generated for the same compression ratio. Alternatively, the same pressure is achieved with less compression in an initially higher packed medium. This mimics very well the real properties of shocks in less compressible media such as liquids and solids (Zeldovich & Raizer, 1966). The maximum compression that can be achieved across a shock wave can be found by letting $\pi \to \infty$ in (3.14). This corresponds to letting the denominator in (3.14) equate to zero, yielding a quadratic for which the largest root is the physically attainable solution. The maximum compression achievable is thus

$$\sigma_{max} = \frac{1}{6}(1 + 4\eta_1 + \sqrt{(1 - 8(\eta_1 - 1)\eta_1}) \tag{3.16}$$

which is shown in Figure 3.4.

Next we parametrize the shock jump conditions based on the Mach number of the shock wave M_1. From the mass (3.10) and momentum (3.11) equations and using (2.13), (3.4), (3.5) and (3.15), we can express the overpressure across the shock wave in terms of the shock Mach number and compression ratio across the shock, yielding the so-called

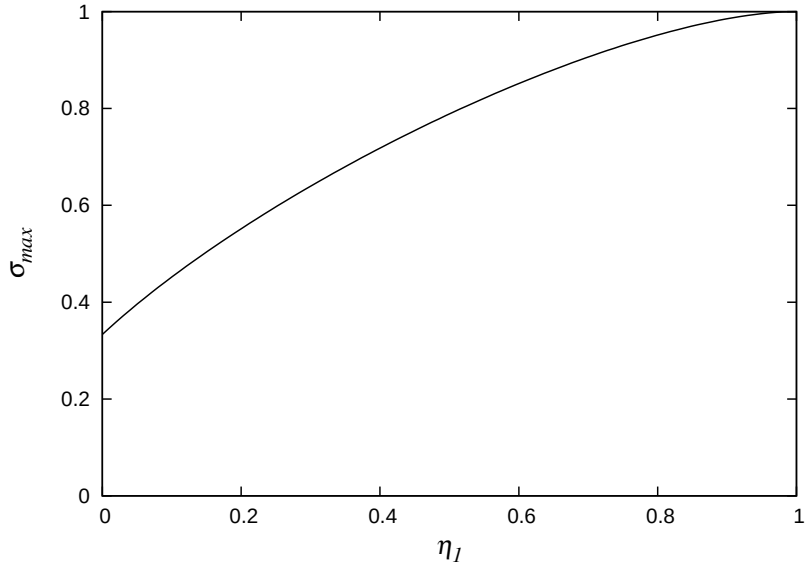


Figure 3.4: The maximum compression ratio achieved by shock compression in terms of the initial packing factor η_1

Rayleigh line

$$\pi = 1 + \psi(1 - \sigma) \tag{3.17}$$

where ψ is simply related to the shock Mach number and the initial value of γ:

$$\psi \equiv \frac{D^2}{p_1 v_1} = \gamma_1 \frac{D^2}{{c_1}^2} = \gamma_1 {M_1}^2 \tag{3.18}$$

Equating (3.14) to (3.17), we obtain the shock jump conditions in closed form in terms of the shock strength parameter ψ, and the initial packing fraction η_1:

$$\frac{v_2}{v_1} = \sigma = \frac{4 + \psi + 4\psi + 4\psi\eta_1}{6\psi} + \frac{\sqrt{-24\psi(1+\psi)\eta_1^2 + (4 + \psi + 4\psi\eta_1)^2}}{6\psi} \tag{3.19a}$$

$$\frac{p_2}{p_1} = \pi = 1 + \psi(1 - \sigma) \tag{3.19b}$$

$$\frac{u_p}{D} = 1 - \sigma \tag{3.19c}$$

$$\frac{T_2}{T_1} = \pi\sigma \frac{1 - \frac{{\eta_1}^2}{\sigma}}{(1 - \eta_1)^2} \tag{3.19d}$$

In order to better compare with the simulated results, we parametrize this solution in terms of the piston velocity, instead of the Mach number. To do this, we use the square

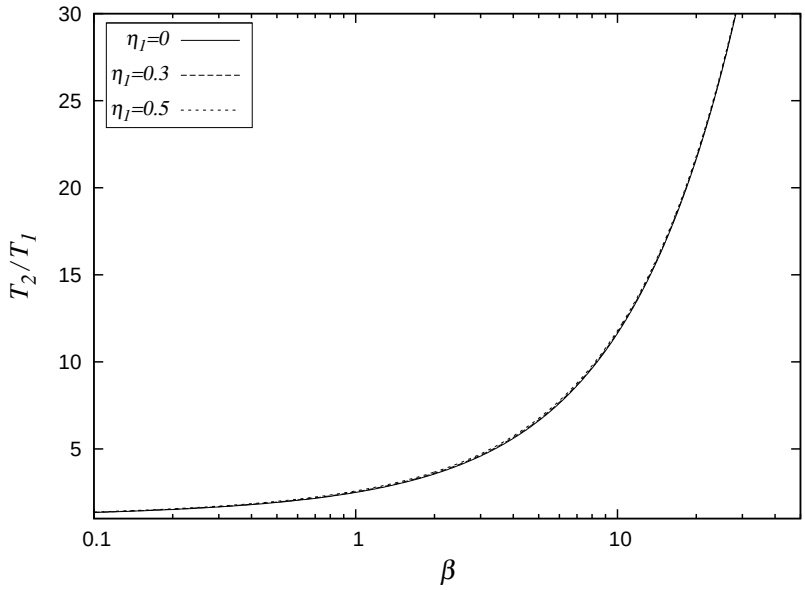


Figure 3.5: Analytical results for the variation of the temperature ratio with β for different initial packing factors

of the piston speed, representing the kinetic energy addition by the piston motion as a parameter, given by

$$\beta \equiv \frac{1}{2}\frac{{u_p}^2}{e_1} \equiv \frac{{u_p}^2}{u_{rms(1)^2}} \tag{3.20}$$

Expressions (3.19) can be expressed implicitly in terms of β using (3.19c), which can be rewritten as

$$\beta = \frac{\psi}{2}\frac{(1-\sigma)^2}{(1-\eta_1)^2} \tag{3.21}$$

Even in the closely packed regimes, the temperature ratio is very well approximated by the ideal gas expression, even for the weak shocks, as shown in Figure 3.5. To account for this finding, the temperature ratio can be written directly from the Hugoniot relation (3.13) which, after some manipulation, can be rewritten as

$$\frac{T_2}{T_1} = \frac{e_2}{e_1} = 1 + \frac{1-\sigma}{(1-\eta_1)^2} + \frac{1}{2}\frac{{u_p}^2}{e_1} \tag{3.22}$$

The second term on the right hand side of (3.22), labeled A, is bounded as $u_p \to \infty$, while the third term, labeled B, grows without bounds as $u_p \to \infty$, hence dominates the temperature increase across the shock wave. Formally taking the ratio of these two terms, we obtain

$$\frac{A}{B} = \frac{2}{(1-\sigma)\gamma M^2} \tag{3.23}$$

In the limit of a large shock Mach number, $\sigma \to \sigma_{max}$ (from (3.16)) and hence $A/B \to 0$. Likewise, this term also vanishes for arbitrary shock Mach numbers when the initial compaction ratio is increased. Indeed, the isentropic exponent increases very rapidly with packing factor, as shown in Figure 3.1. Under these conditions, the internal energy and temperature, according to (3.22), are thus incremented by the kinetic energy of the piston, resulting into an equipartition of the available energy into the mean motion (mean kinetic energy of the medium) and thermal energy.

$$\frac{T_2}{T_1} = \frac{e_2}{e_1} \cong 1 + \frac{1}{2}\frac{{u_p}^2}{e_1} \tag{3.24}$$

This result was also obtained by Zeldovich & Raizer (1966) by neglecting the upstream pressure of an arbitrary medium in the governing equations. Instead, the condition that the term given in (3.23) be small can be considered as the strong shock condition in a hard disk fluid. Generally, it was numerically found that the correction for low Mach numbers is very weak, suggesting that (3.24) serves as a very good approximation for all initial compaction ratios of the medium, ranging from the dilute ideal gas to a solid packed medium of hard particles.

3.2 Simulated results for shock waves in elastic hard disks

In order to validate the results obtained analytically in the previous section, we perform a series of molecular dynamic simulations. The initialization and progression of the simulation, as well as the collection of macroscopic properties follows the description in Chapter 2. Since we are only looking at elastic collision in this chapter, we set $\varepsilon = 1$.

An example of a piston propagated shock wave through elastic hard disks is shown in Figure 3.6. As seen from the figures, an increase in the density of particles is easily identifiable, representing a planar shock propagating across the system of hard disks.

We vary η_1 in order to look at the effect that different initial volume fractions have on the shock jump conditions. Three initial conditions are investigated, corresponding to a dilute gas ($\eta_1 = 0.012$), dense gas ($\eta_1 = 0.192$) and much denser liquid state ($\eta_1 = 0.433$). These conditions are summarized in Table 3.1, in addition to the particle radius and domain size normalized by the initial mean free path λ_1. For each packing fraction, the speed of the piston was varied in order to achieve different levels of compression. In all simulations, the density and temperature were obtained by coarse-grain and ensemble averaging of the state of the gas, as described in Chapter 2. An example of the density

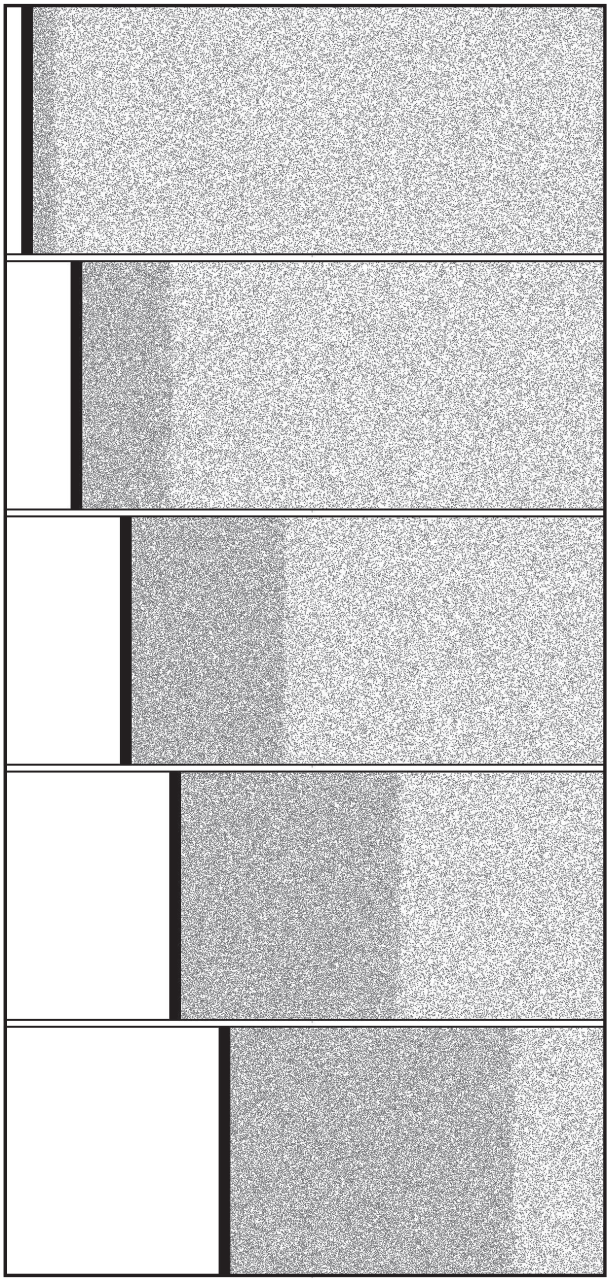


Figure 3.6: Five consecutive snapshots illustrating the shock wave driven by the moving piston for $\beta = 6.25$ and $\eta_1 = 0.192$

Table 3.1: Parameters for the three fluid regimes investigated via EDMD, where the length scales are normalized by the initial mean free path λ_1

Packing fraction η_1	Particle radius R	Domain size $L_x \times L_y$
0.012	0.019	38.3×19.5
0.192	0.43	211.5×107.9
0.433	1.74	569.0×290.6

profile captured during a single simulation is shown in Figure 3.7. The ensemble average of 25 of these realizations for density and temperature profiles are shown in Figures 3.8 and 3.9. As can be seen, the density and temperature relax slowly to their equilibrium values behind the shock jump. The results reported below provide the equilibrated values across the shock wave. In order to determine the shock speed, we track the position of the shock wave over different time steps, tracking where the density increases by 50% of its shocked state.

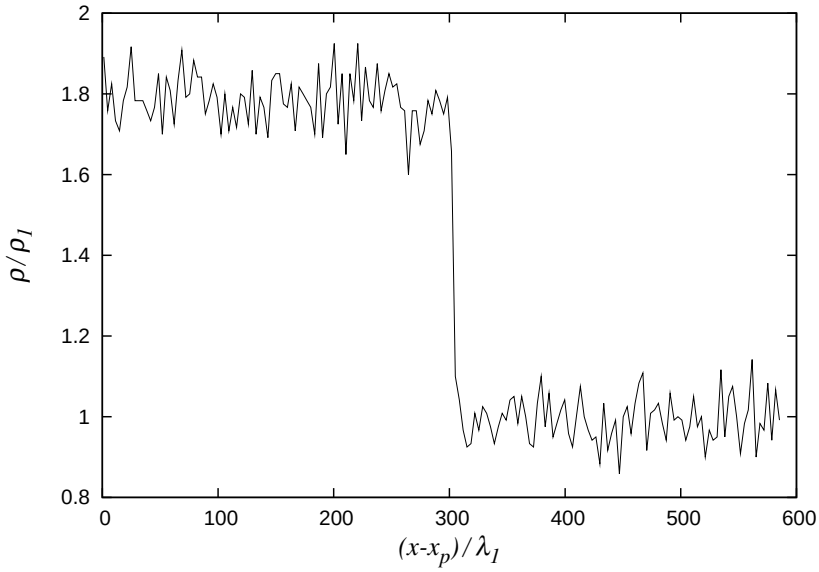


Figure 3.7: The density profile captured for a single realization for $\beta = 16$ and $\eta_1 = 0.192$

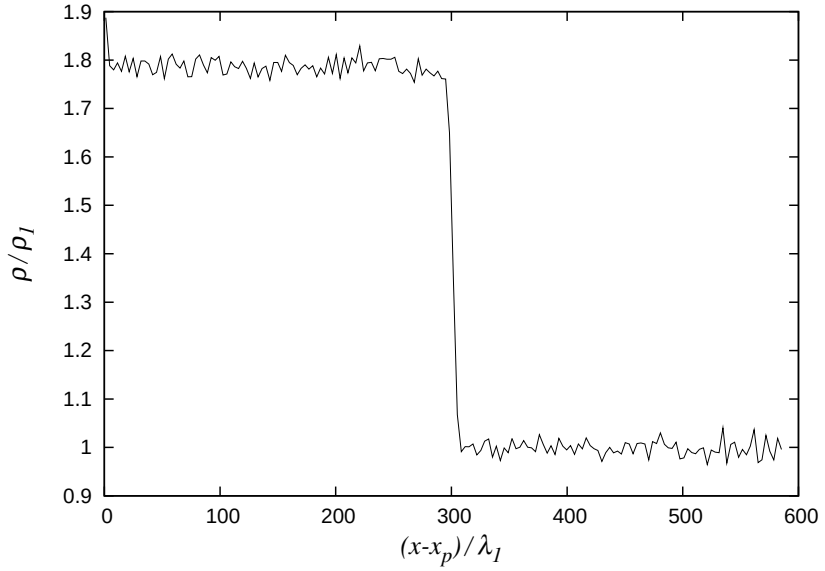


Figure 3.8: The average density profile obtained for $\beta = 16$ and $\eta_1 = 0.192$

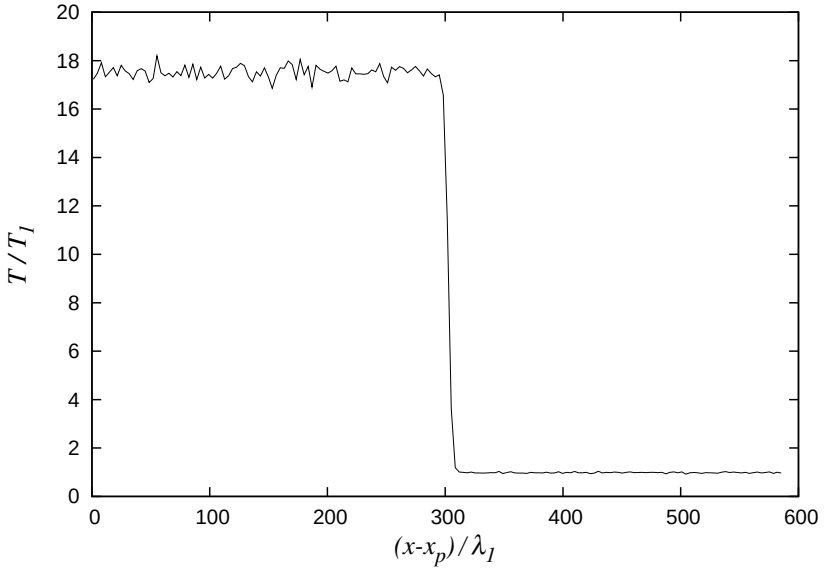


Figure 3.9: The average temperature profile obtained for $\beta = 16$ and $\eta_1 = 0.192$

Figures 3.10, 3.11 and 3.12 show the results obtained for the density jump across the shock wave, shock wave speed and temperature for different initial compaction ratios η_1 and piston speeds u_p. Also shown in these figures are the analytical predictions using the Helfand equation of state detailed in the previous section. In all cases, the agreement was

found to be excellent, with an error not greater than $\sim 3\%$, even for the large compaction ratio and strong shocks investigated. The temperature jump across the shock wave displayed in Figure 3.12 shows the predicted invariance with the compaction ratio given by (3.24). Figure 3.13 shows the results obtained for the overpressure and compression ratio for different initial compaction ratios η_1 and piston speeds u_p. These results are also in good agreement with the analytical predictions for the Hugoniots, which are also shown in this Figure 3.13.

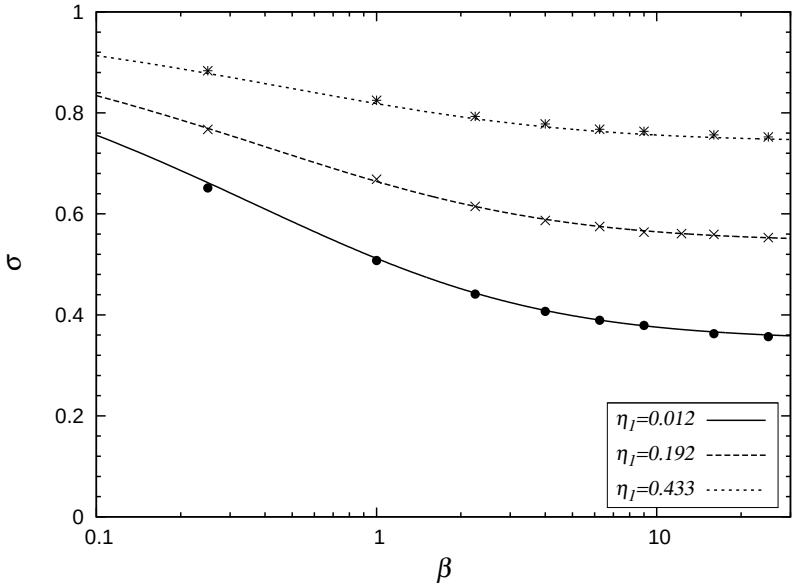


Figure 3.10: Numerical and analytical results for the variation of the compression ratio with β for different initial packing factors

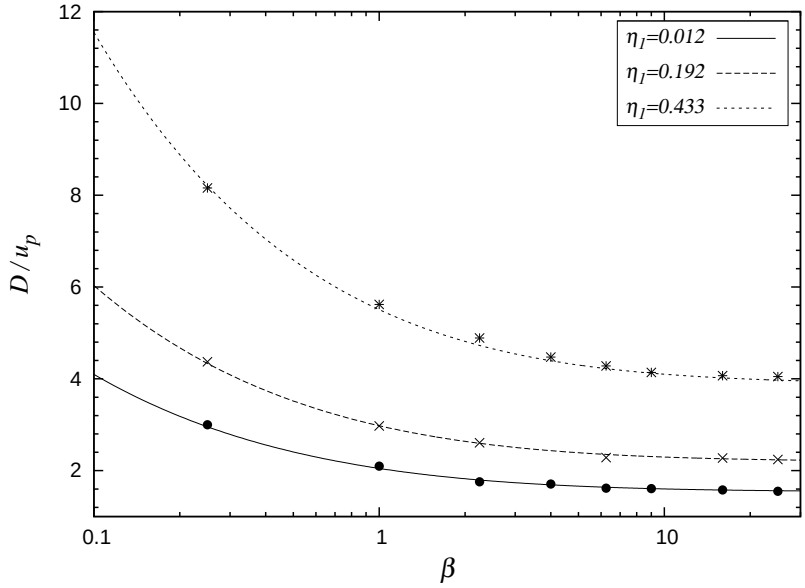


Figure 3.11: Numerical and analytical results for the variation of the shock velocity with β for different initial packing factors

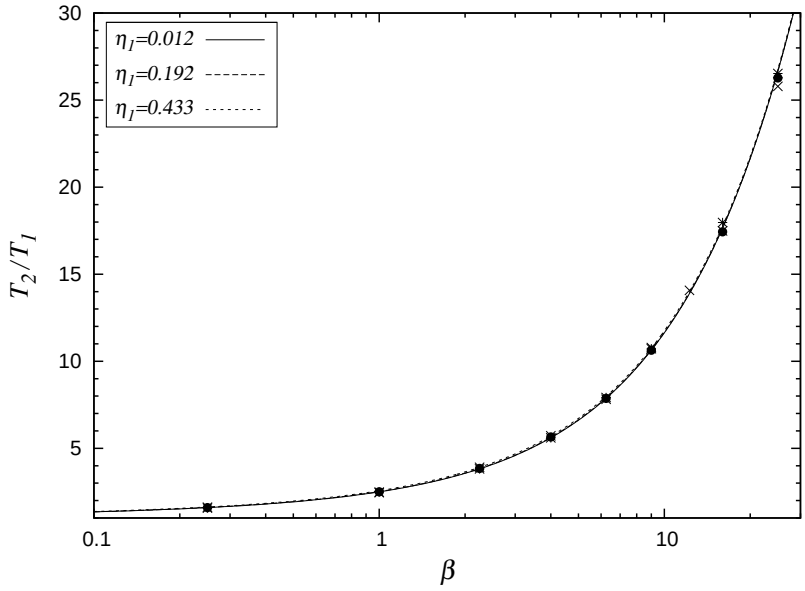


Figure 3.12: Numerical and analytical results for the variation of the temperature ratio with β for different initial packing factors

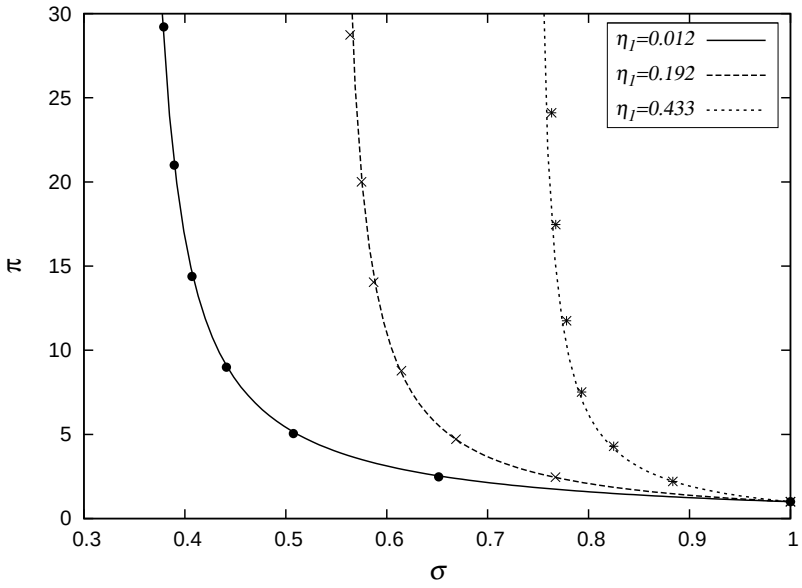


Figure 3.13: Numerical and analytical results for the state Hugoniot for different initial packing factors

In this chapter, the numerical results for the shock structure and distribution of macroscopic properties demonstrate the formation of a stable shock wave. The compressible dynamics and shock wave propagation in a dense hard disk medium were derived using the Helfand equation of state, which was validated against molecular dynamics calculations using the Event Driven Molecular Dynamics technique. In the next chapter, we investigate the departure from this elastic behaviour, where activated collisions will be inelastic, and investigate the effect they have on the instability.

Chapter 4

Molecular dynamics results for shock waves in dissipative media

In the preceding chapter we looked at shock waves in a system of elastic hard disks, deriving the jump conditions and demonstrating the formation of stable planar shock waves. We now wish to look at simulated shock waves where these disks undergo inelastic collisions. As described in Chapter 2, the collisions between hard disks are elastic unless their impact velocities exceed a specified threshold, in order to imitate activated endothermic processes. Since instabilities in the dilute gas regime are the primary focus, we only look at $\eta_1 = 0.012$. This also allows us to compare the results with those found in Chapter 3 for similar packing factor and no dissipation. In this chapter we look at the effect that shock strength, activation energy, and rate of dissipation have on the stability, which is investigated by varying the parameters of piston velocity u_p, activation threshold u_{max}, and the coefficient of restitution ε. The properties for the domain size and particle radius are summarized in Table 4.1, with the lengths normalized by the initial mean free path λ_1. The size of the domain used for the simulations, with 30,000 particles, was found to be an appropriate size to investigate and capture instability, allowing for sufficiently fast computing in order for results to be ensemble averaged.

Table 4.1: Parameters used for piston propagated shock waves through inelastic disks via EDMD, where the length scales are normalized by the initial mean free path λ_1

Packing factor η	Number of particles N	Particle radius R	Domain size $L_x \times L_y$
0.012	30,000	0.019	172.9×17.2

4.1 Results for the structure of shocks through dissipative hard disks using EDMD

4.1.1 Formation of instability

An example of the simulated shock formation with the presence of inelastic collisions is shown in Figure 4.1 for $\varepsilon = 0.95$, $u_p = 20$ and $u_{max} = 10$. The development of a shock wave similar to the elastic case is seen during the first few frames. The density continues to increase behind the shock wave, bringing rise to high density clusters that begin to extend ahead.

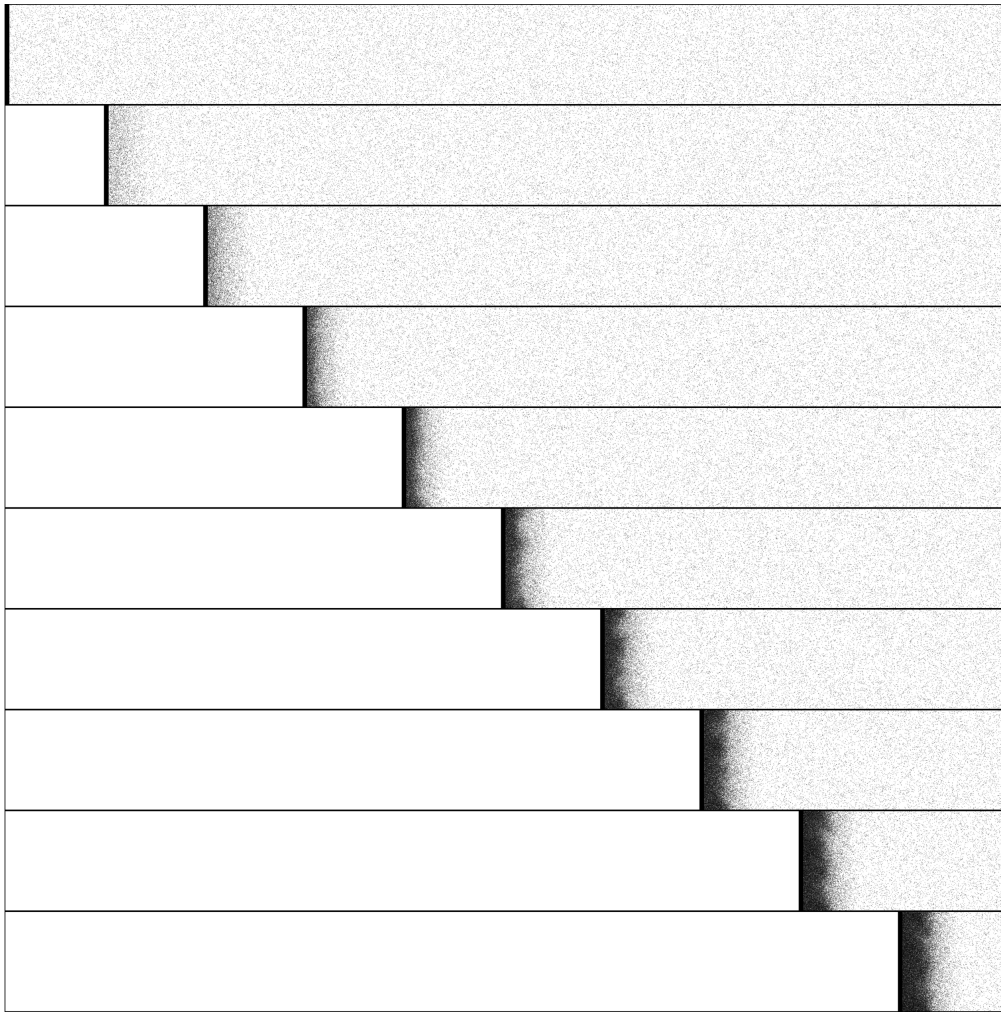


Figure 4.1: Seven consecutive snapshots illustrating the dissipative, unstable shock wave driven by the moving piston for $\varepsilon = 0.95$, $u_p = 20$, $u_{max} = 10$, and $\eta_1 = 0.012$

Figure 4.2 shows the one-dimensional distribution of temperature and density, as well as the streamlines of velocity for a developed, unstable shock wave from the simulation. The distribution of temperature and density in Figure 4.2 are similar to those expected for a relaxing shock wave. The temperature increases following the initial shock and then starts decreasing due to the activated, inelastic collisions. Once the state begins transitioning to the equilibrium region, the flow becomes unstable as is shown through the formation of high density non-uniformities. Retaining kinetic energy in the equilibrium regions also brings rise to convective rolls within these clusters, as seen in the velocity streamlines shown in Figure 4.2.

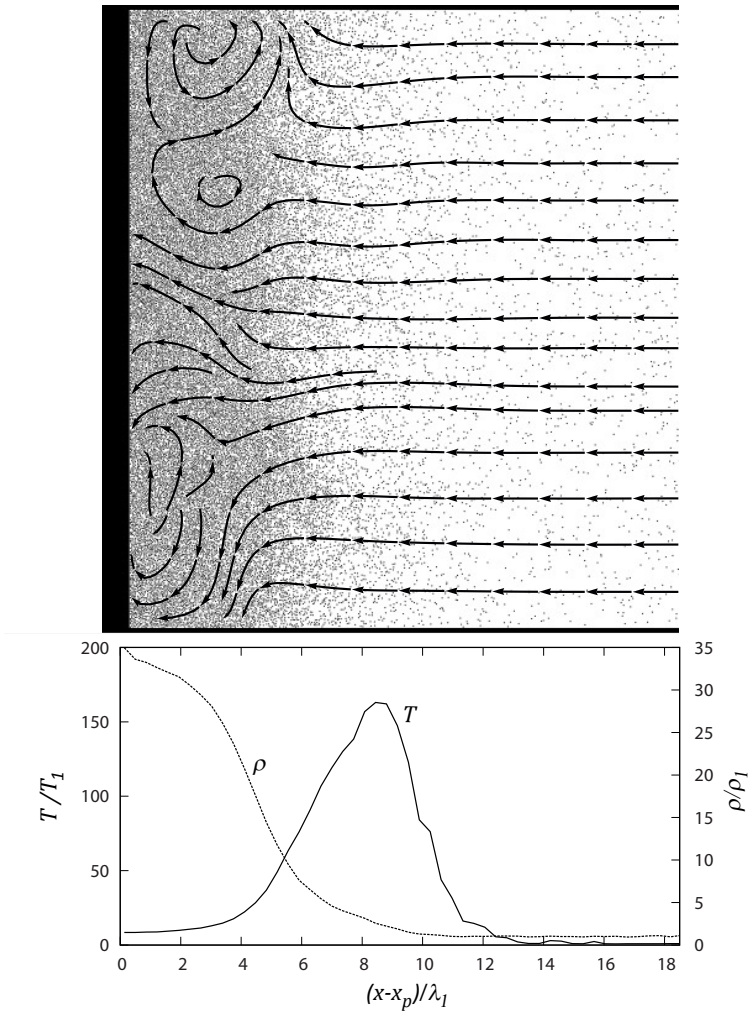


Figure 4.2: Particle distribution and coarse-grained stream-lines (upper), with coarse grained one-dimensional distribution of pressure, temperature and density for $u_p = 20$, $u_{max} = 10$, $\varepsilon = 0.95$ and $\eta_1 = 0.012$.

4.1.2 Shock structure dependence on u_p/u_{max}

The piston velocity u_p and activation velocity threshold u_{max} are varied to see the effect they have on the shock structure. In order to investigate the relationship between these parameters, we observe the results for similar ratios of the two velocities. For this, we vary u_p and u_{max}, maintaining a constant ratio u_p/u_{max}. An example of these results are shown in Figures 4.3-4.5, where u_p and u_{max} are varied, while keeping u_p/u_{max}=2.00 and $\varepsilon = 0.95$. Figure 4.3 shows the unstable shock structure after equal piston displacement, which are similar, with the manifestation of similar sizes of bumpy features. From observation the shock front appears to extend the same distance ahead of the piston.

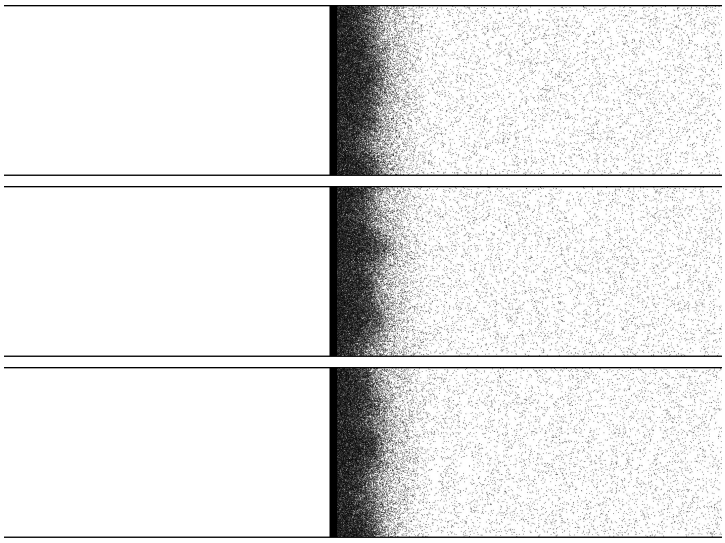


Figure 4.3: Shock structure for similar values of u_p/u_{max}=2, for a mixture of hard disks with $\varepsilon = 0.95$ and $\eta = 0.012$. From top to bottom $u_{max} = 8, 12$ and 15.

This is confirmed in Figure 4.4 and Figure 4.5, which show the coarse grained averaged distribution of temperature and density. Figure 4.5 shows that the distribution of density collapses to the same distribution after equal piston displacement, which agrees with what is seen in Figure 4.3. The temperature distributions for these simulations are shown in Figure 4.4. These results show that the temperature distributions begin rising and approach their peak energies at the same position, demonstrating similar scales for translational excitation. Similarly, the decay of temperature occurs over the same length scales, relaxing to their respective thermalized equilibrium zone after similar lengths. The state at equilibrium is at a higher temperature for higher u_p and u_{max}, yet the length scales are the same for excitation and relaxation of the translational energy of the particles.

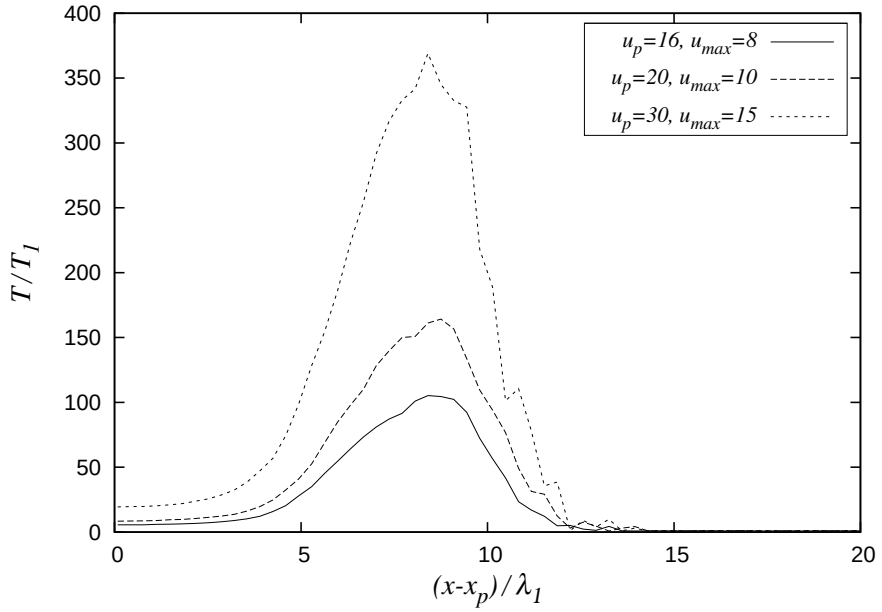


Figure 4.4: Coarse grained one-dimensional distribution of temperature for three different conditions of u_p/u_{max}=2, and $\varepsilon = 0.95$ and $\eta = 0.012$

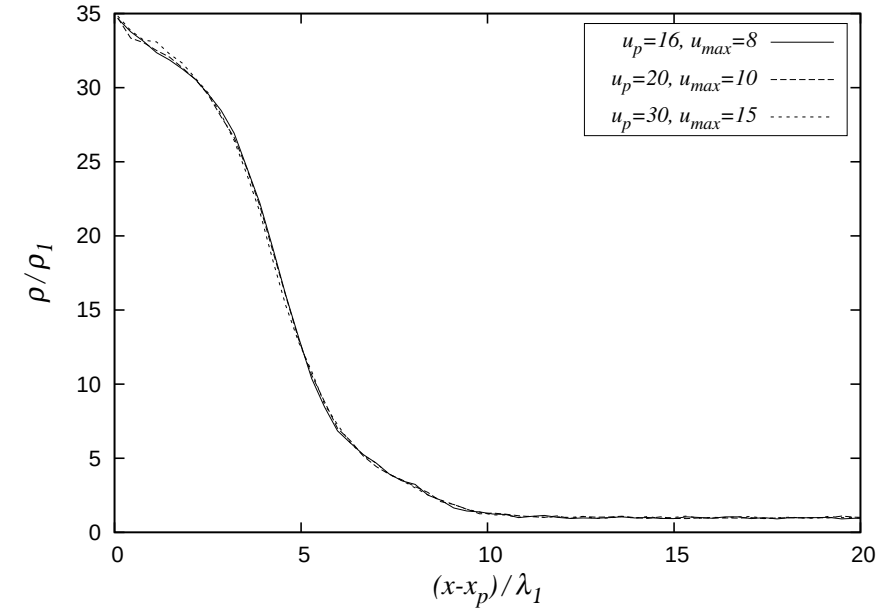


Figure 4.5: Coarse grained one-dimensional distribution of density for three different conditions of u_p/u_{max}=2, and $\varepsilon = 0.95$ and $\eta = 0.012$

The temperature distribution from Figure 4.4 is rescaled in Figure 4.6, where the kinetic energy of the particles is rescaled by the activation energy of the particles (u^2_{rms}/u^2_{max}). This is similar to what is done in combustion whereby the temperature is scaled by the activation energy for exothermic reactions (Williams, 1985).

Rescaling by the activation energy gives a common equilibrium state, as shown in Figure 4.6, which shows that the profiles collapse onto each other after equal piston displacement. This confirms that the excitation and relaxation occur over equal length scales. Figure 4.6 also demonstrates how the kinetic energy in the equilibrium state stabilizes when the kinetic energy is approximately 8% of the activation energy. This is a characteristic that is seen for all simulated shock waves that underwent relaxation. Therefore, from Figures 4.3-4.6, we can conclude that the shock structures and the length scales for the distributions are identical for simulations with equal values of ε and u_p/u_{max}.

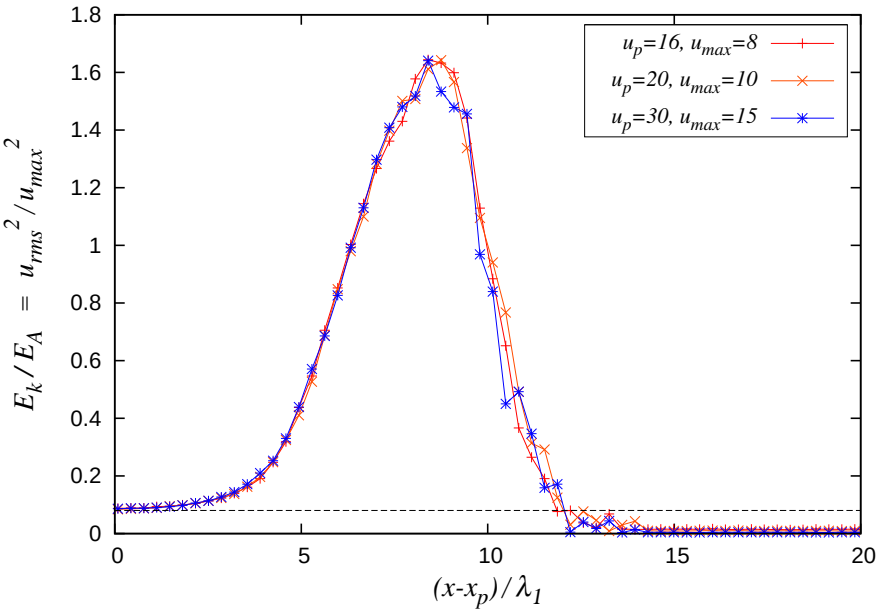


Figure 4.6: Coarse grained kinetic energy distribution for three different conditions of u_p/u_{max}=2, where the kinetic energy ($\frac{1}{2}u^2_{rms}$) is scaled by the activation energy ($\frac{1}{2}u^2_{max}$). For these simulations, $\varepsilon = 0.95$ and $\eta = 0.012$. Dotted line represents the state at equilibrium in the relaxed state, occurring at $E_k/E_A = 0.08$

In order to investigate the structure's dependence on u_p/u_{max}, we keep u_{max} constant and vary u_p. Figure 4.7 shows the structure for increasing values of u_p/u_{max} for $\varepsilon = 0.95$, 0.90 and 0.80, and $u_{max} = 10$. Results demonstrate that instability is noticeable when $u_p/u_{max} \geq 1.25$. It is difficult to observe whether $u_p/u_{max} = 1.00$ develops instabilities

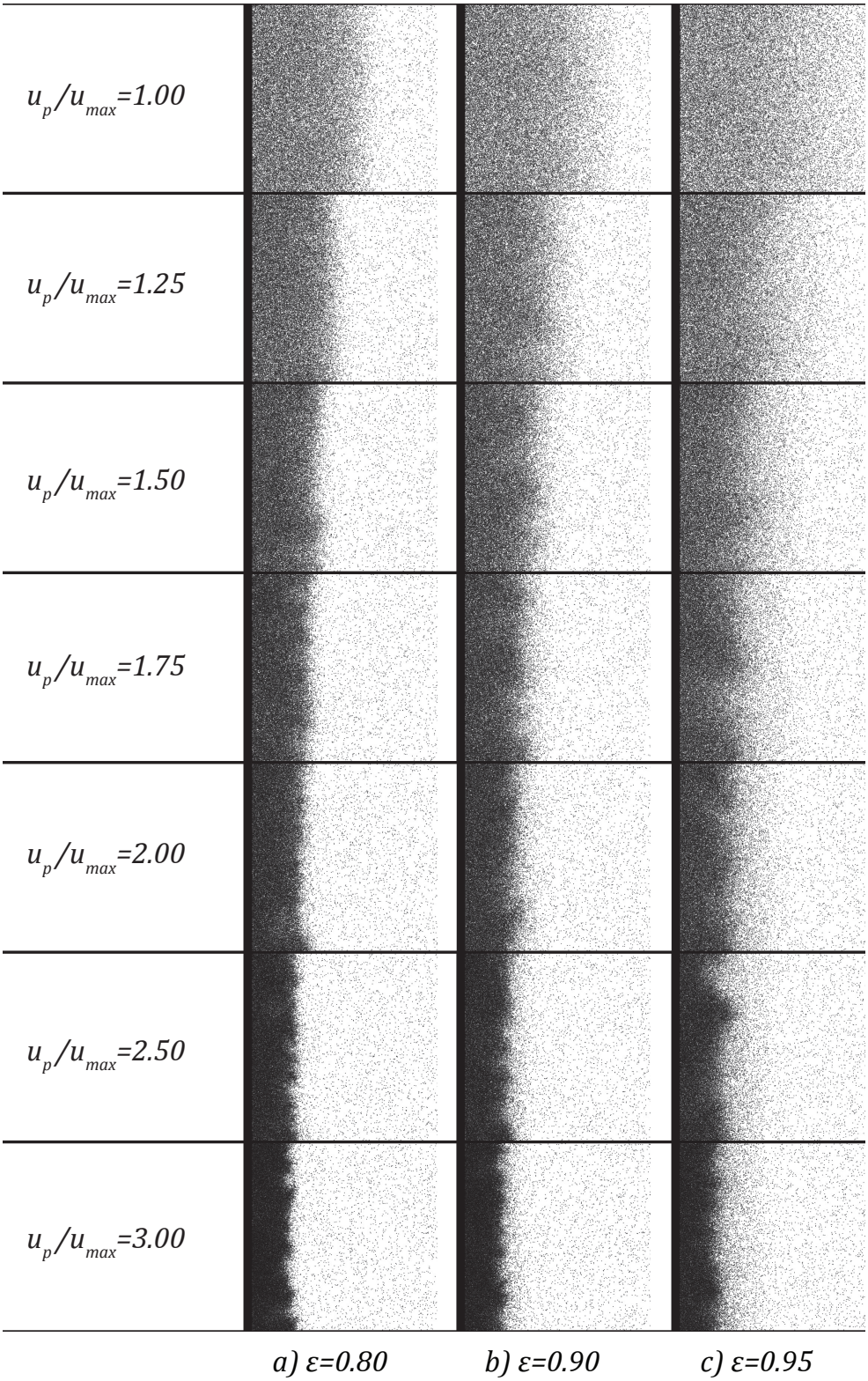


Figure 4.7: Comparison of shock morphology for different values of u_p/u_{max} and ε where $u_{max} = 10$ $\eta_1 = 0.012$

within the domain, although the case where $\varepsilon = 0.8$ appears to be developing some non-uniformities at the shock front. The size of the domain may be limiting instability from being seen at this shock strength. However, for the case of $u_p/u_{max} = 0.75$, the simulated shock waves clearly did not develop instabilities. Therefore, results show a transition from stable to unstable shock waves in the simulations when $u_p/u_{max} \geq 1.00 - 1.25$.

4.1.3 Shock structure dependence on ε

In order to investigate the effect that ε has on the shock structure, we vary ε, while keeping u_p/u_{max} constant. Figure 4.8 shows the resulting shock structure for $\varepsilon = 0.95$, 0.90 and 0.80, with $u_p/u_{max} = 2.00$. Results demonstrate how the structure is more tightly packed for lower values of ε. This is seen through the formation of smaller bumps for lower ε, which do not extend as far ahead as those for higher ε. Results for the distribution of

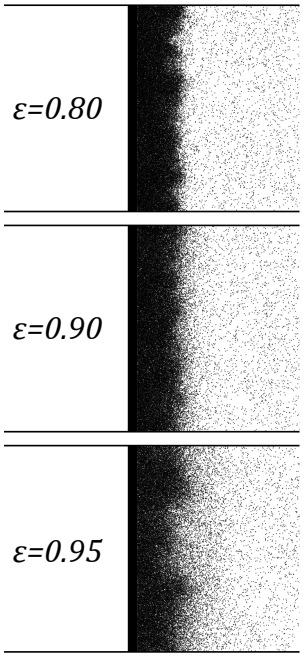


Figure 4.8: Comparison of shock structure after similar piston displacement for different values of $\varepsilon = 0.80$, 0.90 and 0.95 with $u_p/u_{max} = 2.00$ and $\eta_1 = 0.012$

temperature are shown in Figure 4.9 with the same parameters used in Figure 4.8. The results show that the kinetic energy is excited and also relaxes over a shorter length as ε decreases. The peak temperature also decreases as ε decreases due to the more dissipative collisions occurring during the initial excitation. The distribution of density is shown in Figure 4.10 for this case and confirms what we see in Figure 4.8 for the non-uniformities

extending further ahead with increasing ε, causing the density to increase further ahead of the piston. All cases of ε collapse to similar densities at the piston, but the density gradient is much higher for decreasing ε, shown by the steep change in density for $\varepsilon = 0.80$, compared to a more gradual change in density for $\varepsilon = 0.95$ in Figure 4.10.

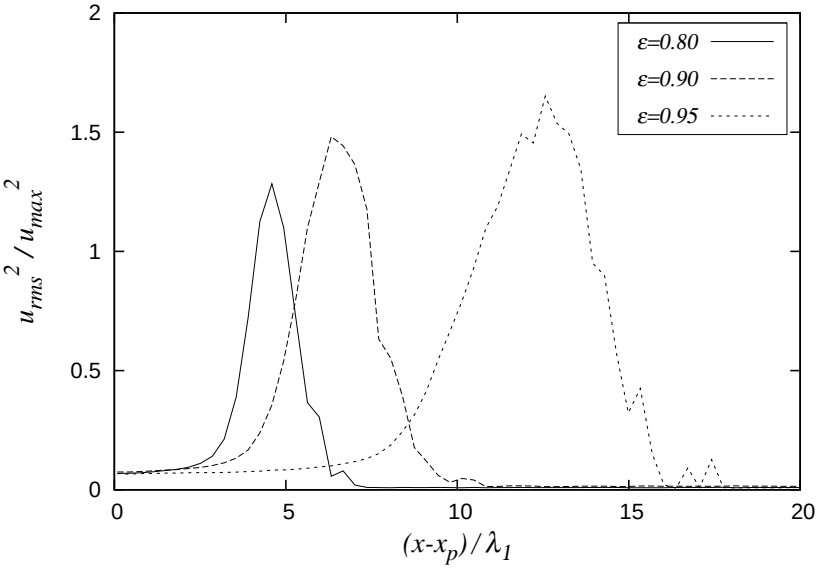


Figure 4.9: Coarse grained one-dimensional distribution of temperature for different values of ε, with u_p/u_{max}=2 and $\eta = 0.012$

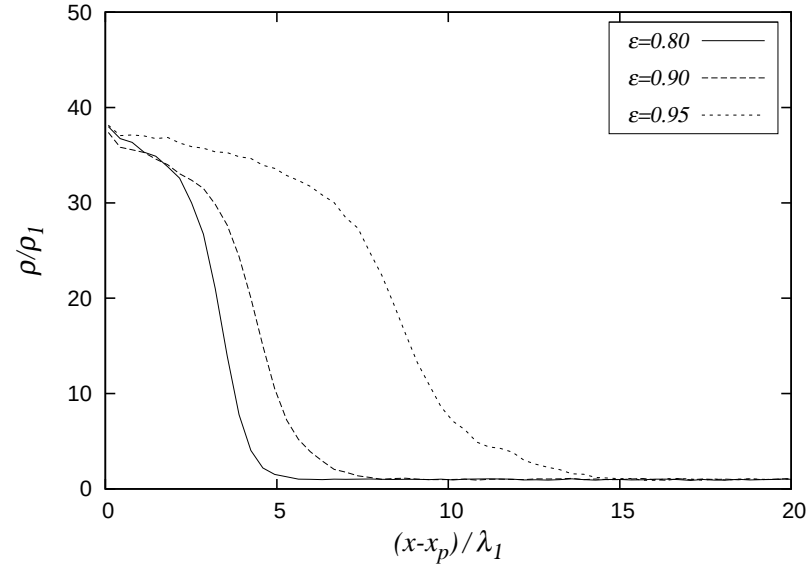


Figure 4.10: Coarse grained one-dimensional distribution of density for different values of ε, with u_p/u_{max}=2 and $\eta = 0.012$

4.2 Parameters controlling instability

Up to this point, the results show that the shock structure does indeed become unstable with the presence of dissipative collisions. The instability manifests itself in the form of high density non-uniformities behind the shock wave, showing a "finger-like" instability. The distribution of temperature in Figure 4.2 shows that the instability begins as the temperature decays to an equilibrium state. In order to better understand the instability, we vary the shock strength u_p, activation velocity u_{max}, and coefficient of restitution ε, to see the effect they have on both the rate of relaxation and the degree of instability. In other words, we wish to investigate the effect varying these parameters have on the relaxation length l_R, and the size of these bumps δ (as shown in Figure 4.11). This allows for the two length scales characterizing the unstable shock waves to be quantified.

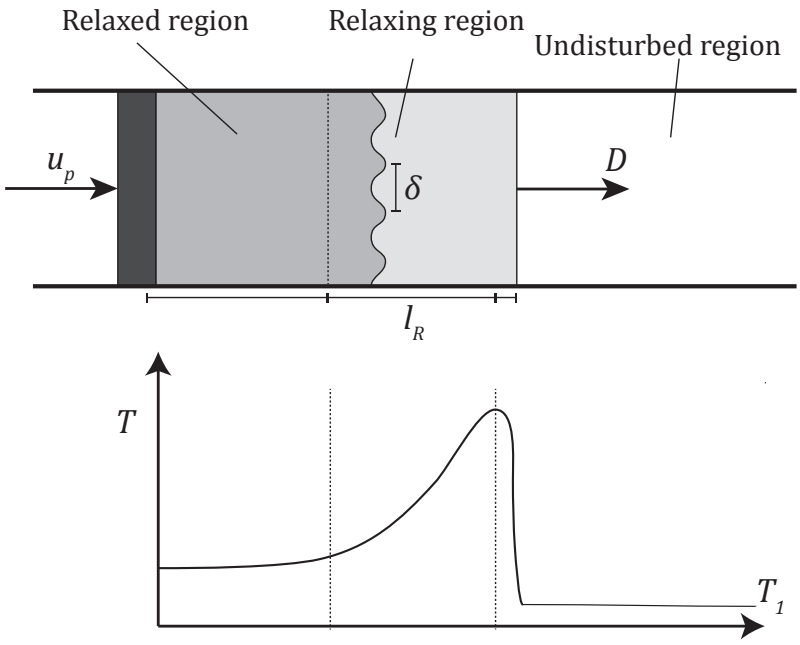


Figure 4.11: Schematic of the properties of the simulations for the relaxation length l_R and spacing of clusters δ

4.2.1 Relaxation length

The relaxation length l_R is found by fitting the decay of kinetic energy across the shock wave to an exponential decay rate with respect to the distance from the peak temperature.

The exponential decay equation takes the form:

$$\frac{u_{rms}^2(l)}{u_{max}^2} = A \exp\left(-\frac{l}{l_R}\right) + b \tag{4.1}$$

A summary of calculated results for l_R are shown in Figure 4.12 for varying u_p/u_{max} and $\varepsilon = 0.8, 0.9$ and 0.95, where the relaxation length is normalized by the initial mean free path. Results show that the l_R is a property of two parameters: ε and u_p/u_{max}. Results demonstrate that as ε decreases, the relaxation length decreases. Similarly, when the shock strength is increased (increasing the ratio between u_p and u_{max}) the relaxation length decreases, until it approaches a limit.

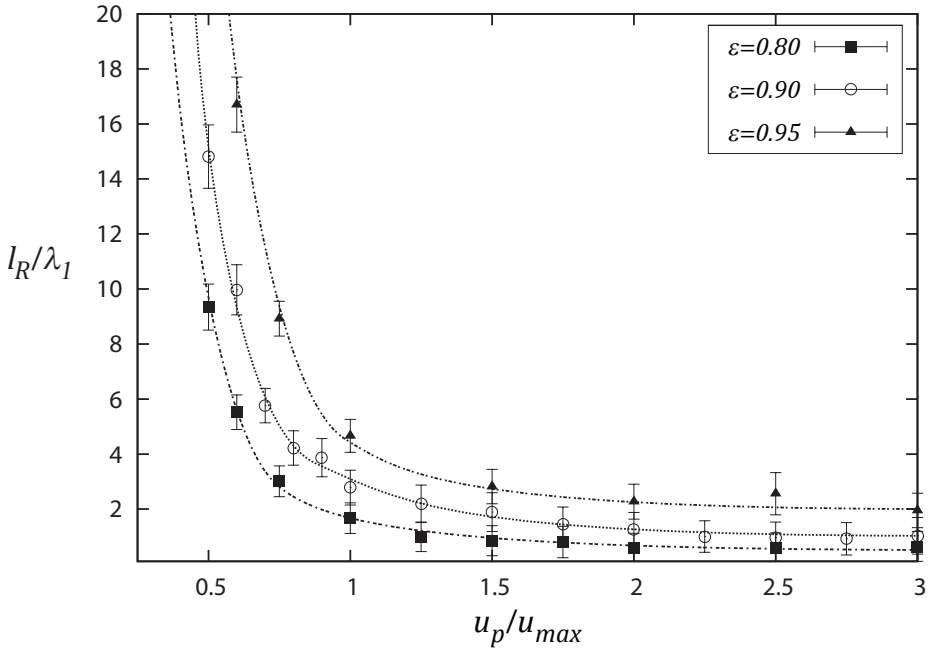


Figure 4.12: Results for the relationship between the relaxation length l_R and the ratio of u_p/u_{max} for $\varepsilon = 0.8$, 0.9 and 0.95

4.2.2 Characteristic spacing of instability

The effect that the controlling parameters have on the size and spacing between the bumps is investigated to see how it compares with the relaxation length. The spacing between the clusters is approximated by counting the number of bumps generated by each unstable shock and dividing this value from the height of the domain. Figure 4.13 shows the results for the relationship between the bump spacing and the relaxation length, for various values

of u_p/u_{max} and ε. The error in Figure 4.13 is due to the lack in accuracy estimating the number of bumps within the domain. Results show that the spacing between the bumps decreases as the relaxation length decreases, which is directly related to u_p/u_{max} increasing. The spacing between the bumps is also found to be proportional to the relaxation length for all cases of ε, where the results show that $\delta \approx 2 - 4 \times l_R$.

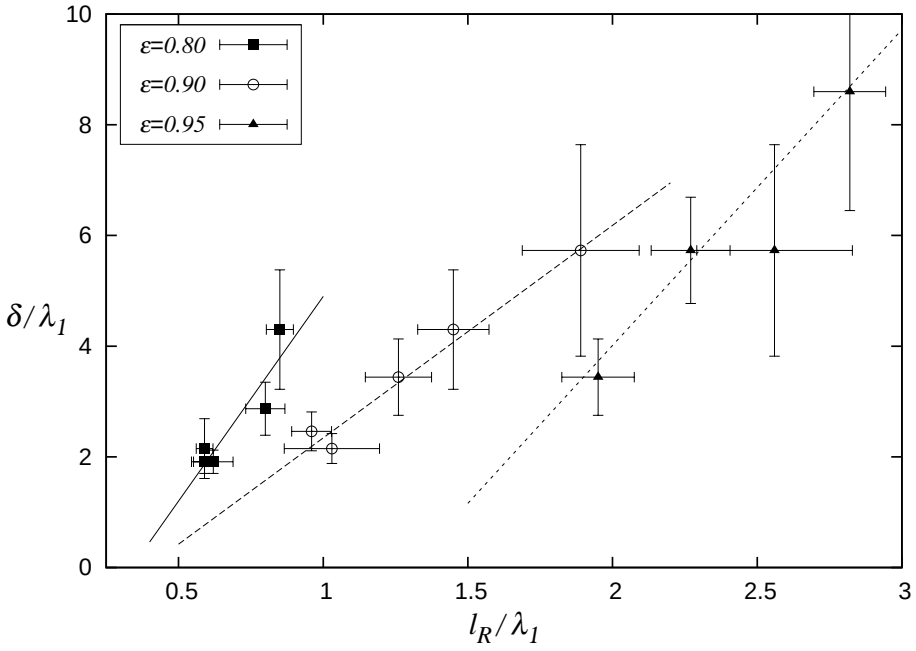


Figure 4.13: Compilation of the numerical results for the relationship between the spacing between clusters δ and the relaxation length l_R

4.3 Properties related to shock theory

The results are investigated in the framework of shock wave theory, in order to see if the simulated instability can be explained by previous mechanisms proposed for explaining the instability in shock waves. This is done by investigating the post-shock jump conditions from this Chapter and compare with the results from Chapter 3 under similar conditions.

4.3.1 Shock Hugoniot

The Hugoniot is constructed in Figure 4.14 for the state at equilibrium for different piston velocities and $u_{max} = 10$. The equilibrium was taken as the point where the kinetic energy was 8% of the activation energy, as revealed from Figure 4.6. The results show that the

post-shock state follows the isotherm corresponding to the equilibrium temperature. By including the Helfand's EOS (3.5) and knowing the temperature of the equilibrium state, the isotherm corresponding to $\frac{u_{rms}^2}{u_{max}^2} = 0.08$ is expressed as:

$$\frac{p_2}{p_1} = 0.08 \frac{v_1}{v_2} u_{max}^2 \left(\frac{1-\eta_1}{1-\frac{v_1}{v_2}\eta_1} \right)^2 \tag{4.2}$$

As discussed in the Introduction, shock instability can be predicted based on the shape of the Hugoniot. One such instability is the D'yakov-Kontorovich instability, which is predicted to occur for states that lie along the Hugoniot where the Hugoniot has a small, positive slope. However, Figure 4.14 demonstrates how the Hugoniot always has a negative slope, therefore instability associated with the D'yakov-Kontorovich instability can be ruled out.

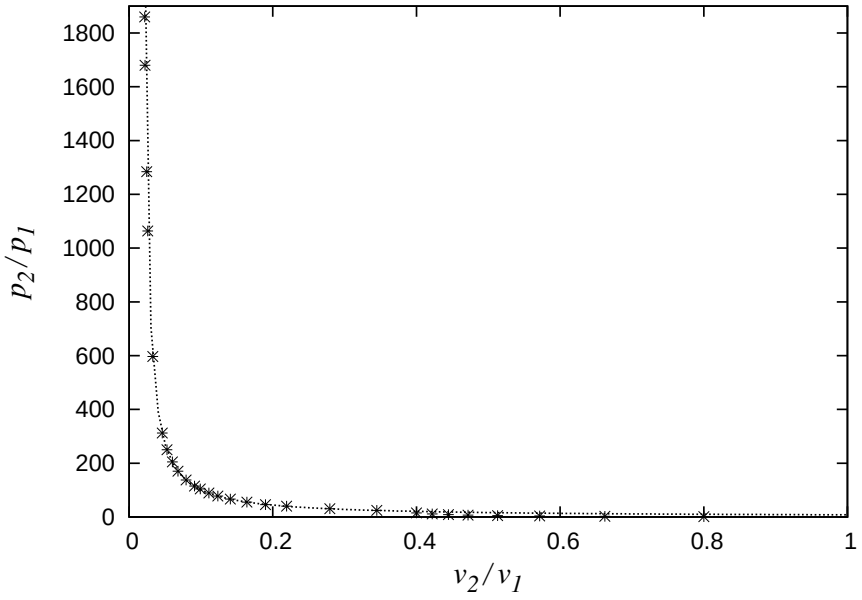


Figure 4.14: Shock Hugoniot of equilibrium states for simulated results for various values of u_p and $u_{max} = 10$

Figure 4.15 shows a zoomed in view corresponding to the shock strengths ranging from $u_p/u_{max} = 0.025$ to $u_p/u_{max} = 0.8$. This range demonstrates that there is a transition in the Hugoniot corresponding to where the piston velocity is high enough to activate the inelastic collisions. Results show that at weaker piston velocities ($u_p/u_{max} \leq 0.2$) the post-shock state follows the theoretical Hugoniot derived in Chapter 3 (3.14), where all

collisions are elastic. A transition occurs from $u_p/u_{max} = 0.2 - 0.3$ where the post-shock state follows the isotherm (4.2) for $u_{max} = 10$.

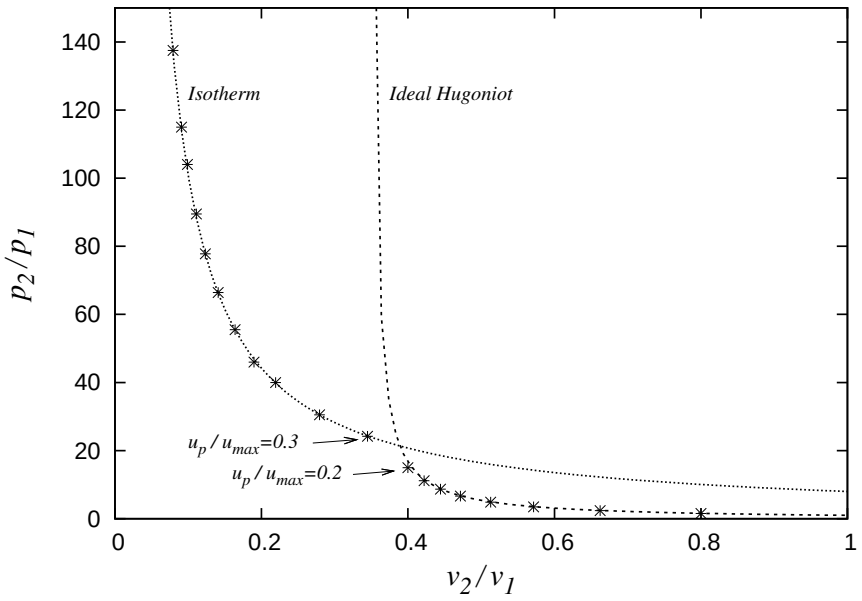


Figure 4.15: Zoomed in view of the results for the shock Hugoniot where the results show the departure from the elastic Hugoniot for $\eta_1 = 0.012$ and begin following the isotherm

4.3.2 Rayleigh line

The shape of the Hugoniot in Figure 4.15 is similar to a media undergoing a phase transition, as demonstrated in Figure 1.4a. As discussed in the Introduction, shock instability is expected in media characteristic of this Hugoniot when the Rayleigh line intersects at multiple points, leading to shock splitting. This is expected to occur when the Rayleigh line intersects the kink in the Hugoniot (in the range of $u_p/u_{max} = 0.2 - 0.3$ for Hugoniot constructed in Figure 4.15).

Figure 4.16 shows the Rayleigh line in the case where $u_p/u_{max} = 2.00$, for $\varepsilon = 0.80$ and 0.95. Although the shock waves for these parameters became unstable, the Rayleigh line does not intersect at multiple points. This can be shown for all cases of u_p/u_{max} since the cases where instability occurs ($u_p/u_{max} \geq 1.00$) correspond to equilibrated states far away from this kink in the Hugoniot. Therefore instability from shock splitting is not expected to occur based on the shape of the Hugoniot.

The Rayleigh curves in Figure 4.16 also demonstrate an increasing concavity as ε

decreases. This is a characteristic expected for more viscous shock waves.

Note that the transient pressure for the Rayleigh line uses the Helfand equation of state for hard disks, which assumes that $\varepsilon = 1$. A more appropriate equation of state would be used to account for ε (Brilliantov & Pöschel, 2004). However, the literature assumes $\varepsilon = \text{const}$, which is different than our simulations, where $\varepsilon = \text{const} < 1$ only when $u_{impact} > u_{max}$. Due to this difference, the Helfand equation of state is used. Since the equilibrium state assumes that the collisions are almost all elastic, the Hugoniot is assumed to be accurate with the Helfand equation of state.

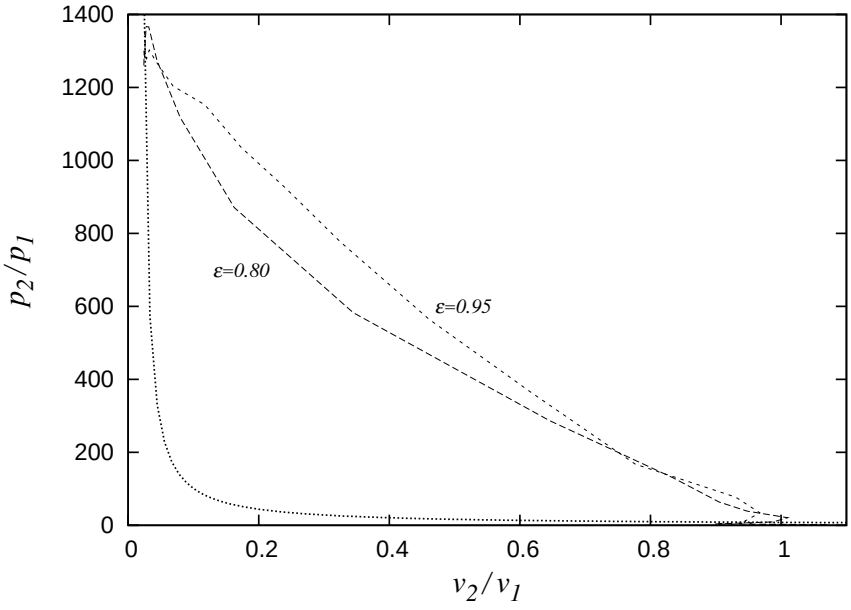


Figure 4.16: Rayleigh curves for $u_p/u_{max} = 2.00$ for $\varepsilon = 0.8$ and 0.95.

4.3.3 Shock front velocity

It is also physically meaningful to find the velocities of the simulated shock waves, in order to compare with the departure from the velocities found in Chapter 3. Figure 4.17 shows the results for the shock velocities for different values of u_p/u_{max} and $\varepsilon = 0.90$. Results show that at the lower velocities, up to $u_p/u_{max} = 0.2$, the velocities of the shock waves follow the velocities for elastic shock waves found in Chapter 3. The shock velocity then deviates from this ideal behaviour between $u_p/u_{max} = 0.3 - 1.0$ until the velocity approaches the Newtonian limit where $D/u_p \approx 1.0$ (Anderson, 2006). This limit of $u_p/u_{max} \approx 1.0$ also corresponds to where instability begins to form in the simulations,

suggesting a link between the Newtonian limit and the onset of instability.

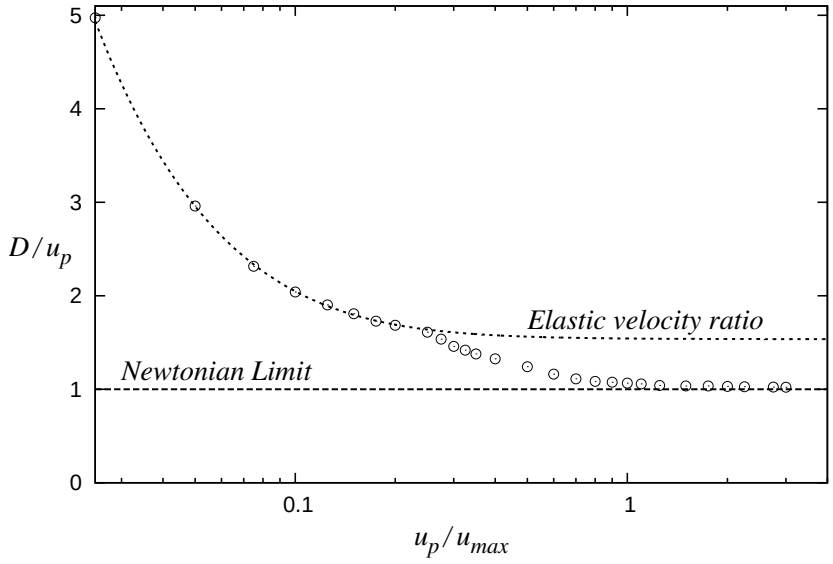


Figure 4.17: Relationship between velocity of shock wave D and piston velocity u_p, for $u_{max} = 10$, $\varepsilon = 0.90$, and $\eta_1 = 0.012$.

Chapter 5

Comparison with clustering instability

In the previous chapter, simulations showed that a shock structure does indeed become unstable with the presence of dissipative collisions. Shock Hugoniot related explanations for the instability were ruled out. We now turn to another mechanism for instability previously documented for homogeneous granular gases. As discussed in the Introduction, a homogeneous system of colliding inelastic particles becomes unstable and forms clusters (Goldhirsch & Zanetti, 1993). The mechanism for this cluster formation is the formation of internal pressure gradients, which drive the material from less dense to denser regions. To appreciate the physical mechanism, consider a density fluctuation in the gas. The denser region exhibits a higher frequency of collisions. Since the collisions are inelastic, the denser region thus loses energy more quickly. As a result, pressure locally drops faster than in less dense regions. The pressure gradient thus drives material towards the denser regions, and high density clusters form.

The clustering instability mechanism has been documented for an idealized setting of a dissipative gas kept at constant volume, such as that shown in Figure 1.6. For this case, the material cools at constant macroscopic specific volume. During this cooling process, non-homogeneities form. In shock waves, however, the specific volume is not constant, as the material flowing across the shock wave compresses as it cools, owing to the hydrodynamic shock compression and relaxation processes. The effect of density changes, in the context of the Goldhirsch and Zanetti instability mechanism, is to locally modify the collision rates, since the collision rates are functions of density. In order to make a meaningful comparison between the instability of the constant specific volume case and the shock case, it is sufficient to adopt a Lagrangian description of the material being

shocked, and compare the time evolution of a material element traversing the shock wave structure (see Figure 5.1) with the time history of cooling in a constant density material element. For a meaningful comparison, and in order to avoid considerations of the density changes, this comparison is to be made on time scales corresponding to the frequency of collisions, i.e., the **local** mean free time. This permits to automatically avoid accounting for density changes.

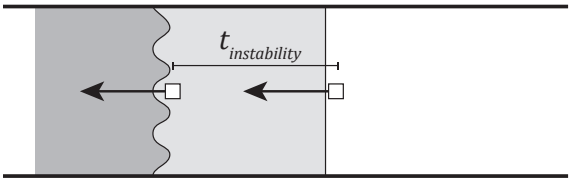


Figure 5.1: Sketch of the to instability from the shock front frame of reference

This chapter thus first revisits the criteria for instability in the constant volume cooling (Goldhirsch & Zanetti, 1993), by formulating a criterion for instability based on **local** mean free time scales. This evolution is then compared with the relaxation phenomena in shock waves, also analyzed on the same time scales.

5.1 Clustering at constant volume

A dissipative gas kept at constant volume cools. In granular flows, this is referred to as the Homogeneous Cooling State, where the temperature evolution is given by Haff's law (see, for example, Brilliantov & Pöschel (2004)). This Homogeneous Cooling State has been shown to be unstable. Previous studies have looked extensively into the parameters controlling this instability (Goldhirsch & Zanetti, 1993; Mitrano *et al.*, 2011; Pöschel *et al.*, 2005). Mitrano *et al.* (2011) concludes that the onset of sensible clustering occurs when the evolution of granular temperature deviates by 5% from Haff's law. In the present work, we adopt the same criterion for onset of instability, and pose the question:
How many local mean free times are required for the gas to develop instability?

5.1.1 Haff's law

We first wish to express Haff's law in a time coordinate normalized by the local mean free time.

The theoretical evolution of temperature for a cooling homogeneous granular gas follows Haff's law (Brilliantov & Pöschel, 2004):

$$\frac{T(t^*)}{T_1} = \frac{1}{\left(1 + t^* \frac{1}{4}\left(1-\varepsilon^2\right)\left(1+\frac{3}{16}a_2\right)\right)^2} \tag{5.1}$$

where

$$a_2 = \frac{16\left(1-\varepsilon\right)\left(1-2\varepsilon^2\right)}{57 - 25\varepsilon + 30\varepsilon^2(1-\varepsilon)} \tag{5.2}$$

Equation (5.1) is expressed with time normalized by the initial mean free time, τ_1. Now we wish to express this relationship with time scaled by the local mean free time, i.e.,

$$t' = \frac{t}{\tau} \tag{5.3}$$

In order to express in term of local mean free time, we look at the ratio between the initial mean free time and the local mean free time, which can be expressed as:

$$\frac{\tau_1}{\tau} = \frac{\rho(\eta) u_{rms}}{\rho_1(\eta_1) u_{rms(1)}} \tag{5.4}$$

The density ρ and packing factor η remain constant, from which we can simplify (5.4) to get

$$\frac{\tau_1}{\tau} = \frac{u_{rms}}{u_{rms(1)}} = \sqrt{\frac{T}{T_1}} \tag{5.5}$$

Including this relationship, Haff's law (5.1) can be expressed in terms of t' as:

$$\frac{T(t')}{T_1} = \frac{1}{\left(1 + t' \sqrt{\frac{T(t')}{T_1}} \left(\frac{1}{4}\left(1-\varepsilon^2\right)\left(1+\frac{3}{16}a_2\right)\right)\right)^2} \tag{5.6}$$

This equation permits us to calculate the theoretical evolution of temperature for a cooling homogeneous granular gas in terms of the number of local mean free times. Figure 5.2 shows an example of this decay for $\eta = 0.1$ and $\varepsilon = 0.9$.

5.1.2 Determination of clustering time using EDMD

Having determined Haff's law in terms of the desired time variable, we can now determine the approximate time for onset of clusters by adopting the 5% criterion of Mitrano *et al.* (2011). We thus conducted constant volume clustering simulations in order to determine the time when the energy of the system departs by more than 5% from Haff's law.

Seven initial conditions were investigated using EDMD for each value of $\varepsilon = 0.8$, 0.9 and 0.95, and N =10,000. These conditions are summarized in Table 5.1, where the radius and domain size are normalized by the initial mean free path λ_1.

Table 5.1: Parameters used to investigate the clustering time via constant volume simulations using EDMD, where the length scales are normalized by the initial mean free path λ_1

Packing factor η	Particle radius R	Domain size $L_x \times L_y$
0.05	0.086	68.5×68.5
0.10	0.189	105.6×105.6
0.125	0.246	123.5×123.5
0.15	0.310	141.7×141.7
0.175	0.379	160.5×160.5
0.20	0.455	180.3×180.3
0.25	0.631	223.9×223.9

Ensemble averages over 10 simulations were performed with equal time increments of $0.1\tau_1$ for the outputs. The local mean free time was estimated from the temperature obtained using (5.4). An example of the temperature decay is shown in Figure 5.2 for $\varepsilon = 0.9$ and $\eta = 0.1$, where simulated results (solid line) are compared with Haff's law (dashed line).

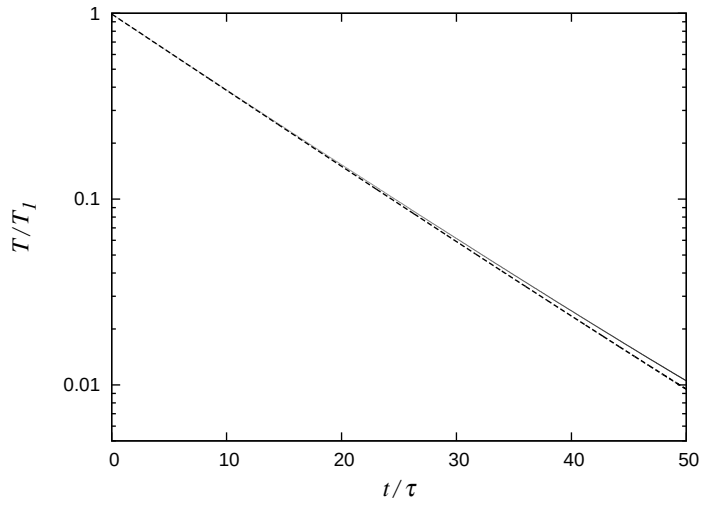
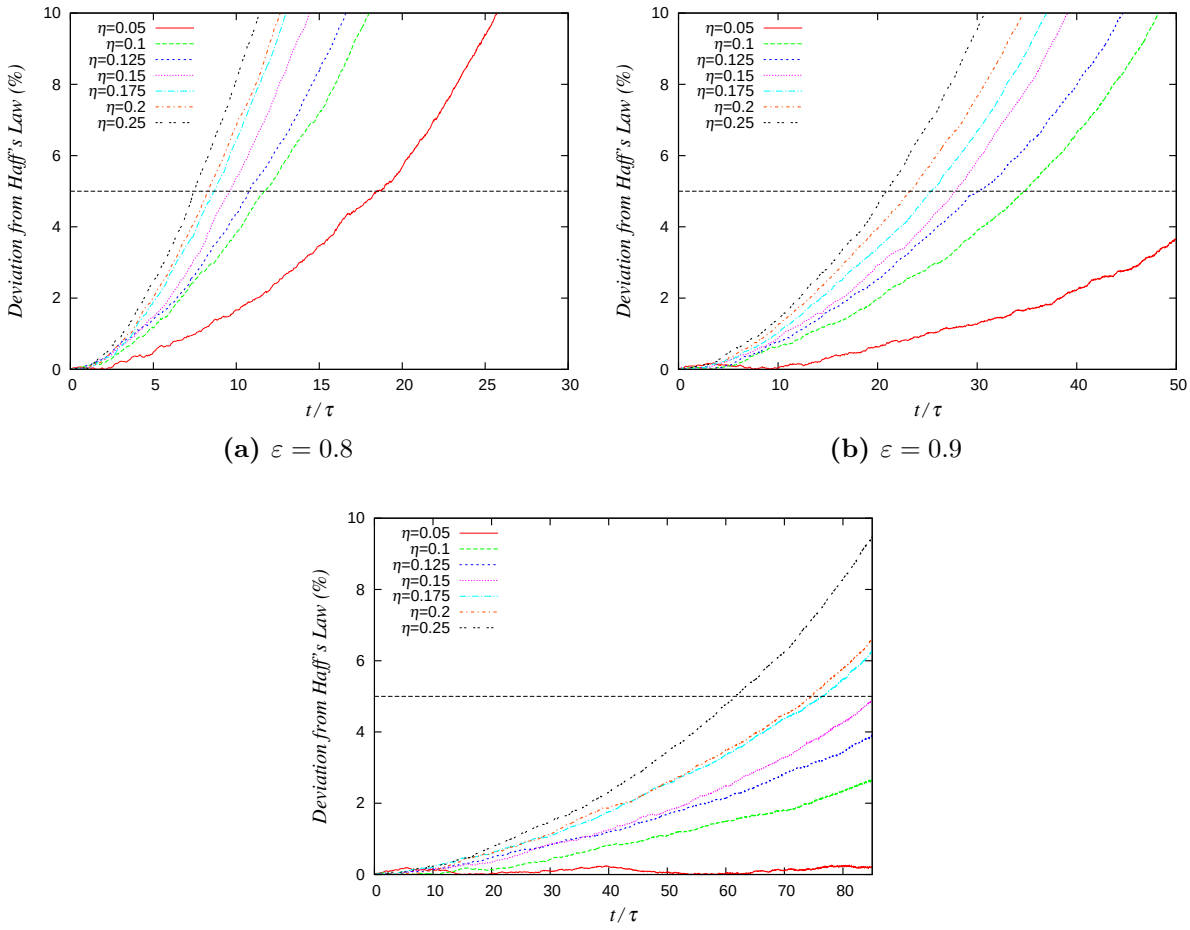


Figure 5.2: Comparison between Haff's law (dashed) with simulated results for the temperature decay in homogeneous granular gas with $\eta = 0.1$ and $\varepsilon = 0.9$

Figure 5.2 demonstrates how the granular temperature follows the theoretical temperature decay at the earlier times and then begins departing from the theoretical decay. This deviation is summarized for different values of ε and η_1 in Figures 5.3(a)-(c), where the results have been ensemble averaged over 10 simulations. The duration of the simulations was $300\tau_1$, which was not a sufficient time for instability to occur for $\eta = 0.05, 0.1, 0.125$ and $\varepsilon = 0.95$.



(a) $\varepsilon = 0.8$ **(b)** $\varepsilon = 0.9$

(c) $\varepsilon = 0.95$

Figure 5.3: Comparison between the deviation from Haff's law over time for $\varepsilon = 0.8, 0.9$ and 0.95 with different initial values of η. A deviation of 5% corresponds to the onset of clustering

Results give the time to clustering, which is expected to occur when the deviation between Haff's law and the numerical results is 5%. A summary of these results are shown in Figure 5.4 which gives the simulated time normalized by the local mean free time for different values of ε and η. Since η does not remain constant across the shock waves, the full range in times for each ε is compared to the time scale we get from the simulated shock waves in the next section.

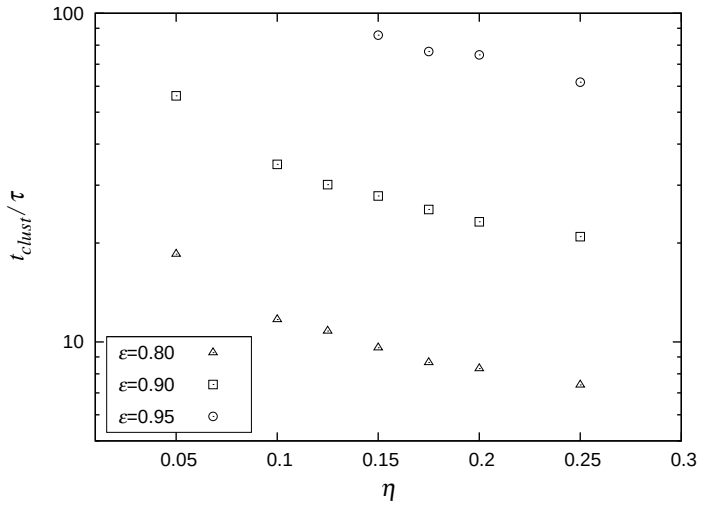


Figure 5.4: Results for the time to clustering t_{clust} for different values of ε and η

As ε decreases, the time to instability also decreases. For example, at $\eta = 0.2$ clustering instability is expected after approximately 8 mean free times for $\varepsilon = 0.80$ compared to 75 mean free times for $\varepsilon = 0.95$. The results also demonstrate that clustering instability is expected to occur after fewer mean free times for increasing η. For $\varepsilon = 0.8$, in the dilute regime of $\eta = 0.05$, instability is expected to occur after approximately 19 mean free times, whereas for the denser regime of $\eta = 0.25$ instability is expected after 7 mean free times.

5.2 Comparison with relaxation mean free times in shock waves

We now turn to the relaxation phenomenon in the shock waves analyzed in Chapter 4. In order to compare with the criteria developed above for the onset of clustering, we need to determine the relaxation time history of a Lagrangian material particle across the shock wave. Let t denote the time coordinate of a material element traversing the shock structure and x' the spatial coordinate of that element in a frame of reference moving with the shock (see Figure 5.5)

Using the invariance of the mass flux across the shock wave, we have

$$\rho_{ref} u_{ref} = \rho(x')u(x') \tag{5.7}$$

Since the velocity of the fluid element can be written as,

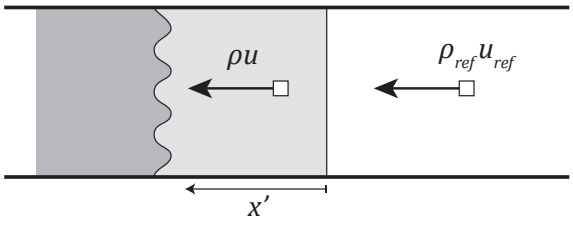


Figure 5.5: Shock frame of reference for unstable shock waves

$$\frac{dx'}{dt} = u(x') \tag{5.8}$$

Making use of (5.7), rearranging and integrating over time we obtain the relation between the passage time t and the spatial coordinate x':

$$t = \int_0^{x'} \frac{\rho dx'}{\rho_{ref} u_{ref}} \tag{5.9}$$

The reference state of the mixture is that of the upstream state, such that ρ_{ref} is the initial density and u_{ref} is the velocity of the shock wave. Parameters on the right-hand side of (5.9) are scaled in accordance with the scaling presented in Chapter 2. Implementing the scaling and adding the ratio between initial mean free time and local mean free time (5.4), the integral takes the following form:

$$t' = \frac{t}{\tau} = \int_0^{x'^*} \frac{\rho^{*2}}{u^*_{ref}} \frac{g(\eta_1 \rho^*)}{g(\eta_1)} \sqrt{T^*} dx'^* \tag{5.10}$$

Equation (5.10) can thus be used to determine the time across a shock wave with respect to the shock front. This is done by taking the results for temperature and density jump across the shock waves in Chapter 4 (such as those seen in Figure 4.4 and Figure 4.5) and integrating over the discrete steps dx which were used for coarse grain averaging.

Figures 5.6-5.8 show the temporal evolution of temperature across the simulated shock waves for different values of u_p/u_{max} and ε. The start time $t' = 0$ is taken at the point of peak temperature.

From these results we wish to get the exponential time constant τ_R found in the exponential decay equation:

$$\frac{u^2_{rms}(t)}{u^2_{max}} = A \exp\left(-\frac{t}{\tau_R}\right) + b \tag{5.11}$$

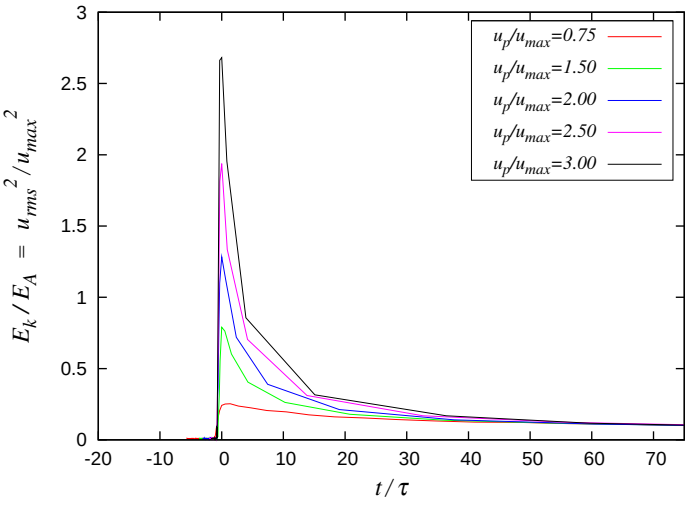


Figure 5.6: Distribution of kinetic energy behind the shock wave in terms of time travelled from shock front for various values of u_p/u_{max} and $\varepsilon = 0.80$

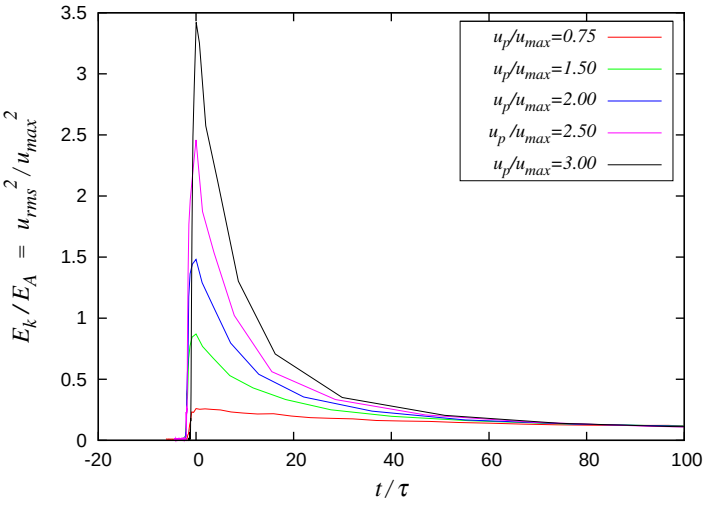


Figure 5.7: Distribution of kinetic energy behind the shock wave in terms of time travelled from shock front for various values of u_p/u_{max} and $\varepsilon = 0.90$

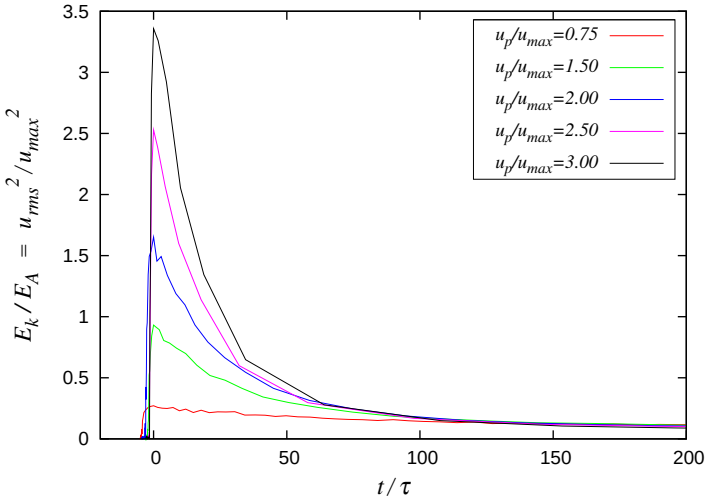


Figure 5.8: Distribution of kinetic energy behind the shock wave in terms of time travelled from shock front for various values of u_p/u_{max} and $\varepsilon = 0.95$

The time τ_R is calculated for different values of ε and u_p/u_{max} by fitting results from Figures 5.6-5.8 to (5.11); these results are shown in Figure 5.9. Results demonstrate that τ_R follows the same trends seen for the relaxation length in Figure 4.12, where the time constant decreases with decreasing ε and increasing u_p/u_{max}. The results show that at higher shock strengths (higher u_p/u_{max}) the time constant approaches some limit, similar to the limit in the relaxation length.

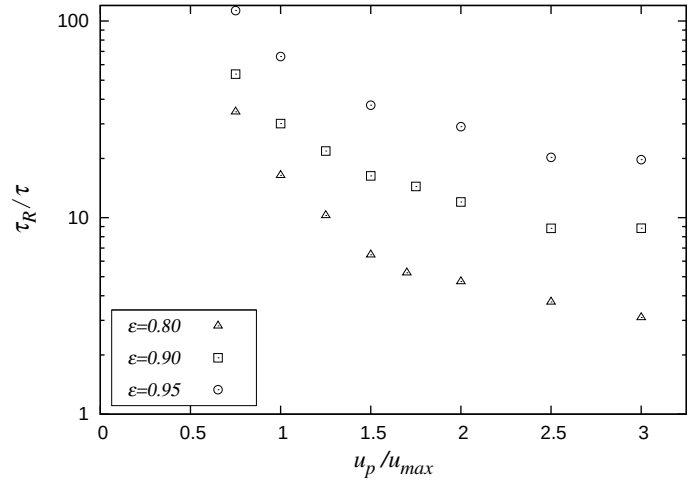


Figure 5.9: Exponential time constant τ_R for the decay across the simulated relaxing shock waves, for various values of u_p/u_{max} and ε

5.3 Comparison between clustering instability and decay rate across simulated shock waves

Since the density increases across the shock wave, the value of η which contributes to the onset of instability can not be determined accurately. For this reason, the full range of clustering time for the range of $\eta = 0.05 - 0.25$ is compared for the calculated results of τ_R for the respective values of ε. Figures 5.10-5.12 compare the values of τ_R with the ranges of clustering time for each value of ε.

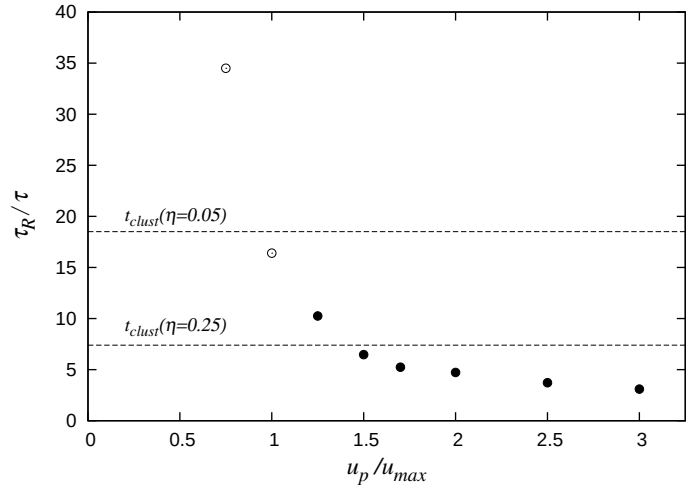


Figure 5.10: The exponential time constant τ_R for different values of u_p/u_{max} and $\varepsilon = 0.80$ with solid points for observed instability, plotted with the range in times for the time to clustering instability for $\varepsilon = 0.80$

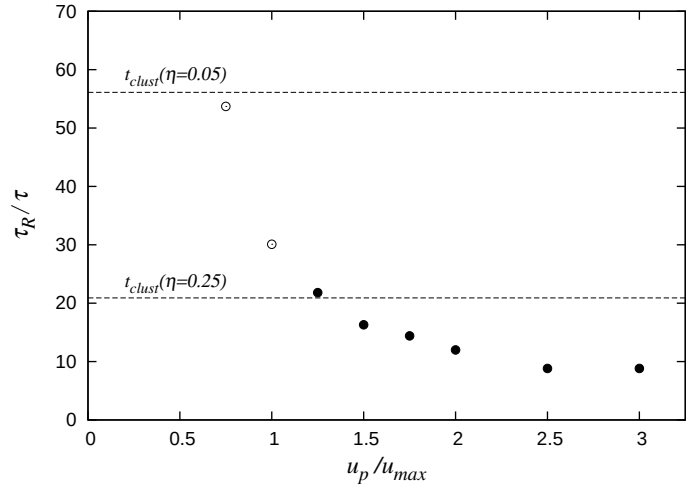


Figure 5.11: The exponential time constant τ_R for different values of u_p/u_{max} and $\varepsilon = 0.95$ with solid points for observed instability, plotted with the range in times for the time to clustering instability for $\varepsilon = 0.90$

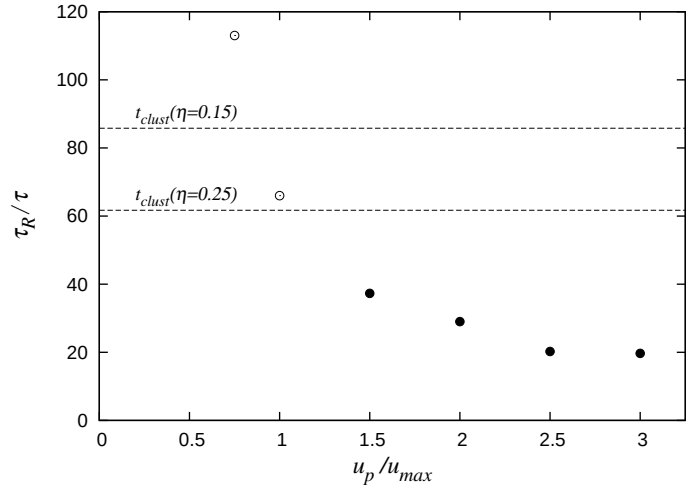


Figure 5.12: The exponential time constant τ_R for different values of u_p/u_{max} and $\varepsilon = 0.95$ with solid points for observed instability, plotted with the range in times for the time to clustering instability for $\varepsilon = 0.95$

Results show similar time scales for the time to the onset of clustering instability and τ_R. For $u_p/u_{max} \geq 1.5$, the values of τ_R lie below these range of clustering times for all ε. For these values the structures are unstable, as shown in Figure 4.7. Similarly, times within these range of times for $u_p/u_{max} \leq 1.25$ show instability, although it is difficult

to observe these instabilities within the domain for $u_p/u_{max} = 1.00$. The value of τ_R for $u_p/u_{max} = 0.75$ lied just on the edge of the range, and in this case no instability was observed.

Instability is seen when τ_R is below, or within the range of times to clustering for similar values of ε. The cases where instability is not observed in the simulations have values of τ_R greater or near the edge to these ranges in time. These results demonstrate a link between the rate of decay across these dissipative shock waves and whether the shock wave will indeed become unstable. Although specific time constants leading to instability are not calculated, the results suggest that instability occurs if the time constant τ_R lies below or within a range in time associated to the clustering instability, then it will become unstable, and if it is higher than this time then the structure remains stable.

Chapter 6

Conclusions

Current shock theory is unable to explain instability that has been observed experimentally for shock waves undergoing strong relaxation effects. The aim of the present study was to shed light on this shock instability by investigating the effect that dissipative processes have on the stability of shock waves.

To the best of our knowledge, the current study is the first to observe shock wave instability in dissipative gases. This was found through a simple molecular dynamics model we developed to look at the classical problem of shock wave driven by a moving piston into a system of hard particles. In order to better emulate activated endothermic processes, collisions between hard particles were elastic, unless an activation threshold was surpassed, representing the activation energy of the reaction. If the threshold velocity was exceeded, the collisions between particles were inelastic, with a constant coefficient of restitution ε.

The numerical experiments conducted have shown that shock waves do indeed become unstable with the presence of activated inelastic collisions, in the form of distinctive high density non-uniformities behind the shock wave. The activation threshold appears to play a fundamental role in permitting the shock state to remain fluidized and continuously energize the shock transition layer. Additionally, the retention of kinetic energy in the equilibrium region leads to convective rolls within the non-uniformities.

The simulated instability was investigated in the framework of shock theory, whereby the constructed Hugoniot ruled out instability associated with shock splitting and the D'yakov-Kontorovich instability. The mechanism responsible for the instability was addressed by studying the time evolution of the material undergoing the shock wave compression and further relaxation. Results show that the shock waves develop the instability on the same times scales as the clustering instability seen in cooling granular gases (Goldhirsch

& Zanetti, 1993). Thus, the results demonstrate that the dominant physical mechanism controlling the instability in the simulations is the same as that involved in the clustering instability in homogeneous granular gases.

In the absence of an activation threshold, the results emulate shock waves through granular media, recovering a similar one-dimensional shock structure to that seen by Goldshtein *et al.* (1996). The present results may also explain the ripple-like phenomena seen in vibrated containers of granular media (Swinney & Rericha, 2004). However, additional effects would need to be accounted for in granular media, such as gravity and non-uniform initial states.

Qualitatively, these results may also explain the mechanism controlling shock instability in relaxing shock waves, where understanding is currently lacking. The results are especially insightful by demonstrating the trends associated with more dissipative shock waves, which can be related to stronger shock waves undergoing increased relaxation effects. Future research will be required to quantify these instabilities for strong shock waves by formulating more realistic models for their prediction. This will require for the modelling of shock wave instability in gases at both the molecular and continuum levels. These future recommendations to advance this study will permit to clarify the origin of anomalous hot spot formation in reactive media and provide predictive analytical models. This will be fundamental in predicting the influence of such hot spots in the reactive flows of typical heavier fuels of practical interest in current and future combustion applications.

Bibliography

Alder, B. J. & Wainwright, T. E. 1957 Phase transition for a hard sphere system. *Journal of Chemical Physics* **27** (5), 1208–1209.

Alder, B. J. & Wainwright, T. E. 1962 Phase transition in elastic disks. *Physical Review* **127** (2), 359–361.

Anderson, John D. 2006 *Hypersonic and High-Temperature Gas Dynamics*. Reston, Virginia: American Institute of Aeronautics and Astronautics, Inc.

Bates, Jason W. & Montgomery, David C. 2000 The Dyakov-Kontorovich instability of shock waves in real gases. *Physical Review Letters* **84** (6), 1180–1183.

Bird, G.A. 1999 *Molecular Gas Dynamics and the Direct Simulation of Gas Flows*. Oxford: Oxford University Press.

Brilliantov, N. & Pöschel, Thorsten 2004 *Kinetic theory of granular gases*. Oxford: Oxford University Press.

Chapman, Sydney & Cowling, T. G. 1970 *The mathematical theory of non-uniform gases: an account of the kinetic theory of viscosity, thermal conduction and diffusion in gases*, 3rd edn. Cambridge: Cambridge University Press.

Fickett, Wildon & Davis, William Chester 2000 *Detonation: Theory and Experiment*. Mineola, N.Y.: Dover Publications.

Glass, I.I. & Liu, W. S. 1978 Effects of hydrogen impurities on shock structure and stability in ionizing monatomic gases. 1. argon. *Journal of Fluid Mechanics* **84**, 55–77.

Goldhirsch, I. & Zanetti, G. 1993 Clustering instability in dissipative gases. *Physical Review Letters* **70** (11), 1619–1622.

Goldshtein, A., Shapiro, M. & Gutfinger, C. 1996 Mechanics of collisional motion of granular materials 3. Self-similar shock wave propagation. *Journal of Fluid Mechanics* **316**.

Griffiths, R. W., Sandeman, R. J. & Hornung, H. G. 1976 Stability of shock waves in ionizing and dissociating gases. *Journal of Physics D-Applied Physics* **9** (12), 1681–1691.

Grun, J., Stamper, J., Manka, C., Resnick, J., Burris, R., Crawford, J. & Ripin, B. H. 1991 Instability of Taylor-Sedov blast waves propagating through a uniform gas. *Physical Review Letters* **66** (21), 2738–2741.

Helfand, E., Frisch, H. L. & Lebowitz, J. L. 1961 Theory of two- and one-dimensional rigid sphere fluids. *Journal of Chemical Physics* **34** (3), 1037–1042.

Hirschfelder, Joseph Oakland, Curtis, Charles F. & Bird, R. Byron 1964 *Molecular theory of gases and liquids*. New York: John Wiley.

Hornung, H. G. & Lemieux, P. 2001 Shock layer instability near the Newtonian limit of hypervelocity flows. *Physics of Fluids* **13** (8), 2394–2402.

Kamenetsky, V., Goldshtein, A., Shapiro, M. & Degani, D. 2000 Evolution of a shock wave in a granular gas. *Physics of Fluids* **12** (11), 3036–3049.

Kapper, M. G. & Cambier, J. L. 2011 Ionizing shocks in argon. Part II: Transient and multi-dimensional effects. *Journal of Applied Physics* **109** (11).

Landau, L. D. & Lifshitz, E. M. 1987 *Fluid mechanics*, 2nd edn. Butterworth-Heinemann.

Mishin, G. I., Bedin, A. P., Yushchenkova, N. I., Skvortsov, G. E. & Ryazin, A. P. 1981 Anomalous relaxation and the instability effect of shock waves in gases. *Zhurnal Tekhnicheskoi Fiziki* **51** (11), 2315–2324.

Mitrano, Peter P., Dahl, Steven R., Cromer, Daniel J., Pacella, Michael S. & Hrenya, Christine M. 2011 Instabilities in the homogeneous cooling of a granular gas: A quantitative assessment of kinetic-theory predictions. *Physics of Fluids* **23** (9), 093303.

Mond, M., Rutkevich, I. & Toffin, E. 1997 Stability of ionizing shock waves in monatomic gases. *Physical Review E* **56** (5), 5968–5978.

Mulero, A, Galan, C. A., Parra, M. I. & Cuadros, F. 2008 Equations of state for hard spheres and hard disks. *Lecture Notes in Physics* **753**, 37–109.

Pöschel, Thorsten, Brilliantov, Nikolai V & Schwager, Thomas 2005 Transient clusters in granular gases. *Journal of Physics: Condensed Matter* **17** (24), S2705.

Pöschel, Thorsten & Schwager, Thomas 2005 *Computational granular dynamics: models and algorithms*. Berlin New York: Springer-Verlag.

Semenov, A., Berezkina, M. & Krassovskaya, I. 2012 Classification of pseudo-steady shock wave reflection types. *Shock Waves* **22**, 307–316, 10.1007/s00193-012-0373-z.

Sirmas, Nick, Tudorache, Marian, Barahona, Javier & Radulescu, Matei Ioan 2012 Shock waves in hard disk fluids. *Shock Waves* .

Swinney, H. L. & Rericha, E. C. 2004 Pattern formation and shocks in granular gases. *Physics of Complex Systems (New Advances and Perspectives)* **155**, 173–204.

Tudorache, Marian 2008 B.A.Sc. Thesis: Shockwave dynamics using event driven theory.

Whitham, G. B. 1974 *Linear and Nonlinear Waves*. John Wiley and Sons.

Williams, F. A. 1985 *Combustion Theory*. Menlo Park, California: The Benjamin/Cummins Publishing Company.

Zeldovich, Ia B. & Raizer, Iu P. 1966 *Physics of shock waves and high-temperature hydrodynamic phenomena*. New York: Academic Press.

Appendix A

Code

The following is the code in C++ that has been implemented to 2-D shock waves through hard disk media.

```
#include <cmath>
#include <fstream>
#include <iostream>
#include <map>
#include <sstream>
#include <vector>
#include <ctime>
#include <cstdlib>
#include <string>
#include <math.h>
#include <sys/stat.h>

using namespace std;

double max_impact=10;   //threshold impact velocity
double piston_velocity=12.5;  //for piston velocity in terms of speed
double packing1=0.012;   //packing factor of reactive medium
double Z1=1+(2*packing1-0.8084*packing1*packing1+.0448*packing1*packing1*packing1)
    /(1-1.9682*packing1+0.9716*packing1*packing1);

const int N = 30000;    // Number of overall particles (must be set as constant)
double ratio=1;     //ratio of A/B particles
int N1=N/2;     //number of particles of type A
int N2=N1/ratio;    //number of particles of type B

unsigned long nstep = 1130000000; // Total number of collisions to be computed
```

```
const int nps =10000;   // Interval for the output of snapshots of the simulation in
    Postscrip format, nprint=1000, noverlap=10000;
const int noverlap = 5000;  // Interval for the test if particles overlap
//const double Q=-50;   // Reactive Energy/collapse
const double eps=0.80 ;      //coefficient of restitution
const double hp = 20;   // the height of the piston
const double dp = 120;   // the diameter of the piston
const double LboxY = 250;  // Size of simulation area. The area ranges from -Lbox to
    Lbox in both directions
const double LboxXR = 2500;  // Right size of simulation area.
const double pssize = 500;  // Scaling factor for the snapshots
const double pi = 3.1415926535897932; // pi number
const double infty = 1e20;  // Very large number which is interpreted by the program as
     infinity
double tol = 1e-10;   // Very small number which is used to prevent numerical errors
double velocityMax = 1.0;  //maximum initial velocity for initialization of system
double ds = 0.025;
//const double R = 1;   // Particle radius
double R = sqrt(packing1*4*LboxY*LboxXR/((N1+N2)*pi)); //particle radius as a function
    of packing factor



int pspage = 0;
int it;
double elastic_collisions=0;
double reacted_collisions=0;


double xp = 0.0;    // the displacement of the piston
double LboxXL = LboxXR - xp;  // Left size of simulation area.
double LboxX = (LboxXL + LboxXR)/2;

double Vmpt, Vrmst, Vmeant;   //the three theoretical typical speeds
double Vmpn, Vrmsn, Vmeann;   //the three theoretical typical speeds
int countMax = 0;
int countM = 0;
double speedPiston = 0.0;
double gamma1 = 2.0;     //gas monoatomic in 2D1
//double gamma1 = 1.4;     //gas monoatomic in 3D
double speedSound;
//parametersat equilibrium
double Abox = (LboxXR + LboxXR) * 2.0 * LboxY;
```

```
double lambda = 1/(4.0*sqrt(2.0)*R*(N1+N2)/Abox); //Mean Free Path
double pstep = 100;
double stepX = 10;                    //strip length x direction
double stepY = 10;                    //strip length y direction
double meanFreeTime;
double Temperaturet = 0.0; //temperature represents the k*T/m theoretic
double Pressuret = 0.0; //temperature represents the P/m theoretic
double Temperaturen = 0.0; //temperature represents the k*T/m numeric
double Pressuren = 0.0; //temperature represents the P/m numeric
long int totalPairCollision = 0;
long int totalWallCollision = 0;
double timeEq=0.0;
double pistonStep=0.0;
double meanFreeTimeN = 0.0;
double pistonPosition = 0.0;
int whatColide = 0; //0-left(piston), 1-right wall, 2-top wall, 3 - bottom wall, 4 -
    pairwise
int initeps = 0;


// GENERAL VARIABLES
int reacted[N];                     //defines whether particle has reacted
int speedcount[N];  //speed counts
int density_counter[Nstrip][Nsy]; //the counter for density
double vx2D_sum[Nstrip][Nsy];
double vy2D_sum[Nstrip][Nsy];
double vx2_2D[Nstrip][Nsy];
double vy2_2D[Nstrip][Nsy];
double Pressure2D[Nstrip][Nsy];
double Temperature2D[Nstrip][Nsy];

vector<double> velocity_sum(N); //the sum of velocity along a strip during the shock
    wave
vector<double> velocity_squared(N);
vector<double> vx_sum(N);
vector<double> vy_sum(N);
vector<double> vx2(N);
vector<double> vy2(N);
vector<double> x(N), y(N); // Stores X and Y positions of every particle
vector<double> vx(N),vy(N);  // Stores X and Y speeds of every particle
vector<double> clist[N];  // Collision schedule
vector<double> fvt(N),probt(N), fvn(N), probn(N), vt(N), vn(N);  // Stores the values
    for the probability function and the probability
```

```
vector<double> speed(N);  //the value of the counted speed
double Time = 0;
typedef map<double, pair<int,int> > ctypes; // Renamed from ctype
ctypes cseq;
ofstream fenergy("kinenergy");
ofstream distribution;

//Sub-routines
void init(double); //initializes the particles positions and velocities
void propagation(double); //advances particles and piston to next computed event
void collision(pair<int,int>); //calculates the time for collision between two
    particles
void wcollision(int); //calculates the minimum time to collision between a particle and
     a boundary
void update(int); //updates the schedule of times for events
bool checkoverlap(int, int); //checks for overlap between particles or with particles
    and a boundary
double ppcoll(int,int,double); //calculates post-collision velocity between two
    colliding particles
double pwcoll(int,double); //calculates post-collision velocity after a particle
    collision with a boundary
double kinenergy();
void velocityTypical(double); //calculates the numerical mean free time of the
    particles
double maxVelocity(); //calculates the maximum velocity in the system

void BolzmannMaxwell(); //computess the Maxwell Boltzmann distribution and outputs to .
    txt
void svgplot(int); //outputs .svg with particle distribution and layout of domain
void ShockParameters(double, int, double); //outputs the 1-d distribution of shock
    properties
void DxDy(double, int, double); //outputs the 2-d distribution of shock properties
double round(double, unsigned int);

double ranf(double x){ return (2*double(rand())/(1+double(RAND_MAX))-1)*x;} //function
    to randomize initil positions and trajectories of particles


int main()
{
  double tnext, tn=0.0;
  double v12;
       srand ( time(NULL) );
```

```
  int it = 0;
  char xr = 'C';
  pair<int,int> ijnext;

  init(tol); //initialization of the system
  BolzmannMaxwell();
  svgplot(pspage);
  DxDy(stepX, pspage, Time + timeEq);
  ShockParameters(stepX, pspage, Time + timeEq);
  pspage++;
  cout << "Piston position: " << fabs(LboxXR - LboxXL) << " time= " << Time + timeEq <<
       " Number of pair collisions: " << totalPairCollision << " Number of wall
      collisions: " << totalWallCollision << endl;
  cout << " Number of reactive collisions: "<< reacted_collisions<< endl;
  cout << "
      ================================================================================"
      << endl;
  do
  {
//    if(it == nps)
//    {
//    }

    tnext=cseq.begin()->first;
    ijnext=cseq.begin()->second;

    if(ijnext.second==-1)
    {
      tn=pwcoll(ijnext.first,tol);

      if (tn == infty)
        tn = tnext;

      propagation(tn-Time);

      Time=tnext;
      wcollision(ijnext.first);
      update(ijnext.first);
      totalWallCollision++;
    }
    else
    {
      tn=ppcoll(ijnext.first,ijnext.second,tol);
```

```
    meanFreeTimeN += tn-Time;
    v12 = fabs(sqrt(vx[ijnext.first] * vx[ijnext.first] + vy[ijnext.first] * vy[
        ijnext.first]) -
      sqrt(vx[ijnext.second] * vx[ijnext.second] + vy[ijnext.second] * vy[ijnext.
          second]));

    if (tn == infty)
      tn = tnext;

    propagation(tn-Time);
    Time=tnext;
    collision(ijnext);
    update(ijnext.first);
    update(ijnext.second);
    totalPairCollision ++;
  }

  if(xr=='C')
  {
    velocityTypical(ds);
    if(Time >= 10*meanFreeTime)              //piston commences after a prescribed
        number of mean free times
    {

      { DxDy(stepX, pspage, Time + timeEq);
        meanFreeTimeN = meanFreeTimeN/totalPairCollision;
                        cout << meanFreeTime <<endl;
        //The equilibrium was reached
        cout << "THE EQUILIBRIUM WAS REACHED AFTER " << Time << " TIME UNITS - START
            THE PISTON" << endl;
        cout << "MEAN FREE TIME: " << meanFreeTimeN << endl;
        xr = 'S';
        BolzmannMaxwell();
        svgplot(pspage);
        xr = (char)toupper(xr);
        speedPiston = 0.001;
        it = 0;
      }

    }
  }

  if (((it == nps) && it > 0) || speedPiston >0)
```

```
{
  if(xr=='C')
  { DxDy(stepX, pspage, Time + timeEq);
    ShockParameters(stepX, pspage, Time + timeEq);
    BolzmannMaxwell();
    svgplot(pspage);
    xr = (char)toupper(xr);
    it = 0;
  }
  switch ( xr )
  {
    case 'C' :
      pspage ++;
      break;
    case 'S' :
      if(gamma1 == 2.0)
        speedSound = Vrmsn;
      xp = LboxXR*2;
      speedPiston = piston_velocity;
      xr = 'F';
      timeEq = Time;
      Time = 0.0;
      cseq.clear();
      for(int i=0; i<N1+N2; i++) update(i);
      pspage ++;

      break;
    case 'F' :

      if (totalPairCollision==nstep || fabs(LboxXR - LboxXL)>=4700 || pspage==480)
      { cout<<"The simulation ended after "<< totalPairCollision<<" collisions"<<
          endl;
        return 0;
      }

      if(speedPiston == 0)
      {
        cout << "THE PISTON FINISHED ITS MOTION AFTER " << Time << " TIME UNITS -
            STOP THE APPLICATION" << endl;
        return 0;
      }
      if(pistonStep >= pstep)
      { DxDy(stepX, pspage, Time + timeEq);
```

```
            svgplot(pspage);
            ShockParameters(stepX, pspage, Time + timeEq);
            pspage ++;
            //check = checkoverlap(ijnext.first, ijnext.second);
            cout << "Piston position: " << fabs(LboxXR - LboxXL) << " time= " << Time +
                timeEq << " Number of pair collisions: " << totalPairCollision << "
                Number of wall collisions: " << totalWallCollision << endl;
            cout << " Number of reactive pair collisions: "<< reacted_collisions<< endl;
            cout << "
                ================================================================================
                " << endl;
            it = 0;
            pistonStep = 0.0;
          }
          break;
        default :
          return 0;
      }
    }
    it++;
  }while(true);
}

void update(int i)
{
  double ct;

  for(unsigned int ii=0; ii!=clist[i].size(); ii++) cseq.erase(clist[i][ii]);
  clist[i].clear();

  for(int j=0; j<N1+N2; j++)
  {
    if(i!=j)
    {
      ct = ppcoll(i, j, tol);
      if(ct < infty)
      {
        cseq[ct]=pair <int, int> (i,j);
        clist[i].push_back(ct);
        clist[j].push_back(ct);
      }
    }
  }
```

```
  ct=pwcoll(i, tol);
  if(ct < infty)
  {
    cseq[ct]=pair<int, int>(i,-1);
    clist[i].push_back(ct);
  }
}

void propagation(double tprop)
{
  //move the piston
  if((LboxXR - LboxXL < xp) && (speedPiston>0))
  {


    LboxXL -= tprop*speedPiston;
    pistonStep += tprop*speedPiston;
    pistonPosition += tprop*speedPiston;
    Abox = (LboxXR + LboxXR) * 2.0 * LboxY;
  }

  //move molecules
  for(int i=0; i<N1+N2; i++)
  {
    x[i]+=tprop*vx[i];
    if(x[i] + LboxXL <= LboxXR - x[i])
    {
      if(x[i] - R <= -LboxXL)
      {
        x[i] = -LboxXL + R;
      }
    }
    else //check for right side
    {
      if(x[i] + R >= LboxXR)
      {
        x[i] = LboxXR - R;
      }
    }

    y[i]+=tprop*vy[i];
    if(y[i]<=0) //check for top
```

```
   {const int Nstrip = 2000;
const int Nsy=100;
     if(y[i] - R <= -LboxY)
     {
       y[i] = -LboxY + R;
     }
   }
   else //check for right side
   {
     if(y[i] + R >= LboxY)
     {
       y[i] = LboxY - R;
     }
   }

  }
}

bool checkoverlap(int k1, int k2)
{
  double dist;
  double xi; double xj;
  double yi; double yj;


  for(int i=1; i<N1+N2; i++)
  {
    for(int j=0; j<i; j++)
    {
      xi = x[i]; xj = x[j];
      yi = y[i]; yj = y[j];
      if (!( ((k1 == i) && (k2 == j)) || ((k1 == j) && (k2 == i))))
      {
        dist=sqrt((xi-xj)*(xi-xj)+(yi-yj)*(yi-yj))-2*R;
        if(dist<0)
        {
          cout <<" OVERLAP "<<i<<" "<<j<< " " << dist << endl;
          return true;
        }
      }
    }
  }
  for(int i=0; i<N1+N2; i++)
```

```
  {
    if((xi-R <-LboxXL) || (yi-R < -LboxY) || (xi+R > LboxXR) || (yi+R>LboxY))
    {
      xi = x[i];
      yi = y[i];
      cout<<"OVERLAP WALL "<<i<< " y="<<y[i]+R << " x=" <<x[i]+R<< endl;
      return true;
    }
  }
  return false;
}
void collision(pair<int,int> ij)
{
  int i=ij.first;
  int j=ij.second;
  double xi = x[i]; double yi = y[i];
  double xj = x[j]; double yj = y[j];
  double vxi = vx[i]; double vyi = vy[i];
  double vxj = vx[j]; double vyj = vy[j];

  double dx = xi-xj;
  double dy = yi-yj;
  double dist=sqrt(dx*dx+dy*dy);
  if (2*R <= dist + tol)
    dist = 2*R;
  double ndx=dx/dist;
  double ndy=dy/dist;
  double h=((vx[i]-vx[j])*ndx+(vy[i]-vy[j])*ndy)/2;
  double relative_norm=sqrt(h*h*4);
  double mult;

  if (relative_norm>max_impact)
  {   mult=(1+eps);
    reacted_collisions++;

  }

  else { elastic_collisions++;
  mult=2;}

  vx[i]-=mult*h*ndx;
  vy[i]-=mult*h*ndy;
  vx[j]+=mult*h*ndx;
```

```
  vy[j]+=mult*h*ndy;

  vxi = vx[i]; vyi = vy[i];
  vxj = vx[j]; vyj = vy[j];

  if(LboxXR - LboxXL >= xp && speedPiston > 0)
  {
    //stop the piston
    speedPiston = 0.0;
  }

}
void wcollision(int i)
{
  double xi = x[i]; double yi = y[i];
            //the case when the ball is upper(-) or lower wall(+)
  if  (((fabs(xi - R + LboxXL) <= tol) && (fabs(yi - R + LboxY) <= tol)) || //the case
      when the ball is left upper corner - both are negative
    ((fabs(xi - R + LboxXL) <= tol) && (fabs(yi + R - LboxY) <= tol)) || //the case
        when the ball is left lower corner - left(-) lower (+)
    ((fabs(xi + R - LboxXL) <= tol) && (fabs(yi - R + LboxY) <= tol)) || //the case
        when the ball is right upper corner - right(+) upper (-)
    ((fabs(xi + R - LboxXL) <= tol) && (fabs(yi + R -LboxY) <= tol))) //the case when
        the ball is right lower corner - right(+) lower (+)
  {
    vx[i] = -vx[i];
    vy[i] = -vy[i];
  }
  else
  {
    switch(whatColide)
    {
      case 0:
        {
          vx[i] = 2.0 * speedPiston - vx[i];
          break;
        }
      case 1:
        {
          vx[i] = - vx[i];
          break;
        }
      case 2:
```

```
      {
        vy[i] = - vy[i];
        break;
      }
    case 3:
      {
        vy[i] = - vy[i];
        break;
      }
    default:
      cout << "Unknown collision" << endl;
  }

 }

 if(LboxXR - LboxXL -xp >= tol && speedPiston > 0)
 {
  //stop the piston
  speedPiston = 0.0;
 }
}
double pwcoll(int i,double tol)
{
 double tx, ty, txr, txl;
 double RR = R;
 double xi = x[i]; double yi = y[i];
 double vxi = vx[i]; double vyi = vy[i];
 int whatColideH, whatColideV;

 if((LboxXL+xi>R) || (LboxXR-xi<R) || (LboxY+yi>R) || (LboxY-yi<R))
 {
  RR=R-tol;
 }

 if(vxi==0)
 {
  tx=infty;
 }
 else
 {
  if(vxi >0)
  {
    txl = (LboxXL+xi-RR)/(-vxi+speedPiston); // the time to reach the piston
```

```
    txr = (LboxXR-xi-RR)/(vxi); //the time to reach the right wall
    if((txl > 0) && (txr > 0))
      if(txl >= txr)
      {
        tx = txr;
        whatColideH = 1; //0-left(piston), 1-right wall, 2-top wall, 3 - bottom wall,
            4 - pairwise
      }
      else
      {
        tx = txl;
        whatColideH = 0; //0-left(piston), 1-right wall, 2-top wall, 3 - bottom wall,
            4 - pairwise
      }
    else
      if ((txl > 0) && (txr<0))
      {
        tx = txl;
        whatColideH = 0; //0-left(piston), 1-right wall, 2-top wall, 3 - bottom wall,
            4 - pairwise
      }
      else
        if ((txr > 0) && (txl<0))
        {
          tx = txr;
          whatColideH = 1; //0-left(piston), 1-right wall, 2-top wall, 3 - bottom wall
              , 4 - pairwise
        }
        else
        {
          tx=infty;
          whatColideH = -1; //-1 - no collision, 0-left(piston), 1-right wall, 2-top
              wall, 3 - bottom wall, 4 - pairwise
        }

  }
  else
  {
    txl = (LboxXL+xi-RR)/(-vxi+speedPiston); // the time to reach the piston
    txr = infty;

    if(txl > 0)
    {
```

```
      tx = txl;
      whatColideH = 0; //0-left(piston), 1-right wall, 2-top wall, 3 - bottom wall, 4
          - pairwise
    }
    else
    {
      tx = infty;
      whatColideH = -1; //0-left(piston), 1-right wall, 2-top wall, 3 - bottom wall, 4
           - pairwise
    }

  }
}

if(vyi==0)
{
  ty=infty;
}
else
{
  if(vyi >0)
  {
    ty=(LboxY-yi-RR)/vyi;
    whatColideV = 3; //0-left(piston), 1-right wall, 2-top wall, 3 - bottom wall, 4 -
         pairwise
  }
  else
  {
    ty=(-LboxY-yi+RR)/vyi;
    whatColideV = 2; //0-left(piston), 1-right wall, 2-top wall, 3 - bottom wall, 4 -
         pairwise
  }
}
if(tx<=ty)
{
  whatColide = whatColideH;
}
else
{
tx = ty;
whatColide = whatColideV;
}
return Time+tx;
```

```
}
double ppcoll(int i, int j, double tol)
{
  double dx,dy,dvx,dvy,xci,xcj,yci,ycj,scalar,q,w,ct,vv2;
  double xi = x[i]; double yi = y[i];
  double xj = x[j]; double yj = y[j];
  double vxi = vx[i]; double vyi = vy[i];
  double vxj = vx[j]; double vyj = vy[j];

  dx=xi-xj;
  dy=yi-yj;
  dvx=vxi-vxj;
  dvy=vyi-vyj;
  scalar=dx*dvx+dy*dvy;
  if(scalar>=0)
  {
    return infty;
  }
  else
  {
    double dist2 = dx*dx+dy*dy;
    double RR= (sqrt(dist2)>=R+R ? R+R : R+R-2*tol);
    vv2 = dvx * dvx + dvy * dvy;
    q = dist2 - RR*RR;
    w = scalar * scalar - q * vv2;

    if(w<0)
    {
      whatColide = -1;
      return infty;
    }
    else
    {
      ct=Time+q/(-scalar+sqrt(w));
      xci=xi+(ct-Time)*vxi;
      xcj=xj+(ct-Time)*vxj;
      if((xci>LboxXR-R)||(xci<-LboxXL+R)||(xcj>LboxXR-R)||(xcj<-LboxXL+R))
      {
        return infty;
        whatColide = -1;
      }
      else
      {
```

```
        yci=yi+(ct-Time)*vyi;
        ycj=yj+(ct-Time)*vyj;
        if((yci>LboxY-R)||(yci<-LboxY+R)||(ycj>LboxY-R)||(ycj<-LboxY+R))
        {
          return infty;
          whatColide = -1;
        }
        else
        {
          whatColide = 4;
          return ct;
        }
      }
    }
  }
}

void init(double tol)
{
  bool overlap;
  double angle;
  int j;

  //generate the first atom
  x[0]=ranf(LboxX-R-tol);
  y[0]=ranf(LboxY-R-tol);

  //generate the speed for the first atom
  angle = ranf(pi * 2.0);
  //vx[0]=velocityMax* 5 * cos(angle);
  //vy[0]=velocityMax* 5 * sin(angle);
  reacted[0]=1;
  for(int i=1; i<N1+N2; i++)
  {
    if(!(i % nps )) cout << "Init "<< i << endl;
    do
    {
      angle = ranf(pi * 2.0);
      overlap=false;
      x[i]=ranf(LboxX-R-tol);
      y[i]=ranf(LboxY-R-tol);
      j=0;
      do
```

```
      {
        //check for overlap
        overlap=((x[i]-x[j])*(x[i]-x[j])+(y[i]-y[j])*(y[i]-y[j])<4*(R+tol)*(R+tol));
      }while((++j<i) && !overlap);
    }while(overlap);

    vx[i]= velocityMax * cos(angle);
    vy[i]= velocityMax * sin(angle);
                          if (i>N1)
                          reacted[i]=0;
                          else
                          reacted[i]=1;}
  for(int i=0; i<N1+N2; i++) update(i);
}
double kinenergy() {
  double ekin=0;

  for(int i=0; i<N1+N2; i++)
  {ekin+=vx[i]*vx[i]+vy[i]*vy[i];
  }
  return ekin;
}

void svgplot(int Page)
{
  // Create a SVG file
  string leading;
  double vv;
  int j;

  if (Page < 10)
  {
    leading = "00000";
  }
  else
  {
    if (Page < 100)
    {
      leading = "0000";
    }
    else
    {
      if (Page < 1000)
```

```
    {
      leading = "000";
    }
    else
    {
      if (Page < 10000)
      {
        leading = "00";
      }
      else
      {
        if (Page < 100000) leading = "0";
      }
    }
  }
}

stringstream ss;
ss << "event" << leading << Page << ".svg" << endl;
string filename;
ss >> filename;
ofstream fps(filename.c_str());   // ps file
// file opened?
if (! fps) {
  // NO, abort program
  cerr << "can't open output file \"" << filename << "\"" << endl;
  exit (EXIT_FAILURE);
}
fps <<
  "<?xml version=\"1.0\" standalone=\"no\"?>" <<
endl;

fps <<
  "<!DOCTYPE svg PUBLIC \"-//W3C//DTD SVG 1.1//EN\" \"http://www.w3.org/Graphics/SVG
      /1.1/DTD/svg11.dtd\">" <<
endl;

fps << "<svg width=\"100%\" height=\"100%\" version=\"1.1\" xmlns=\"http://www.w3.org
    /2000/svg\">" <<
  endl;

fps << "<rect x=\"10\" y=\"2\" width=\""<< LboxXR*2+24<< "\" height=\""<<LboxY*2+4<<"
    \" style=\"fill:white;stroke:black;stroke-width:4;fill-opacity:1\"/>" <<
```

```
  endl;

fps << "<rect x=\"" << 10 + LboxXR - LboxXL
  << "\" y=\"6\" width=\"20\" height=\""<<LboxY*2-4<<"\" style=\"fill:black;stroke:
      black;stroke-width:4;fill-opacity:1\"/>" <<
  endl;

for (int i = 0; i <N1+N2; i++)
{
  vv = sqrt(vx[i]*vx[i] + vy[i]*vy[i]);



  if(reacted[i]==0)
    j = 0;
  else if (reacted[i]==1)
    j = 1;
  else if (reacted[i]==2)
    j = 2;
    else
    j = 3;



  switch(j)
  {

    case 0:
      fps <<
        "<circle cx=\"" << x[i] + LboxXR + 32 <<
          "\" cy=\"" << y[i] + LboxY + 4 <<
          "\" r=\"" << R <<
          "\" fill=\"black\"/> " <<
      endl;
      break;
    case 1:
      fps <<
        "<circle cx=\"" << x[i] + LboxXR + 32 <<
          "\" cy=\"" << y[i] + LboxY + 4 <<
          "\" r=\"" << R <<
          "\" fill=\"black\"/> " <<
      endl;
      break;
```

```
    case 2:
      fps <<
        "<circle cx=\"" << x[i] + LboxXR + 32 <<
          "\" cy=\"" << y[i] + LboxY + 4 <<
          "\" r=\"" << R <<
          "\" fill=\"black\"/> " <<
      endl;
      break;
    case 3:
      fps <<
        "<circle cx=\"" << x[i] + LboxXR + 32 <<
          "\" cy=\"" << y[i] + LboxY + 4 <<
          "\" r=\"" << R <<
          "\" fill=\"black\"/> " <<
      endl;
      break;

  }

}

//generate the axis
//print typical velocities
//cout << "Piston position: " << fabs(LboxXR - LboxXL) << " time= " << Time + timeEq
    << " Number of colitions: " << totalCollision << endl;

fps << "<text x=\"" << 10
  << "\" y=\"" << LboxY * 2 + 20
  << "\" font-family=\"Verdana\" font-size=\"12\" fill=\"blue\" > Piston position= "
  << fabs(LboxXR - LboxXL) << " time = " << Time + timeEq <<//" Number of pair
      collisions: " << totalPairCollision << " Number of wall collisions: " <<
      totalWallCollision <<
  " Inelastic Collision: "<<reacted_collisions<<" Elastic Particle Collisions: "<<
      elastic_collisions<<"</text>"
  <<endl;

  fps << "<text x=\"" << 10
  << "\" y=\"" << LboxY * 2 +35
  << "\" font-family=\"Verdana\" font-size=\"12\" fill=\"blue\" > Piston Velocity: "
  <<piston_velocity<<"  Packing Factor: "<< ((N1+N2)*pi*R*R)/(LboxXR*LboxY*4)<<"
      Reacting Impact Velocity: "<<max_impact<<" </text>"
  << endl;
```

```
  fps << "</svg>" << endl;
  fps.close();
  //cout << "OK";
}

void velocityTypical(double _ds)
{
  double v = 0.0;
  double _vmeann = 0.0;
  double _vmpn = 0.0;
  int m = 0;
  int j1 = 0;
  vector<double> vn(N);
  vector<int> _speedcount(N);
  countMax = 0;

  do
  {
    v = vx[j1]*vx[j1] + vy[j1]*vy[j1];
    m = (int)ceil((sqrt(v)/_ds));
    if((double)m * _ds == sqrt(v))
    {
      m++;
    }
    if( sqrt(v) == 0)
      m=1;

    _speedcount[m]++;
    if(countM <= m && m !=0)
    {
      countM = m;
    }

    _vmeann += sqrt(v);
    j1++;
  }while(j1<N1+N2);


  //total probability is
```

```
  for(int m = 0; m<=countM; m++)
  {
    vn[m] = m *_ds;
    if(countMax <= _speedcount[m])
    {
      countMax = _speedcount[m];
      _vmpn = vn[m];
    }

  }
  _vmeann = _vmeann / double(N);
  Vmpn = _vmpn;
  Vmeann = _vmeann;

}

void BolzmannMaxwell()
{
  countMax = 0;
  double sum_fv, sum_prob, v;
  int i1 =0;
  int j1 = 0;
  int m = 0;
  int intStat;

  sum_fv = 0.0; sum_prob = 0.0;
  string leading;
  struct stat stFileInfo;


  for(int i = 0; i<N1+N2; i++)
  {
    speedcount[i] = 0;
    vt[i] = 0.0;
    vn[i] = 0.0;
    fvt[i] = 0.0;
    probt[i] = 0.0;
    probn[i] = 0.0;
    fvn[i] = 0;
  }

  stringstream ss;
  //ss << "bolzt" << leading << Page << ".txt" << endl;
```

```
ss << "BolzmannMaxwell_Theoretic.txt" << endl;
string filename;
ss >> filename;
ofstream distribution(filename.c_str());   // ps file

distribution << "This file contains theoretical values for Bolzmann-Maxwell
    probability density and probability function.\n" << endl;
distribution << "
    ==========================================================================================
    n" << endl;
distribution << endl;
distribution << "Theoretical distribution" << endl;
i1=0;
distribution << "Step (i) vt Probability Density Probability Sum of Probabilities" <<
     endl;
double _vmpt=0.0; double _vmeant = 0.0; double _vrmst = 0.0;
double _vmpn=0.0; double _vmeann = 0.0; double _vrmsn = 0.0;
double prob_max = 0.0; double probdens_max = 0.0;

do
{
  vt[i1] = i1 * ds; // theoretical velocity
  fvt[i1] = vt[i1] / pow(velocityMax,2.0) * exp(-pow(vt[i1],2.0) / pow(velocityMax
      ,2.0)); // probability function
  probt[i1] = exp(-pow(vt[i1],2.0)/pow(velocityMax,2.0)) - exp(-pow(vt[i1]+ds,2.0)/
      pow(velocityMax,2.0));
  sum_prob = sum_prob+probt[i1];
  if(probt[i1] > prob_max)
  {
    prob_max = probt[i1];
    _vmpt = vt[i1];
  }
  _vmeant += vt[i1]*probt[i1];
  _vrmst += vt[i1]*vt[i1]*probt[i1];
  distribution << i1 << " " << vt[i1] << " " << fvt[i1] << " " << probt[i1] << " " <<
       sum_prob << endl;
  i1 = i1 + 1;;
} while(i1>0 && sum_prob < 0.999 );
_vrmst = sqrt(_vrmst);
meanFreeTime = _vrmst * lambda;
distribution.close();

ss << "BolzmannMaxwell_Numerical.txt" << endl;
```

```
ss >> filename;
ofstream distribution1;

intStat = stat(filename.c_str(),&stFileInfo);
if(intStat == 0)
  distribution1.open(filename.c_str(), ios::out | ios::app);   // ps file
else
  distribution1.open(filename.c_str());

distribution1 << "The snapshot time: " << Time << endl;
distribution1 << "Mean Free Path = " << lambda << endl;
distribution1 << "Mean Free Time = " << meanFreeTime << endl;
distribution1 << endl;
distribution1 << "Step (i) Number of molecules Velocity Probability Density
    Probability Sum Probability" << endl;
distribution1 << endl;


//generate the numerical probability density and probability
j1=0;
sum_prob = 0;
countMax = 0;
countM = 0;
m = 0;
do
{
  v = vx[j1]*vx[j1] + vy[j1]*vy[j1];
  m = (int)ceil((sqrt(v)/ds));
  if((double)m * ds == sqrt(v))
  {
    m++;
  }
  if( sqrt(v) == 0)
    m=1;

  speedcount[m]++;
  if(countM <= m && m !=0)
  {
    countM = m;
  }

  probn[m] = (double)speedcount[m]/(double)N;
  fvn[m] = probn[m]/ (2*ds);
```

```
  _vrmsn += v;
  _vmeann += sqrt(v);
  j1++;
}while(j1<N1+N2);


//total probability is
prob_max = 0;
for(int m = 0; m<=countM; m++)
{
  vn[m] = m *ds;
  sum_prob = sum_prob + probn[m];
  if(countMax <= speedcount[m])
  {
    countMax = speedcount[m];
    _vmpn = vn[m];
  }

  distribution1 << m << " " << speedcount[m] << " " << vn[m] << " " << fvn[m] << " "
      << probn[m] << " " << sum_prob << endl;
}
_vmeann = _vmeann / double(N);
_vrmsn = sqrt(_vrmsn / (double)N);

Temperaturen = _vmeann * _vmeann / 2.0;
Pressuren = (N1+N2) * _vmeann * _vmeann /(2*Abox);

Temperaturet = _vmeant * _vmeant / 2.0;
Pressuret = (N1+N2) * _vmeant * _vmeant /(2*Abox);

distribution1 << endl;
distribution1 << " Most probable velocity Mean velocity Root mean square velocity
    temperature pressure" <<endl;
distribution1 << "Theoretic " << _vmpt << " " << _vmeant << " " << _vrmst << " " <<
    Temperaturet << " " << Pressuret << endl;
distribution1 << "Numeric " << _vmpn << " " << _vmeann << " " << _vrmsn << " " <<
    Temperaturen << " " << Pressuren << endl;
distribution1 << endl;

distribution1.close();
Vmpn = _vmpn;
Vrmsn = _vrmsn;
Vmeann = _vmeann;
```

```
  Vmpt = _vmpt;
  Vrmst = _vrmst;
  Vmeant = _vmeant;
}

double round(double val, unsigned int decimals)
{
  //ASSERT(val!=0);//val must be different from zero to avoid overflow!
  double sign = fabs(val)/val;//we obtain the sign to calculate positive always
  double tempval = fabs(val*pow((double)10, (double)decimals));//shift decimal places
  unsigned int tempint = (unsigned int)tempval;
  double decimalpart = tempval-tempint;//obtain just the decimal part
  if(decimalpart>=0.5)//next integer number if greater or equal to 0.5
    tempval = ceil(tempval);
  else
    tempval = floor(tempval);//otherwise stay in the current interger part

  return (tempval*pow((double)10,-(double)decimals))*sign;//shift again to the normal
      decimal places
}

double maxVelocity()
{
  double v_max = sqrt(vx[0]*vx[0] + vy[0]*vy[0]);
  for(int i = 1; i<N1+N2; i++)
  {
    if(sqrt(vx[i]*vx[i] + vy[i]*vy[i]) > v_max)
      v_max = sqrt(vx[i]*vx[i] + vy[i]*vy[i]);
  }
  return v_max;

}

void svgplot1(int Page)
{
  // Create a SVG file
  string leading;
  int j;
  double vv;
  //double _height;

  if (Page < 10)
  {
```

```
  leading = "00000";
}
else
{
  if (Page < 100)
  {
    leading = "0000";
  }
  else
  {
    if (Page < 1000)
    {
      leading = "000";
    }
    else
    {
      if (Page < 10000)
      {
        leading = "00";
      }
      else
      {
        if (Page < 100000) leading = "0";
      }
    }
  }
}

stringstream ss;
ss << "boltz" << leading << Page << ".svg" << endl;
string filename;
ss >> filename;
ofstream fps(filename.c_str());   // ps file
// file opened?
if (! fps) {
  // NO, abort program
  cerr << "can't open output file \"" << filename << "\"" << endl;
  exit (EXIT_FAILURE);
}
fps <<
  "<?xml version=\"1.0\" standalone=\"no\"?>" <<
endl;
```

```
  fps <<
    "<!DOCTYPE svg PUBLIC \"-//W3C//DTD SVG 1.1//EN\" \"http://www.w3.org/Graphics/SVG
        /1.1/DTD/svg11.dtd\">" <<
  endl;

  fps << "<svg width=\"100%\" height=\"100%\" version=\"1.1\" xmlns=\"http://www.w3.org
      /2000/svg\">" <<
    endl;

  //generate axes

  fps << "</svg>" << endl;
  fps.close();
  //cout << "OK";
}
void ShockParameters(double _stepX, int pspage, double time)
{
  //find the shock density
  int j = 0;
  double xi, vi, vmean, Pressure, Temperature, Density, xmax, xmin, xboundmin,
      xboundmax, vmean2, Z2, packing2;
  int maxM = 0;
  int intStat;
  int count;
  double average_impact;
  string leading;

  stringstream ss;
  string filename;
  struct stat stFileInfo;

  xmax = x[0];
  xmin = x[0];


  for(int i1= 0; i1<Nstrip; i1++)
  {
    density_counter[i1][0] = 0;
    vx_sum[i1]=0;
    vy_sum[i1]=0;
    vx2[i1]=0;
    vy2[i1]=0;
  }
```

```
for(int i = 0; i<N1+N2; i++)
{
  xi = x[i];
  if(xmax < xi)
    xmax = xi;
  if(xmin > xi)
    xmin = xi;

}
j = 0;
do
{
  xboundmin = xmin + (double)j * _stepX;
  xboundmax = xmin + (double)(j+1) * _stepX;
  if(xboundmax > xmax)
    xboundmax = xmax;
  for(int i = 0; i<N1+N2; i++)
  {
    xi = x[i];
    if(xi>=xboundmin && xi < xboundmax)
    { vx_sum[j]+=vx[i];
      density_counter[j][0] ++;
    }
  }
            for(int i = 0; i<N1+N2; i++)
  {
    xi = x[i];
    if(xi>=xboundmin && xi < xboundmax)
    { vx2[j]+=(vx[i]-vx_sum[j]/density_counter[j][0])*(vx[i]-vx_sum[j]/
        density_counter[j][0]);
      vy2[j]+=vy[i]*vy[i];
    }
  }
  j++;
}while(xboundmax < xmax);
maxM = j;

if (pspage< 10)
{
  leading = "00000";
}
else
```

```
{
  if (pspage < 100)
  {
    leading = "0000";
  }
  else
  {
    if (pspage< 1000)
    {
      leading = "000";
    }
    else
    {
      if (pspage< 10000)
      {
        leading = "00";
      }
      else
      {
        if (pspage < 100000) leading = "0";
      }
    }
  }
}

ss << "ShockParameters-"<<leading<<pspage<<".txt" << endl;
ss >> filename;
ofstream distribution1;

intStat = stat(filename.c_str(),&stFileInfo);
if(intStat == 0)
  distribution1.open(filename.c_str(), ios::out | ios::app);   // ps file
else
  distribution1.open(filename.c_str());

distribution1 << "The snapshot time: " << time << endl;
distribution1 << endl;
distribution1 << "Step (i) Number of molecules Density Ratio Temperature Ratio
    Pressure Ratio" << endl;
distribution1 << endl;


for(j = 0; j<maxM; j++)
```

```
    {
    vmean2 = ((vx2[j]/density_counter[j][0])-(vx_sum[j]/density_counter[j][0])*(vx_sum[
        j]/density_counter[j][0]))+((vy2[j]/density_counter[j][0])-(vy_sum[j]/
        density_counter[j][0])*(vy_sum[j]/density_counter[j][0]));

    Density = (density_counter[j][0] / (_stepX*2.0 * LboxY))/(N/(4*LboxY*LboxXR )) ;

    Temperature = (vx2[j]/density_counter[j][0])+ (vy2[j]/density_counter[j][0]);

    packing2=packing1*Density;

    Z2=1+(2*packing2-0.8084*packing2*packing2+.0448*packing2*packing2*packing2)
        /(1-1.9682*packing2+0.9716*packing2*packing2);

    Pressure = Temperature*Density*Z2/Z1;

    distribution1 << (j*_stepX/lambda) << " " << density_counter[j][0] << " " <<Density
        << " " << Temperature<< " " << Pressure << endl;

  count=0;
  average_impact=0;}
  distribution1 << endl;
  distribution1.close();
  NN=0;
  return;
}

void DxDy(double _stepX, int pspage, double time)
{
  //find the shock density
  int j = 0;
  int m = 0;
  double xi, yi, vi, vmean, Density, Pressure, Temperature, xmax, xmin, xboundmin,
      xboundmax, ymax, ymin, yboundmin, yboundmax, vmean2, packing2, Z2;
  int maxM = 0;
  int maxN = 0;
  int intStat;
  int count;
  double average_impact;

  stringstream ss;
  string filename1;
```

```
  string filename2;
  string filename3;
  string filename4;
  string filename5;
  struct stat stFileInfo;
string leading;

  xmax = x[0];
  xmin = x[0];
  ymax = y[0];
  ymin = y[0];

  for(int i1= 0; i1<Nstrip; i1++)
  { for(int j1= 0; j1<Nstrip; j1++)
    {density_counter[i1][j1] = 0;
    vx2D_sum[i1][j1]=0;
    vy2D_sum[i1][j1]=0;
    vx2_2D[i1][j1]=0;
    vy2_2D[i1][j1]=0;
}

  }

  for(int i = 0; i<N1+N2; i++)
  {
    xi = x[i];
    if(xmax < xi)
      xmax = xi;
    if(xmin > xi)
      xmin = xi;
    yi = y[i];
    if(ymax < yi)
      ymax = yi;
    if(ymin > yi)
      ymin = yi;
  }
  j = 0;
  do
  {
    xboundmin = xmin + (double)j * _stepX;
    xboundmax = xmin + (double)(j+1) * _stepX;

    if(xboundmax > xmax)
```

```
  xboundmax = xmax;

m=0;
do
{
for(int i = 0; i<N1+N2; i++)
{


  xi = x[i];
  yi = y[i];
  vi = sqrt(vx[i]*vx[i] + vy[i]*vy[i]);

  yboundmin = ymin + (double)m * stepY;
    yboundmax = ymin + (double)(m+1) * stepY;

    if(xi>=xboundmin && xi < xboundmax && yi>=yboundmin && yi < yboundmax)
    {density_counter[j][m] ++;
    vx2D_sum[j][m]+=vx[i];
    vy2D_sum[j][m]+=vy[i];}
}

for(int i = 0; i<N1+N2; i++)
{


  xi = x[i];
  yi = y[i];
  vi = sqrt(vx[i]*vx[i] + vy[i]*vy[i]);

  yboundmin = ymin + (double)m * stepY;
    yboundmax = ymin + (double)(m+1) * stepY;

    if(xi>=xboundmin && xi < xboundmax && yi>=yboundmin && yi < yboundmax)
    {vx2_2D[j][m]+=(vx[i]-vx2D_sum[j][m]/density_counter[j][m])*(vx[i]-vx2D_sum[j][m
        ]/density_counter[j][m]);
    vy2_2D[j][m]+=vy[i]*vy[i];}
}



m++;
}while(yboundmax<ymax);
```

```
    j++;
  }while(xboundmax < xmax);
  maxM = j;
  maxN = m;

  if (pspage< 10)
  {
    leading = "00000";
  }
  else
  {
    if (pspage < 100)
    {
      leading = "0000";
    }
    else
    {
      if (pspage< 1000)
      {
        leading = "000";
      }
      else
      {
        if (pspage< 10000)
        {
          leading = "00";
        }
        else
        {
          if (pspage < 100000) leading = "0";
        }
      }
    }
  }

//Density 2D//
  ss << "Density-"<<leading<<pspage<<".txt" << endl;
  ss >> filename1;
  ofstream distribution1;

  intStat = stat(filename1.c_str(),&stFileInfo);
  if(intStat == 0)
    distribution1.open(filename1.c_str(), ios::out | ios::app);
```

```
  else
    distribution1.open(filename1.c_str());



  for(m = 0; m<maxN; m++)
    {
    for(j = 0; j<maxM; j++)
        {


    Density = (density_counter[j][m] / (_stepX*stepY))/(N/(4*LboxY*LboxXR )) ;

    Temperature2D[j][m] = (vx2_2D[j][m]/density_counter[j][m])+ (vy2_2D[j][m]/
        density_counter[j][m]);

    packing2=packing1*Density;

    Z2=1+(2*packing2-0.8084*packing2*packing2+.0448*packing2*packing2*packing2)
        /(1-1.9682*packing2+0.9716*packing2*packing2);

    Pressure2D[j][m] = Temperature2D[j][m]*Density*Z2/Z1;




distribution1 << " " << Density;




}

    distribution1 <<endl;
}

  distribution1 << endl;
  distribution1.close();



//Pressure 2D//
```

```
  ss << "Pressure"<<leading<<pspage<<".txt" << endl;
  ss >> filename2;
  ofstream distribution2;

  intStat = stat(filename2.c_str(),&stFileInfo);
  if(intStat == 0)
    distribution2.open(filename2.c_str(), ios::out | ios::app);
  else
    distribution2.open(filename2.c_str());



  for(m = 0; m<maxN; m++)
    {
    for(j = 0; j<maxM; j++)
        {


distribution2 << " " << Pressure2D[j][m];




}

    distribution2 <<endl;
}

  distribution2 << endl;
  distribution2.close();


//Temperature 2D//
  ss << "Temperature"<<leading<<pspage<<".txt" << endl;
  ss >> filename3;
  ofstream distribution3;

  intStat = stat(filename3.c_str(),&stFileInfo);
  if(intStat == 0)
    distribution3.open(filename3.c_str(), ios::out | ios::app);
  else
    distribution3.open(filename3.c_str());
```

```
  for(m = 0; m<maxN; m++)
    {
    for(j = 0; j<maxM; j++)
        {


distribution3 << " " << Temperature2D[j][m];




}

    distribution3 <<endl;
}

  distribution3 << endl;
  distribution3.close();

//X-velocity 2D//
  ss << "VelX"<<leading<<pspage<<".txt" << endl;
  ss >> filename4;
  ofstream distribution4;

  intStat = stat(filename4.c_str(),&stFileInfo);
  if(intStat == 0)
    distribution4.open(filename4.c_str(), ios::out | ios::app);
  else
    distribution4.open(filename4.c_str());




  for(m = 0; m<maxN; m++)
    {
    for(j = 0; j<maxM; j++)
        {

if (density_counter[j][m]>0)
distribution4 << " " << vx2D_sum[j][m]/density_counter[j][m];
```

```
else
distribution4 << " " << 0;




}

    distribution4 <<endl;
}

  distribution4 << endl;
  distribution4.close();

//Y-velocity 2D//
  ss << "VelY"<<leading<<pspage<<".txt" << endl;
  ss >> filename5;
  ofstream distribution5;

  intStat = stat(filename5.c_str(),&stFileInfo);
  if(intStat == 0)
    distribution5.open(filename5.c_str(), ios::out | ios::app);
  else
    distribution5.open(filename5.c_str());




  for(m = 0; m<maxN; m++)
    {
    for(j = 0; j<maxM; j++)
        {

if (density_counter[j][m]>0)
distribution5 << " " << vy2D_sum[j][m]/density_counter[j][m];
else
distribution5 << " " << 0;




}
```

```
    distribution5 <<endl;
}

  distribution5 << endl;
  distribution5.close();


  NN=0;
  return;
}
```