QUANTIFYING NON-EQUILIBRIUM IN HYPERSONIC FLOWS USING ENTROPY GENERATION

THESIS

Ryan W. Carr, Second Lieutenant, USAF

AFIT/GAE/ENY/07-M07

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GAE/ENY/07-M07

QUANTIFYING NON-EQUILIBRIUM IN HYPERSONIC FLOWS USING ENTROPY GENERATION

THESIS

Presented to the Faculty

Department of Aeronautics and Astronautics

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Aeronautical Engineering

Ryan W. Carr, B.S.

Second Lieutenant, USAF

March 2007

AFIT/GAE/ENY/07-M07

QUANTIFYING NON-EQUILIBRIUM IN HYPERSONIC FLOWS USING ENTROPY GENERATION

Ryan W. Carr, BS
Second Lieutenant, USAF

Approved:

_____	_____
Major Richard Branam (Chairman)	Date
_____	_____
José Camberos (Member)	Date
_____	_____
Lt. Col. Raymond Maple (Member)	Date
_____	_____
Robert Greendyke (Member)	Date

Abstract

The constitutive relations traditionally used for finding shear stress and heat flux in a fluid become invalid in non-equilibrium flow. Their derivation from kinetic theory only demonstrates they are valid only for small deviations from equilibrium. Because it is fundamentally linked to non-equilibrium, entropy generation is used to investigate the limits of the continuum constitutive relations. However, the continuum equations are inherently limited to near equilibrium conditions due to the constitutive relations; thus kinetic theory may be used as a basis for comparison. Direct Simulation Monte Carlo (DSMC), a particle method alternative to continuum methods, is based on kinetic theory and is used to develop a flow solution for benchmark comparison.

Solutions were obtained for hypersonic flow over two axi-symmetric geometries using both a continuum solver and DSMC. Formulations for entropy generation are presented for each method, and the two solutions are compared. The continuum solutions fail to capture regions of non-equilibrium as evidenced by thicker shocks in the DSMC solution.

To extend the useful range of the continuum constitutive relations, the Lennard-Jones model is offered as an alternative to Sutherland's Law for calculating viscosity and thermal conductivity. The two are compared, and parameters offering a good fit for these flows are suggested for the Lennard-Jones model.

Acknowledgements

The author would like to thank Major Richard Branam (AFIT) and Dr. José Camberos (AFRL) for their guidance throughout the project. Their enthusiastic assistance on research, writing, presenting, and traveling have been invaluable. Funding for this project from AFRL/VA is gratefully acknowledged. Without the DSMC code supplied by Dr. Boyd at the University of Michigan the entire basis of this research would not have been possible. Additional thanks go to Timothy Holman on loan from U. Michigan for his expertise in DSMC.

Table of Contents

Page

List of Figures

Page

Page

List of Tables

Page

List of Symbols

h	Planck's constant = 6.62606873 x 10^{-34} J/K
k	Boltzmann's constant = 1.3806503 x 10^{-23} J/K
δ_{ij}	Kroneker delta
ε_{rot}	Rotational energy
ε_{vib}	Vibrational energy
λ	Mean free path; second coefficient of viscosity
μ	Kinematic viscosity coefficient
μ'	Kinematic viscosity at characteristic temperature T'
$\mu^{(1)}$	First approximation to the kinematic viscosity coefficient (Sonine expansion)
μ_B	Bulk viscosity coefficient
v	Lennard-Jones coefficient
ρ	Mass density
τ_{ij}	Shear stress tensor
$\tau_{ij(0)}$	Equilibrium shear stress tensor
$\tau_{ij(1)}$	First order perturbation of the shear stress tensor
ζ	Particle collision partner
Φ_1	First order perturbation from an equilibrium probability distribution
Ω	Total number of microstates of a system
A_i	Coefficient vector in first order perturbation term of Chapman-Enskog solution
A	Integral function in coefficient vector

B_{ij} Coefficient matrix in first order perturbation term of Chapman-Enskog solution

B Integral function in coefficient matrix

c_i Particle velocity in the *i* direction

Ci Thermal or peculiar velocity in the *i* direction

$\mathcal{C}_j$ Number of quantum states in the j^{th} energy grouping

Cp Coefficient of pressure

dV_C Differential segment of three dimensional velocity space

E Energy

f Probability distribution (or density) function

f_o Maxwellian (equilibrium) PDF

f_1 First order perturbation from the Maxwellian PDF

$\vec{f}$ Flux vector

F_i External force in the *i* direction

g Relative speed between two collision partners

K Thermal conductivity

$K^{(1)}$ First approximation to thermal conductivity (Sonine expansion)

Kn Knudsen number

Kn_S Knudsen-like entropy generation parameter

L Characteristic length

m Mass of a molecule

n Number density, particles per unit volume

N Total number of particles

N_j Number of particles in the j^{th} energy grouping

q_j Heat flux vector in the *i* direction

$q_{j(0)}$ Equilibrium heat flux vector in the *i* direction

$q_{j(1)}$ First order perturbation of the heat flux vector in the *i* direction

$\vec{q}$ State vector

Q Generic macroscopic variable

$\langle Q \rangle$ Expectation value of Q

S Entropy, Sutherland's constant

$\dot{S}_{gen}$ Entropy generation density

St Stanton number

T Temperature

T' Characteristic temperature

u_i Average velocity in the *i* direction

$\forall$ Volume

QUANTIFYING NON-EQUILIBRIUM IN HYPERSONIC FLOWS USING ENTROPY GENERATION

I. Introduction

On November 10, 2006 a multi-national research partnership agreement was signed by top scientists from the United States Air Force and the Australian government in Canberra, Australia. The Hypersonic International Flight Research Experimentation program (HiFire) pledges $54 million dollars to research focused solely on observing and understanding hypersonic flows. According to a recent article in the Air Force Research Laboratory newsletter, this research will enable the Air Force to exploit speed and responsiveness for a multitude of applications, anywhere from air-breathing hypersonic cruise missiles in the near term to operational space access in the far term (Barr, 2006). Researchers from around the world will participate in computational and ground test research which will culminate in ten experimental flights.

This program is evidence of the aerospace community's increasing interest in hypersonic flight. The Air Force has a particular interest in developing hypersonic vehicles and weapons. According to the Chief Scientist of the Air Force, Mark Lewis, "In modern air warfare, speed is the critical issue. I think hypersonics holds the potential for giving us that capability." (Honest Broker, 2007: 8). It is apparent from his statement that interest in hypersonics research reaches to the very top levels of defense planning. Despite this,

hypersonic flow is currently poorly understood, and in order to realize the dreams of the Air Force leadership, more research needs to be done.

Unfortunately, it is very difficult and expensive to experimentally study hypersonic flow. For this reason, computational methods have played a prominent role in hypersonics research. These methods facilitate the early stages of design and analysis, reducing the need for extensive experimentation and decreasing the risk in actual flight tests. Understanding non-equilibrium flow is key in improving the experimental methods currently being used in hypersonics research.

Non-Equilibrium Defined

To begin a study of non-equilibrium flow, it is important to first properly understand what is meant by equilibrium. Classical thermodynamics deals principally with systems that are at equilibrium. Specifically, the properties of a system in thermodynamic equilibrium will not change with time. The system is at a steady state and remains unchanged unless disturbed by an applied force, temperature gradient, or chemical reaction. Equilibrium thermodynamics tells nothing about the rate the system will then change; it can only describe the state to which it will arrive after the system has once again reached equilibrium. As the system adjusts from one state of equilibrium to another, it is said to be at non-equilibrium.

As an element of fluid at equilibrium travels through a flow field it experiences changes due to the field itself. This causes changes in the equilibrium state of the element. Often it is assumed the changes occur quickly compared to characteristic time scales of the flow,

and equilibrium may be assumed throughout the change. However, this is not a valid assumption when changes occur quickly and the characteristic time is small, such as in a shockwave standing off the body of a hypersonic vehicle. In this case, it is necessary to discard the assumption of equilibrium and develop a method of examining non-equilibrium processes in a flow.

To understand what is occurring as the system changes from one state of equilibrium to another, it is instructive to view the flow not as a continuum of mass, but rather as a collection of particles. Particles in a system at equilibrium constantly experience changes in translational and internal energy due to collisions with other particles. The collisions occur such that macroscopic variables like temperature and pressure, defined as an average of the energy or momentum of the molecules, experience no overall change. When an external process causes a change to occur to the molecules, the overall effect of the collisions is to cause the macroscopic variables to change. This occurs very quickly compared to the characteristic time of most flows.

However, as mentioned, in a shockwave, the characteristic time is much smaller. A molecule encountering a shock experiences drastic changes very quickly due to the increased frequency of collisions with other molecules and the high amount of energy exchanged in those collisions. Traveling downstream, the molecule continues to transfer energy to other molecules in the flow through collisions. Thus the macroscopic variables continue to change downstream as the molecules redistribute their energy. If the time between collisions is on the same order of magnitude as the time it takes to travel through

the shock, non-equilibrium changes to the macroscopic variables are significant and equilibrium-based thermodynamics will not adequately describe the flow in this region.

The thermodynamics normally used in the traditional continuum equations of fluid dynamics (Euler and Navier-Stokes) do little to compensate for non-equilibrium effects. The basic mechanics of the equations are correct; however, the constitutive relationships become invalid. These relationships seek to model bulk transfer of mass, momentum, or energy as a continuum rather than by a multitude of particle collisions. The shear stress tensor and the heat flux vector are two important constitutive relationships used to find closure for the Navier-Stokes equations. The previous discussion indicates these relationships are invalid for non-equilibrium flow. This point will be discussed further in conjunction with the Chapman-Enskog solution to the Boltzmann Equation in the Theory section of this work.

Continuum Methods versus Particle Methods

Because the continuum equations fail to properly model non-equilibrium phenomena, it is necessary to understand where in a flow these effects may be significant. Kinetic theory models fluids as a collection of particles rather than as a continuous mass. Therefore, it models the physics of non-equilibrium previously described. Many computational models base their development on this principle, especially as technological advances increase computing power. For example, one method known as Molecular Dynamics Simulation (MD) models each particle in a flow, tracking position, velocity, and internal energy. This method is computationally expensive due to the large number of particles required

to simulate most engineering applications. For this reason it is primarily limited to flows with few particles and collisions, such as flow over Micro Electro-Mechanical Systems (MEMS) or other very low-density flows.

Another method of modeling a system as a collection of particles more appropriate for engineering analysis is known as Direct Simulation Monte Carlo. This method decreases the computational load of MD models by representing a large number of particles ($\sim 1 \times 10^{12}$) by a single "virtual" particle. Collisions between virtual particles are determined statistically based on their position and velocity relative to one another. Translational energy is exchanged with internal energy during collisions. The specific details of choosing a collision pair, modeling energy transfer, and modeling wall collisions varies depending on the specific code. However, the post collision values of velocity and internal energy are normally drawn from a normal distribution. This method certainly introduces an assumption of equilibrium into the solution method. Nevertheless, a great deal of research indicates that DSMC is still able to capture non-equilibrium effects in a flow better than the Navier-Stokes equations (Ewin and others, 1989; Gallis and others, 2006).

By modeling the actual physics of the particles within the flow, DSMC and MD directly reflect physical reality, whereas the continuum equations seek to model reality by solving differential equations. However, the use of either particle method discussed here generally requires more computational effort than a continuum method.

Parameters Indicating Non-Equilibrium

Because continuum methods generally require fewer CPU hours, it is desirable to use them whenever possible. The ideal method would combine the computational efficiency of a continuum method with the fundamental physics of a kinetic method. Codes that switch between the two methods, called hybrid codes, are currently under development (Schwartzentruber and others, 2006). To use these codes it is necessary to predict whether a continuum method is adequate for the specific flow. As discussed above, non-equilibrium effects become important when the non-equilibrium relaxation time is on the same order as the characteristic time of the flow. This can also be analyzed in terms of length scales, i.e., when the distance between collisions is similar in magnitude to some characteristic length. The ratio of the two length scales is often given as the Knudsen number:

$$Kn = \frac{\lambda}{L} \tag{I.1}$$

If the Knudsen number is much smaller than one, the characteristic length is large compared to the distance in which collisions occur and the flow relaxes to equilibrium. The flow can be thought of as a continuous mass. If the number is much greater than one, collisions are less common, and the relaxation distance will be longer than the characteristic length of the flow and non-equilibrium effects could be significant. At this point the flow should be considered on the molecular scale. Knudsen numbers near one fall into a sort of transition regime. The Knudsen number may be examined for any flow regardless of Mach number; however, hypersonic vehicles are much more likely to

exhibit high Knudsen numbers because of the conditions in which they typically operate (high altitude).

Unfortunately the choice of the characteristic length scale is not clear. It could be based on a geometric structure like the nose of a hypersonic vehicle or a flow structure like the thickness of a normal shock. The choice of a length scale could result in a flow with Knudsen numbers ranging from very small (inviscid continuum limit) to very large (rarified gas – collisionless flow). For this reason, researchers have suggested other parameters to indicate the breakdown of the continuum assumption. Many of these mimic the Knudsen Number. Schrock gives an overview of these parameters in his thesis work (2005: 8-11). All of the parameters reviewed suggest the shear stress tensor and the heat flux vector are indicators of non-equilibrium. This is physically intuitive. A system which is truly at an equilibrium state will experience no shear or heat flux, as in, for example, a uniform flow field. Any shear or heat flux term will tend to change the state of the flow, thus causing it to experience some degree of non-equilibrium. This concept will be discussed in more detail in context of the Chapman-Enskog expansion in chapter two.

Entropy Generation as an Indicator of Non-Equilibrium

At this point it is convenient to introduce the formulation for entropy generation given by Camberos (2001: 6). A detailed discussion of this equation will be deferred to the theory section of this document:

$$\dot{S}_{gen} = \frac{\tau_{ji}}{T}\frac{\partial u_i}{\partial x_j} - \frac{q_j}{T^2}\frac{\partial T}{\partial x_j} \tag{I.2}$$

Contained within are the shear stress tensor and heat flux vector, the two indicators of non-equilibrium. This agrees with the result understood from the second law of thermodynamics which states a system goes from one equilibrium state to another through the creation of entropy. Further evidence of this is given by Boltzmann's famous H-Theorem, which relates entropy to a monotonically increasing function of the probability distribution of the velocity or energy state of a particle.

Entropy generation represents non-equilibrium. For this reason it has been studied as a parameter similar in function to the Knudsen number, to indicate the range of validity in the continuum equations. Schrock concluded that it is not possible to use the continuum equations when attempting to predict non-equilibrium based on entropy generation, as they are inherently limited to near-equilibrium and fundamentally cannot capture non-equilibrium effects. It is necessary to use a method based on kinetic theory.

Unfortunately, a parameter used to determine the validity of continuum equations that can only be calculated using a method based on kinetic theory has little practical use. If the computationally expensive kinetic method has already been performed, there is little need to study the limits of validity of the continuum equations. For this reason, entropy generation is not immediately useful in a hybrid kinetic-continuum code. However, understanding non-equilibrium behavior in a flow is not uniquely a problem for hybrid codes. Every scientist or engineer seeking a computational solution to a specific problem must understand the limits of the tools used. Non-equilibrium is a poorly understood limitation to many of the tools available in computational fluid dynamics (CFD).

The current work will use entropy generation as a means of quantifying non-equilibrium. Because it is directly tied to perturbations from an equilibrium state, entropy generation gives a glimpse of where the flow is most active and where the traditional formulations for the constitutive relations are no longer valid. It will indicate where macroscopic variables are no longer changing according to the equations normally defined by equilibrium methods. By quantifying non-equilibrium, it may be possible to modify the relationships for the shear stress tensor and the heat flux vector to better capture non-equilibrium, in somewhat the same way empirical models find closure in turbulent flows. Other research suggests this is possible, but has not benefited from an understanding of entropy generation as an indicator of non-equilibrium (Baganoff, 2002; Chen and others, 2001).

The above modifications in shear stress and heat flux will effectively extend the non-equilibrium range of the Navier-Stokes equations. To do this, a method comprising five steps is suggested:

1. Obtain a good solution using a kinetic-based solver. These types of solvers have traditionally been used to study non-equilibrium in the past, and have shown good results. Unfortunately, they require more computing resources than continuum solvers. The solution should match experimental data.

2. Use the flow solution to generate the shear stress tensor and the heat flux vector using both kinetic and continuum formulations. Use these constitutive relations to calculate the entropy generation in the flow. The entropy generation is indicative of regions of non-equilibrium.

3. Tune the formulations for the continuum constitutive equations until the continuum entropy generation matches the kinetic entropy generation. Entropy generation provides a scalar measure of the nine components of the shear and heating.

4. Use the new continuum formulations for shear and heating to resolve the flowfield. Compare the new solution with the previously attained kinetic flowfield using entropy generation. Check that the new solution predicts similar regions of non-equilibrium as the kinetic solution.

5. Apply the new continuum formulations to a number of other flow geometries and freestream conditions. Investigate the range of utility of the newly generated continuum formulations.

Using the above method will extend the range of the continuum Navier-Stokes equations to accurately capture regions of non-equilibrium beyond the traditional limits of validity. This would serve as an extremely useful tool to cut down on the need to perform many DSMC calculations to investigate hypersonic flows. A single DSMC calculation could be run, and based on the results, many accurate CFD solutions could be found for similar flows.

This research complements and follows the research done by Christopher Schrock (2005). Much of the theory necessary to understand the current study was presented in detail in Schrock's thesis. In order to avoid duplicating the information presented there, subjects treated in his work will be presented in abbreviated form here. However, many

additional concepts require attention for the current work, and these will now be reviewed in greater depth.

II. Theory

To understand non-equilibrium, it is instructive to review basic elements of kinetic theory. In kinetic theory, flow is not assumed to be a continuous mass, but rather a large collection of molecules. These molecules move about with some velocity, colliding with each other and exchanging momentum and energy. Because of the large amount of information that would be required to track each individual molecule, it is necessary to describe the state of these molecules probabilistically. This is done with the probability density function.

The Probability Density Function

Rather than attempting to track the exact position, velocity and internal energy of a particle, it is often convenient to instead speak of the probability a single particle will be found at a certain velocity or internal energy state. This can be described by a probability density function, f. It is possible to define a velocity space containing all three velocity vectors $dV_C = dC_1 dC_2 dC_3$. The probability of finding a molecule in a certain velocity in the range [V_C to $V_C + dV_C$] can be expressed by the integral:

$$\int_{V_C}^{V_C + dV_C} f dV_C \tag{II.1}$$

This means if the integral is over all the possible velocities of a particle, from negative to positive infinity, the resulting sum will be one:

$$\int_{-\infty}^{\infty} f dV_C = 1 \tag{II.2}$$

This integral defines the distribution function. In this case, the function *f* is the velocity distribution function (vdf). It is also possible to express the probability of finding a molecule at a specific internal energy state. This will be referred to as the internal energy (rotational, vibrational) probability distribution function (PDF). The summation over all internal energy states also applies to the internal energy PDFs.

It is possible, using a distribution function, to define macroscopic quantities, as long as they are functions of the variable represented by the PDFs. In other words, if Q is some macroscopic variable and Q is a function of the molecular velocity, c_i, then the average or expectation value, $<Q>$, is given as (Vincenti and Kruger, 1967: 29-31):

$$< Q > = \int_{-\infty}^{\infty} Q f dV_C \tag{II.3}$$

The Boltzmann Equation

With an understanding of probability density functions it is possible to define one of the fundamental equations of kinetic theory, the Boltzmann Equation. The generalized Boltzmann equation models the probability a single particle will be found in a certain position, velocity, and internal energy state space. It does so by balancing changes brought about by time, convection into a position, changes in velocity by body forces, and collisions with other particles.

As mentioned, the Boltzmann equation models a single particle. The more generalized Liouville equation is actually an equation for the N-particle distribution function, a much more broad and exact representation of statistical mechanics. The Boltzmann equation is limited compared to the Liouville, namely that it is only appropriate for electrically neutral, low-density situations where only binary collisions are considered. For this reason, the Boltzmann equation is especially useful for low-density flows, such as dilute gases (Bird, 1994: 7).

The Boltzmann equation simplifies further by limiting it to elastic collisions, reducing the complexity of the following theory. If internal energy modes were included, extra variables would be introduced to the distribution functions. Studying this simplified form of the Boltzmann equation with the Chapman-Enskog expansion gives a kinetic based explanation to the limits of the continuity equations. Internal energy effects significantly complicate this analysis, but the general concepts derived from the monatomic gas treatment still apply.

It is appropriate to discuss the Boltzmann equation for position and velocity in some detail. It can be derived from the Liouville equation as (Vincenti and Kruger, 1967: 330):

$$\frac{\partial}{\partial t}\left[nf(c_i)\right] + c_j \frac{\partial}{\partial x_j}\left[nf(c_i)\right] + \frac{\partial}{\partial c_j}\left[F_j nf(c_i)\right] = \left\{\frac{\partial}{\partial t}\left[nf(c_i)\right]\right\}_{coll} \qquad \text{(II.4)}$$

This equation models the distribution of molecules in a phase space defined by position and velocity. Molecules inside the phase space are denoted by c_i. The first term on the left hand side represents the change in time of the distribution. The second term on the

left models the convection of particles into the phase space due to their own velocity. The third and last term on the left side represents the convection of particles into the velocity space due to an external force *F*.

The integral on the right hand side of the equation sums the effects of intermolecular collisions. It gives the rate of increase in molecules in the position and velocity space. An additional assumption is made that only collisions involving two molecules are important. This rules out tertiary collisions and limits the use of the Boltzmann equation to a dilute gas. While a dense gas version of the collision integral can be derived, the present work requires only the simpler version (Chapman and Cowling, 1952: 275). It can be represented as (Vincenti and Kruger, 1967: 332):

$$\left\{\frac{\partial}{\partial t}\left[nf(c_i)\right]\right\}_{coll} = \int_{-\infty}^{\infty}\int_{dP_c} n^2\left[f(c_i')f(\zeta_i') - f(c_i)f(\zeta_i)\right]g\,dP_c\,dV_\zeta \qquad \text{(II.5)}$$

Here, the term ζ_i is the velocity of the collision partner and the terms $f(c_i')$ and $f(\zeta_i')$ are the post-collision particle distributions. The relative speed, *g*, is defined as $|\zeta_i - c_i|$. The integral is summed over the differential cross-section dP_c, which when integrated from zero to 4π is the area of a unit sphere. It is also integrated over dV_ζ, the phase volume of the collision partner being collided with. This integral is difficult to use and simplifications on particle structure and interaction are often made to simplify it. However, because the intermolecular collisions largely determine the behavior of the flow, care should be made when applying any sort of assumptions.

The Maxwell-Boltzmann Equilibrium Distribution

If the flow is in equilibrium, the collision integral in Equation (II.5) should equal zero because no changes are occurring to the distribution function over time. This is only possible if the integrand is equivalently zero, or:

$$f(c_i')f(\zeta_i') = f(c_i)f(\zeta_i) \tag{II.6}$$

This condition maintains equilibrium because each collision is exactly balanced. A general solution for f satisfying Equation (II.6) can be found in terms of thermal velocities using equilibrium kinetic theory (Vincenti and Kruger, 1967: 43-44). Thermal, or peculiar, velocity is a particle's velocity with respect to the bulk motion of the flow, or $C = (c - u)$. The equilibrium distribution, also known as the Maxwell-Boltzmann distribution, is given as:

$$f_o = f(C_i) = \left(\frac{m}{2\pi kT}\right)^{\frac{3}{2}} \exp\left(-\frac{m}{2kT}C^2\right) \tag{II.7}$$

Equation (II.7) contains k, the Boltzmann constant, m, the mass of each individual molecule, and T, the temperature defined in terms of the expectation value of C^2:

$$T = \frac{m}{3k} < C^2 > \tag{II.8}$$

Where C^2 is defined as:

$$C^2 \equiv C_1^2 + C_2^2 + C_3^2 \tag{II.9}$$

The Chapman-Enskog Solution to the Boltzmann Equation

Because of the difficulty in attaining a solution to the Boltzmann equation, many attempts have been made to simplify the mathematics involved. Chapman and Enskog developed one such method independently (Chapman and Cowling, 1952: 107-133). A simplified version of this method appears here. The method begins by decomposing the distribution function into components:

$$f = f_o(1 + \Phi_1 + \Phi_2 ...) \tag{II.10}$$

Here f_o represents the equilibrium Maxwellian distribution, and higher terms are perturbations from equilibrium. The r^{th} approximation to the above distribution function can be written for any order:

$$f_r = f_o(1 + \Phi_1 + \Phi_2 + ... + \Phi_r) \tag{II.11}$$

The zeroth order distribution function is f_o, the Maxwellian. Substituting the r^{th} distribution into the Boltzmann equation and equating like terms produces formulations for the r^{th} order perturbation term, Φ_r, for $r > 0$. The first order perturbation term, Φ_1, can be found by substituting $f_1 = f_o(1+\Phi_1)$ into Equation (II.4):

$$\Phi_1 = -\frac{1}{n}\left[\sqrt{\frac{2kT}{m}} A_j \frac{\partial}{\partial x_j}\left(\ln T\right) + B_{jk}\frac{\partial \overline{c}_j}{\partial x_k}\right] \tag{II.12}$$

$$A_i = A\frac{C_i}{\sqrt{2kT/m}} \tag{II.13}$$

$$B_{ij} = B\left(C_i C_j - \frac{1}{3}C^2\delta_{ij}\right) \tag{II.14}$$

The derivation of this perturbation term requires that mass, momentum, and energy are conserved for the f_o distribution. It is not a simple derivation, and the reader should consult the references (Vincenti and Kruger, 1967: 386-390; Chapman and Cowling, 1952: 118-121) for details. The above A and B are integral functions of thermal velocity and temperature obtained by an expansion in series of Sonine Polynomials (Burnett, 1935: 385). Now, with formulations developed for f_o and f_1, it is possible to explore the effects a small perturbation from equilibrium has on a flow. Specifically, the shear stress tensor and the heat flux vector will be examined. As was indicated in the introduction of this work these two are good indicators of non-equilibrium.

It is possible to write both the shear stress tensor and the heat flux vector for a monatomic gas in terms of quantities available through kinetic theory. The shear stress tensor is (Vincenti and Kruger, 1967: 325):

$$\tau_{ij} = -[\rho < C_i C_j > - p\delta_{ij}] \tag{II.15}$$

And the heat flux vector based on translation:

$$q_i = \frac{1}{2}\rho < C_i C^2 > \tag{II.16}$$

By substituting the equilibrium distribution, f_o, from equation (II.7) into equation (II.15) and (II.16), and including the definition of the expectation values as given in equation (II.3), the resulting shear stress and heat flux are:

$$\tau_{ij(0)} = -[p\delta_{ij} - p\delta_{ij}] = 0 \tag{II.17}$$

$$q_{i(0)} = 0 \tag{II.18}$$

This gives conditions equivalent to an inviscid, adiabatic flow field, where viscosity and heat flux are completely absent. Using (II.17) and (II.18) in conjunction with conservation of mass, momentum, and energy it is possible to obtain the Euler Equations of fluid dynamics. This derivation gives information about the nature of the Euler Equations, namely, they describe purely equilibrium flow. Knowing this gives a more fundamental understanding of the limits of applicability of the Euler Equations. They are only valid in portions of a flow where changes in the fluid state are occurring at a very slow rate.

It is also interesting to apply the same process to small perturbations from equilibrium. It is first necessary to realize that mass, momentum, and energy conservation are achieved through the equilibrium solution alone, and the perturbation term, Φ_1, contributes nothing to conservation. In other words:

$$\int_{-\infty}^{\infty} f_o \Phi_1 dV_C = 0 \tag{II.19}$$

$$\int_{-\infty}^{\infty} c_i f_o \Phi_1 dV_C = 0 \tag{II.20}$$

$$\int_{-\infty}^{\infty} C^2 f_o \Phi_1 dV_C = 0 \tag{II.21}$$

The first order approximation is obtained by substituting f_1 into (II.15) and (II.16):

$$\tau_{ij(1)} = -[\rho \int_{-\infty}^{\infty} f_o \left(1+\Phi_1\right) C_i C_j dV_C - p\delta_{ij}] \tag{II.22}$$

$$q_{i(1)} = \frac{1}{2}\rho \int_{-\infty}^{\infty} f_o (1+\Phi_1) C_i C^2 dV_C \tag{II.23}$$

Which, with a significant amount of manipulation and using equations (II.12), (II.19), (II.20), and (II.21), gives (Vincenti and Kruger, 1967: 391):

$$\tau_{ij(1)} = \frac{2kT}{15}\left\{\int_{-\infty}^{\infty} B\frac{C^4}{\left(2kT/m\right)^2} f_o dV_C\right\}\left(\frac{\partial < c_i >}{\partial x_j} + \frac{\partial < c_j >}{\partial x_i} - \frac{2}{3}\frac{\partial < c_k >}{\partial x_k}\delta_{ij}\right) \quad \text{(II.24)}$$

$$q_{i(1)} = -\frac{2}{3}\frac{k^2 T}{m}\left\{\int_{-\infty}^{\infty} A\frac{C^4}{\left(2kT/m\right)^2} f_o dV_C\right\}\frac{\partial T}{\partial x_i} \quad \text{(II.25)}$$

Now, if portions of the above equations are defined as:

$$\mu(C,T) = \frac{2kT}{15}\left\{\int_{-\infty}^{\infty} B\frac{C^4}{\left(2kT/m\right)^2} f_o dV_C\right\} \quad \text{(II.26)}$$

$$K(C,T) = \frac{2}{3}\frac{k^2 T}{m}\left\{\int_{-\infty}^{\infty} A\frac{C^4}{\left(2kT/m\right)^2} f_o dV_C\right\} \quad \text{(II.27)}$$

Then equations (II.24) and (II.25) become:

$$\tau_{ij(1)} = \mu(C,T)\left(\frac{\partial < c_i >}{\partial x_j} + \frac{\partial < c_j >}{\partial x_i} - \frac{2}{3}\frac{\partial < c_k >}{\partial x_k}\delta_{ij}\right) \quad \text{(II.28)}$$

$$q_{i(1)} = -K(C,T)\frac{\partial T}{\partial x_i} \quad \text{(II.29)}$$

These results have the same form as the shear stress tensor and heat flux vector contained in the Navier-Stokes equations (White, 2006: 66, 70). The viscosity, μ, and the thermal conductivity, K, are here given as functions of both temperature and thermal velocity. As has been noted, Sonine Polynomial expansions give a series solution for A and B. This

process is rather complicated and the interested reader is referred to the literature (Vincenti and Kruger, 1967: 397-402; Chapman and Cowling, 1952: 123-129; Burnett, 1935). The infinite series expansions can be truncated to give approximate formulations for the viscosity and the thermal conductivity. The first approximation, which retains only the first expansion term, is given as:

$$\mu^{(1)} = \frac{\frac{5}{8}\sqrt{\pi mkT}}{\left(\frac{m}{4kT}\right)^4 \int_0^\infty g^7 \sigma_\mu(g) e^{\frac{-mg^2}{4kT}} dg} \tag{II.30}$$

$$K^{(1)} = \frac{15}{4}\frac{k}{m}\mu^{(1)} \tag{II.31}$$

It should be pointed out that the superscript on these two approximations is not the same as the order of the perturbation term included in the Chapman-Enskog solution. It is the approximation to the Sonine polynomials. In the above, *g* is the relative speed between two colliding molecules and σ is the collision cross-section of a molecule. The evaluation of these two parameters depends on the molecular model used. One common assumption is that molecules behave as hard spheres when colliding. To add more realistic physics an attractive force between molecules is added. The resulting viscosity relation, known as Sutherland's formula, is given below (Chapman and Cowling, 1952: 223-224):

$$\mu = \mu'\left(\frac{T}{T'}\right)^{\frac{3}{2}} \frac{T'+S}{T+S} \tag{II.32}$$

TThe *S* in this equation, Sutherland's constant, can be given for a gas within a specified temperature range and is a measure of the attractive force of the molecules. The μ'

represents the viscosity at some characteristic temperature *T'*. This, coupled with equation (II.31) is a very common method of calculating viscosity and thermal conductivity, and is used in the CFD computations to follow.

Due to the form of the Boltzmann equation used (equation (II.4)) the derivation of this result is strictly valid only for perfect monatomic gases, that is, only the translational energy mode is considered. It is possible to modify the preceding results to account for energy exchange between the translational and internal modes. However, the details are tedious and one can gather several interesting facts from the simpler monatomic version.

First, no bulk viscosity (μ_B) has been predicted by the monatomic Chapman-Enskog solution, hinting that the bulk viscosity contained in the Navier-Stokes equations is somehow related to the internal energy modes. Experiments support this, suggesting the bulk viscosity is significant when dealing with the structure of shock waves where the increase of translational energy exchange between molecules causes the internal energy modes to activate (White, 2006: 67; Vincenti and Kruger, 1967: 407-412). The link between bulk viscosity and non-equilibrium is discussed in more detail in chapter IV.

Second, the formulations for viscosity and thermal conductivity, which have been given purely as functions of temperature and thermal velocity, are only valid for small perturbations from equilibrium. Thus the constitutive relations found in the Navier-Stokes equations have inherent assumptions limiting them to small perturbations from equilibrium. Flows exhibiting higher degrees of non-equilibrium are not adequately described by the constitutive relations given in equations (II.28) and (II.29). For these situations the continuum formulation is no longer adequate and a method that captures

more of the non-equilibrium effects is needed. The question remains: How does one determine when the continuum formulations are invalid?

Boltzmann's *H*-theorem

The many parameters previously studied as possible indicators of non-equilibrium have already been mentioned in the introduction. The current work will use entropy generation to measure the extent of non-equilibrium. A more detailed justification of entropy generation as an indicator of non-equilibrium is appropriate here.

It is first necessary to find a definition of entropy that is free of any equilibrium assumptions. This will be done by finding entropy as a function of the general velocity distribution function. Beginning with the famous Boltzmann's relation:

$$S = k \ln \Omega \tag{II.33}$$

This relation is a fundamental description of entropy as a measure of disorder. It relates entropy, a macroscopic thermodynamic variable, to the number of possible microstates in the system, Ω. Following the derivation given by Vincenti and Kruger (1967: Chap. 4), the number of microstates can be enumerated:

$$\ln \Omega = \sum_j N_j \left(\ln \frac{\mathcal{C}_j}{N_j} + 1 \right) \tag{II.34}$$

The energy of the system resides in j energy levels, each containing a certain number of quantum energy levels. Here N_j is the number of particles contained within the j^{th} energy grouping, and $\mathcal{C}_j$ is the number of quantum energy states in the j^{th} group. For translational

energy, the number of particles in each group can be defined in terms of the distribution function:

$$\frac{N_j}{N} = f(c)dV_C \tag{II.35}$$

Additionally, $\cancel{C}_j$ is derived to be:

$$\cancel{C}_j = \left(\frac{m^3 \cancel{V}}{h^3}\right) dV_C \tag{II.36}$$

With m as the particle mass, $\cancel{V}$ as the volume and h is Planck's constant. These two relations change the distribution from energy levels to using particle velocities. It can reasonably be assumed that the velocity levels are spaced very close together, thus turning the summation over all energy levels in equation (II.34) into an integration over all possible velocities. Combining equations (II.33)-(II.36) one obtains:

$$S = kN \int_{-\infty}^{\infty} f(c)\left(1 + \ln\left(\frac{m^3 \cancel{V}}{h^3 N f(c)}\right)\right) dV_C \tag{II.37}$$

The sum in the integrand can be separated into two integrals, and recalling the definition of the distribution function from equation (II.2), the sum of the first integral is equal to one. Rearranging and defining the particle density $n = N/V$:

$$S = kN - kV \int_{-\infty}^{\infty} \ln\left(\frac{h^3}{m^3} nf\right) nf dV_C \tag{II.38}$$

This expression for translational entropy is a function of the distribution of particle velocities and is free from assumptions of equilibrium. The change of entropy with time is found by taking the derivative of (II.38) with respect to time:

$$\frac{\partial S}{\partial t} = -kV \int_{-\infty}^{\infty} \frac{\partial}{\partial t} \ln\left(\frac{h^3}{m^3} nf\right) nf dV_C = -kV \int_{-\infty}^{\infty} \left[1 + \ln\left(\frac{h^3}{m^3} nf\right)\right] \frac{\partial (nf)}{\partial t} dV_C \qquad \text{(II.39)}$$

If a gas is assumed to have spherical molecules, with no external forces and uniform throughout, the Boltzmann equation from (II.4) and (II.5) reduces to (Chapman and Cowling, 1952: 69-70):

$$\frac{\partial (nf)}{\partial t} = \int_{-\infty}^{\infty} \int_{dP_c} n^2 \left[f(c_i') f(\zeta_i') - f(c_i) f(\zeta_i) \right] g dP_c dV_\zeta \qquad \text{(II.40)}$$

Substitute this simplified Boltzmann equation into (II.39) to obtain:

$$\frac{\partial S}{\partial t} = -kV \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{dP_c} \left[1 + \ln\left(\frac{h^3}{m^3} nf\right)\right] n^2 \left[f(c_i') f(\zeta_i') - f(c_i) f(\zeta_i) \right] g dP_c dV_\zeta dV_C \qquad \text{(II.41)}$$

In the reference by Chapman & Cowling an integral transformation is introduced:

$$\begin{aligned} &\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{dP_c} \phi_C \left[f(c_i') f(\zeta_i') - f(c_i) f(\zeta_i) \right] g dP_c dV_\zeta dV_C \\ &= \frac{1}{4} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{dP_c} \left(\phi_C + \phi_\zeta - \phi_C' - \phi_\zeta'\right) \left[f(c_i') f(\zeta_i') - f(c_i) f(\zeta_i) \right] g dP_c dV_\zeta dV_C \end{aligned} \qquad \text{(II.42)}$$

Where ϕ is some function of the velocity distribution. Recognizing:

$$\phi = 1 + \ln\left(\frac{h^3}{m^3} nf\right) \qquad \text{(II.43)}$$

And applying the integral transformation to equation (II.41) it follows that:

$$\frac{\partial S}{\partial t} = -\frac{kVn^2}{4} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{dP_c} \ln\left(\frac{f(c_i) f(\zeta_i)}{f(c_i') f(\zeta_i')}\right) \left[f(c_i') f(\zeta_i') - f(c_i) f(\zeta_i) \right] g dP_c dV_\zeta dV_C \qquad \text{(II.44)}$$

Inspection of the integrand reveals its sign does not depend on the respective signs of the two products, $f(c)f(\zeta)$ and $f(c')f(\zeta')$. If $f(c)f(\zeta) > f(c')f(\zeta')$, the portion in the logarithm is positive, and the subtraction is negative, giving the integrand a negative sign. However, if $f(c)f(\zeta) < f(c')f(\zeta')$, the portion in the logarithm is negative, and the subtraction is positive, again giving a negative sign to the integrand. Thus, the integral is always positive, meaning the change in entropy in time is strictly non-negative. The only way for this integral not to equal zero is if $f(c)f(\zeta) = f(c')f(\zeta')$, which, as previously indicated, is the equilibrium condition. This means entropy will always increase unless it is at equilibrium. This lends support to the second law statement a system moves from one equilibrium state to another through the production of entropy. This is Boltzmann's famous H-theorem, named for the function:

$$H = \int f \ln f dV_C \tag{II.45}$$

Boltzmann showed the time rate of change of this function was monotonically decreasing. He also recognized the link between H and the monatomic gas expression for entropy (equation (II.38)), namely their time rates of change are related by a negative constant. In this way, Boltzmann provided support for the second law of thermodynamics based on kinetic theory. However, it should be pointed out many of Boltzmann's contemporaries did not treat his H-theorem as a final word on the matter. They suggested, given the right conditions, the function would actually increase. This was done by imagining a system of molecules of known position and velocity behaving so $dH/dt < 0$. Next, reverse the velocity of every molecule so their paths are exactly retraced, making $dH/dt > 0$. This mental exercise seems to contradict the second law of thermodynamics. However, this paradox, named after Loschmidt, predates Heisenberg's uncertainty

principle. This principle states one cannot actually know the position and velocity of every particle, and thus the original assumption of reversing every particle is not possible. Additionally, as is stated by one source, the probability a system would experience an exact reversal is extremely small. Thus, "The *H*-theorem is to be regarded as being statistical in nature and the best that we can say is that, first, the most probable state of a system in equilibrium is one for which *H* is a minimum, and second, for a system with a value of *H* greater than the minimum there is an overwhelming probability that *H* will decrease …" (Eyring and others, 1964: 117). In regards to the original purpose of this effort, the *H*-theorem provides statistical assurance that entropy will be generated as a system experiences non-equilibrium. In fact, we will use entropy generation as a tool to quantify deviations from equilibrium.

Entropy Generation

The Navier-Stokes equations are limited to small perturbations from non-equilibrium, and entropy generation results from these perturbations. In order to measure the magnitude of these deviations it is necessary to develop a formulation for entropy generation without inherent assumptions of equilibrium built in.

The work on entropy generation by Schrock (2005) relied on a formulation for entropy derived using statistical mechanics and kinetic theory. The translational portion of this was given in equation (II.37). The left hand side of the moment of Boltzmann's equation (II.4) was used to track changes in entropy, thus entropy production. The final expressions for entropy and entropy generation were given in terms of the distribution

functions of velocity, rotational energy, and vibrational energy. DSMC was used to generate these distribution functions. This method, although theoretically accurate, was computationally demanding and statistically difficult, due to the necessity of sorting particles to create the distribution functions. For this reason, a different approach to finding entropy generation has been taken here, one eliminating the need to sort particles into distribution functions. The subsequent derivation follows Comeaux (1995: 49-52) and Camberos (2001).

The Gibbs equation from classical thermodynamics defines entropy in terms of thermodynamic variables. It is found by combining the definition of entropy for a reversible process with the first law of thermodynamics:

$$ds \equiv \frac{\delta q}{T} \tag{II.46}$$

$$\delta q = de + \delta w = de + pd\left(\frac{1}{\rho}\right) \tag{II.47}$$

thus giving the relation:

$$Tds = de + pd\left(\frac{1}{\rho}\right) \tag{II.48}$$

The above can be rewritten for a moving element in terms of the total derivatives:

$$\rho T \frac{Ds}{Dt} = \rho \frac{De}{Dt} - \frac{p}{\rho}\frac{D\rho}{Dt} \tag{II.49}$$

Conservation of mass, momentum (neglecting external forces), and energy (neglecting external heating) can be written (Tannenhill and others, 1997: 251-257):

$$\frac{D\rho}{Dt} + \rho \frac{\partial u_i}{\partial x_i} = 0 \tag{II.50}$$

$$\rho \frac{D\vec{V}}{Dt} = \frac{\partial}{\partial x_j}\left(-p\delta_{ij} + \tau_{ij}\right) \tag{II.51}$$

$$\rho \frac{De}{Dt} + p \frac{\partial u_j}{\partial x_j} = -\frac{\partial q_j}{\partial x_j} + \tau_{ij} \frac{\partial u_i}{\partial x_j} \tag{II.52}$$

Substitution of the mass and energy equations into equation (II.49) yields:

$$\rho T \frac{Ds}{Dt} + \frac{\partial}{\partial x_i}\left(\frac{q_i}{T}\right) = -\frac{\tau_{ij}}{T}\frac{u_i}{\partial x_j} + \frac{q_i}{T^2}\frac{\partial T}{\partial x_i} \geq 0 \tag{II.53}$$

The terms on the left hand side of this equation represent the transport of entropy in a flow. The right hand side of the relation is positive definite, and is identified as the irreversible entropy generation. Both sides of equation (II.53) should be greater than or equal to zero to satisfy the second law of thermodynamics.

Because they are non-negative and associated with irreversibilities, the terms on the right hand side of the equality are the entropy generation density:

$$\dot{S}_{gen} = \frac{\tau_{ij}}{T}\frac{\partial u_i}{\partial x_j} - \frac{q_i}{T^2}\frac{\partial T}{\partial x_i} \tag{II.54}$$

It was mentioned at the beginning of this derivation that a formulation for entropy generation free from assumptions of equilibrium was needed. The mass, momentum, and

energy conservation equations (II.50) - (II.52) do not, of themselves, assume equilibrium. The constitutive relations for the shear and heating found in the conservation equations are where equilibrium assumptions are normally made. However, since these two have not yet been defined, they do not make any assumption of equilibrium.

The other component of concern in this derivation is the Gibbs equation (II.48). The derivation assumes this equation is valid for non-equilibrium situations, but there is substantial debate as to the truthfulness of this assumption (Comeaux, 1995: 74). This debate is the foundation of Extended Irreversible Thermodynamics (EIT), in which the Gibbs equation is modified for non-equilibrium by introducing the shear and heating as new thermodynamic variables. This discipline is relatively new, and beyond the scope of this document. One consolation given by Comeaux is the Gibbs equation is valid at least to second order in the Knudsen number. For this research it will be assumed the Gibbs equation holds for high enough orders of accuracy to produce meaningful results, while recognizing it has some equilibrium limitations. Future analysis will seek to implement a formulation able to remove these limitations.

The next task is to find expressions for the shear stress tensor, τ_{ij}, and the heat flux vector, q_i, free from equilibrium assumptions. Traditionally, as found in the Navier-Stokes equations for a continuum, they can be expressed as follows:

$$\tau_{ij} = \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_i}{\partial x_i}\right) + \lambda \frac{\partial u_i}{\partial x_i}\delta_{ij} \tag{II.55}$$

$$q_i = -K\frac{\partial T}{\partial x_i} \tag{II.56}$$

These forms are quite similar to the expressions found using the Chapman-Enskog solution to the Boltzmann equation (the bulk viscosity is related to the second viscosity coefficient by the definition $\lambda \equiv \mu_B - 2/3\ \mu$) valid only for small deviations from equilibrium. Many researchers have proposed methods to extend the continuum equations to be valid for higher levels of non-equilibrium. Although it is not useful to list them all here, a few will be mentioned to demonstrate how they seek to extend the range of validity of the continuum equations, but unfortunately introduce added complications.

One way of extending the validity of continuum relations is to use the Chapman and Enskog method, but include higher order perturbation terms. By including the second order, a complicated set of equations called the Burnett equations results. The Burnett equations can be thought of as the next step after the Euler and Navier-Stokes equations. Some evidence has shown the Burnett equations do indeed extend the limits of non-equilibrium (Fiscko and Chapman, 1989; Pham-Van-Diep and others, 1991). However, some researchers found these equations lead to instabilities, and negative entropy generation (Comeaux, 1995: 37-39, 53-56).

Another method of extending the range of validity of the continuum equations was developed by Grad after realizing the closure relations found by the Chapman-Enskog solution to the Boltzmann Equation are limited by approximating the velocity distribution functions with only two thermodynamic variables (1963). According to Grad, this did not allow the system to capture a wide range of distribution functions, inherently limiting such a system to near-equilibrium. His solution was to introduce new variables to describe the distribution functions. He expanded the distributions to higher moments, and

thus obtained as new thermodynamic variables the components of the shear and the heating. These, instead of being represented by constitutive relations, now helped to define the thermodynamic state of the system. With the added relaxation equations for these new variables, Grad constructed 13 equations. Although this method proved initially successful at capturing additional non-equilibrium effects, it too proved unstable for certain situations (Comeaux, 1995: 119-120).

An different way of obtaining the shear stress tensor and heat flux vector is to completely abandon the continuum formulations and instead resort to a particle method. Although any computational must rely on some equilibrium assumptions, particle methods seek to attain a greater physical realism by simulating molecular interactions. A great deal of research has shown that DSMC is a valid tool for capturing non-equilibrium in a flow (Schwartzentruber and others, 2006; Erwin and others, 1989, Boyd, 1989; Gallis and others) The particle interactions in DSMC are based on kinetic theory, and it is not difficult to generate shear and heating terms. The kinetic definitions for these two constitutive relations are provided in equations (II.15) and (II.16). They are given in terms of the thermal velocity and internal energy of each particle, both known when using the DSMC method.

III. Numerical Methods and Implementation

Changes to Previous Entropy Generation Calculations

In the previous chapter, it was explained that although the method used in the preceding work by Schrock is theoretically accurate, it was computationally undesirable. Particles had to be sorted into the probability distribution functions (PDFs); a time consuming process. The values of entropy and entropy generation were highly sensitive to the number of particles collected into each PDF. It was also necessary to calculate the gradient of the entropy flux, a process which introduces numerical error, particularly for the unstructured grid. Finally, the modifications made to the code to implement these calculations limited it to serial computations.

Schrock's work was purely one dimensional, making the calculations relatively simple, so the above limitations were not an issue (2005). However, two and three-dimensional problems require much larger grid sizes, and thus many more particles. Increasing the number of particles used has a detrimental effect on computational time as will be demonstrated later in this section. Although the author parallelized the code to run on multiple processors, the PDF sorting routines made the code almost unusable because of the great amount of computation time needed.

In addition to these problems, the entropy generation obtained by integrating over the PDFs was statistically poor. Increasing the number of particles collected into each PDF improved the quality of the solution, but it was difficult to achieve sufficient numbers

with large grids. Although the method of finding entropy generation using the distribution functions is interesting because it reveals information about the shape of the distributions themselves and thus links directly to Boltzmann's *H*-Theorem, it was finally discarded for the current method.

The preceding chapter ended with an alternate formulation for entropy generation valid for non-equilibrium flows. This method is superior because it does not rely on a sorting routine to evaluate PDFs. Instead, it matches the original structure of the DSMC code by summing moments and using the sums to calculate macroscopic expectation values. This decreases the computing time and decreases the statistical sensitivity of results.

Problem Setup

It is desirable to compare the continuum formulation with the DSMC formulation to ensure they are compatible. To do this, two experimental cases performed at the Calspan-University of Buffalo Research Center (CUBRC) were modeled using Navier-Stokes based CFD and DSMC. The geometries of the two experiments, a hollow cylinder flare (run 5) and a double cone (run 7) are pictured in Figs. 1 and 2 (Holden and Wadhams, 2007). All measurements are given in inches for both figures.

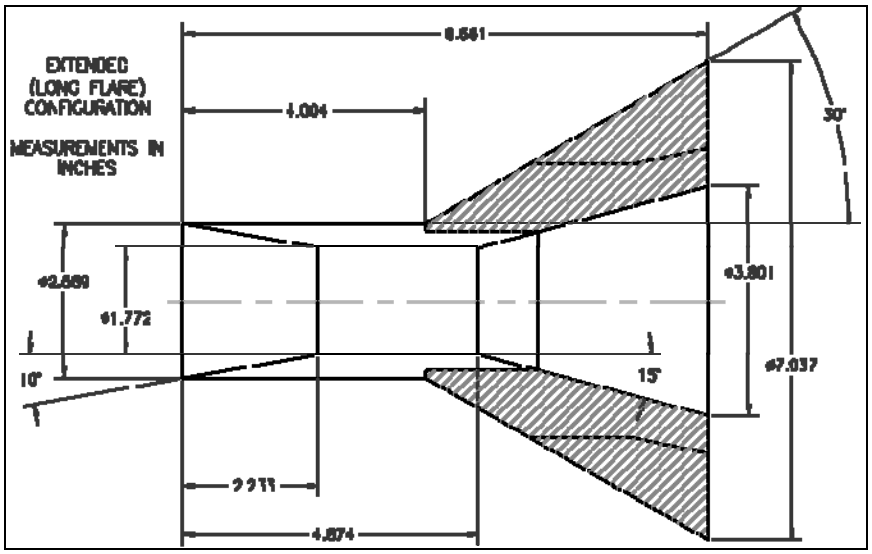


Figure 1: Hollow Cylinder-Flare Geometry (run 5), taken from Holden and Wadhams (2007)

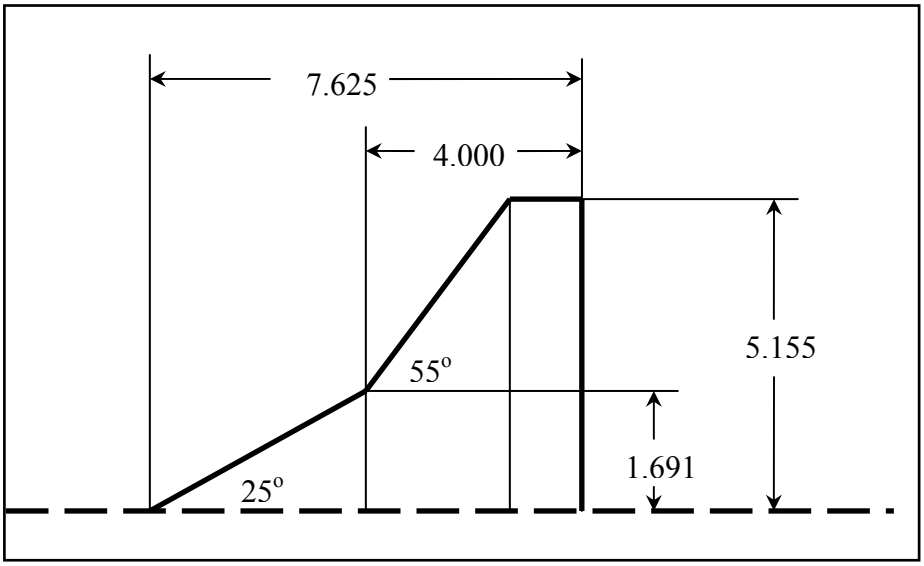


Figure 2: Double Cone Geometry (run 7), Holden and Wadhams (2007)

The flow conditions of runs 5 and 7 are described in Table 1. The low freestream Reynolds Numbers ensure both flows remain laminar.

Table 1: Flow Properties

Run	Mach	Re	T_{inf} (K)	P_{inf} (Pa)	ρ_{inf} (m^3/kg)	V_{inf} (m/s)
5	15.3	3.20E+04	52.28	2.523	0.00016	2252.47
7	15.6	4.19E+04	42.611	2.227	0.000176	2072.64

An investigation Knudsen number gives an idea of what type of flow may be involved with these particular cases. For run 5, the freestream mean free path is 3.096×10^{-4} m. To find the freestream Knudsen number, use the length along the x-axis of the geometry, L, as the characteristic length. By equation (I.1) this gives $Kn_\infty = 0.0014$. The freestream Kn is well within the accepted limits of continuum flow ($Kn << 1.0$).

The local Knudsen number for a shock is quite different however. If the thickness of a shock is used as the characteristic length, the Kn is more in the range of 0.3. This cannot be seen as a continuum. As discussed in the introduction to this document, the choice of the characteristic length is somewhat arbitrary.

Other research groups have previously used these test cases to validate DSMC and CFD codes (Harvey, 2003). Codes are validated comparing the solution at the wall to the experimental data. This relies on two parameters, the coefficient of pressure and the Stanton number, defined as:

$$Cp = \frac{p - p_\infty}{\frac{1}{2}\rho V_\infty^2} \tag{III.1}$$

$$St = \frac{q}{\frac{1}{2}\rho V_\infty^2} \tag{III.2}$$

CFD–Based Analysis

The CFD software used in this analysis was FLUENT, a standard fluid dynamics package (FLUENT, 2004). Although this software has the ability to make some compensation for real gas effects, for this research the fluid was modeled as an ideal perfect gas. This was done to illustrate the limitations of a CFD solution not taking into account real gas effects as compared to the particle based DSMC. The solutions are not expected to agree well with the experimental data.

Unstructured grids were adapted to flow gradients in preliminary solutions. Three adaptive iterations were performed to ensure that the cell sizing was adequate to capture the shock zones and the boundary layers. Figure 3 and Figure 4 show the final grids used.

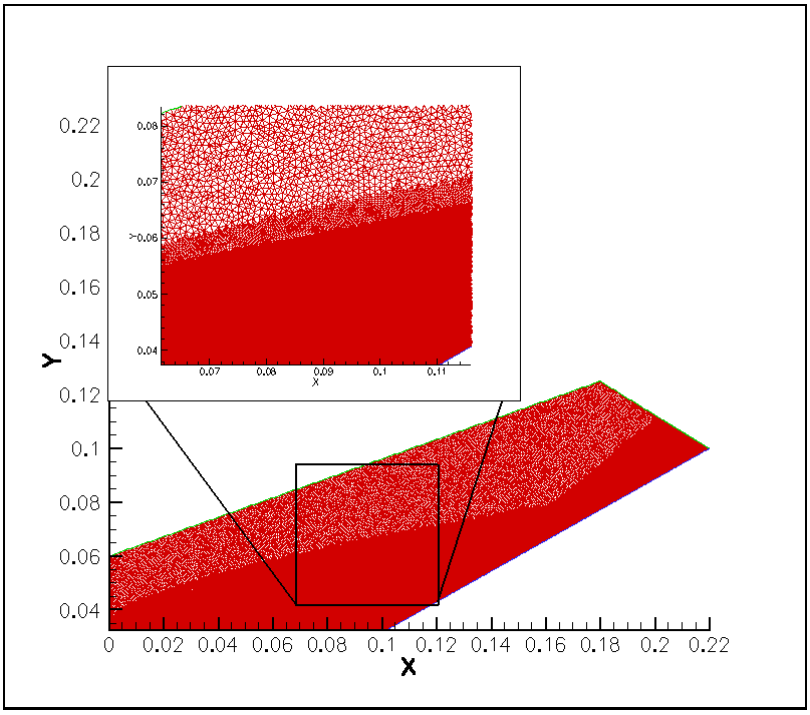


Figure 3. Run 5, FLUENT adapted grid

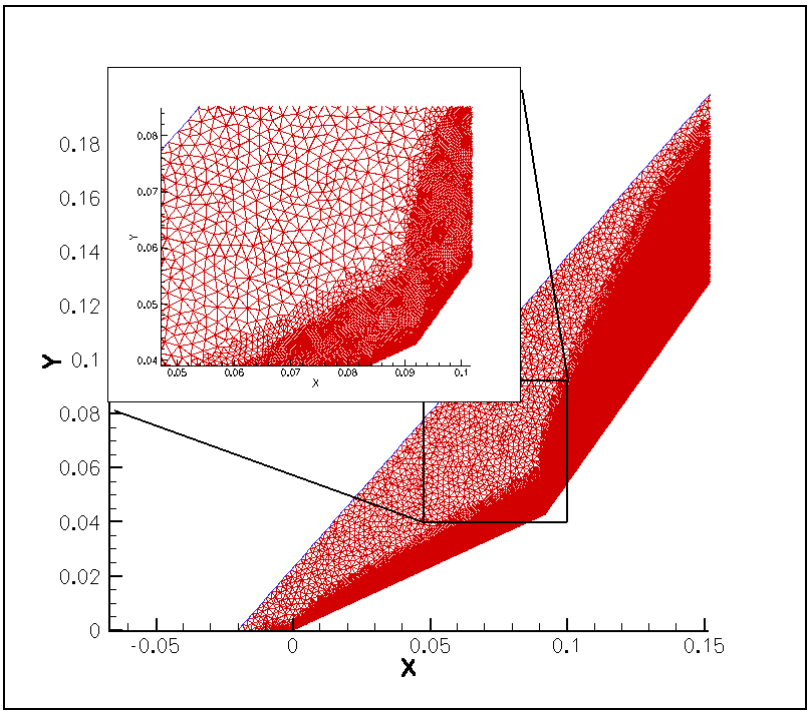


Figure 4. Run 7, FLUENT adapted grid

Perfect gas laminar solutions were obtained using a 2nd order upwind scheme for the fluxes and Sutherland's law for the viscosity. The residuals of the two runs did not converge, perhaps because of the low pressure of the flow. However, the wall data, compared to experimental data, provides a validation for the solution.

To calculate entropy generation for the CFD simulations, equations (II.55) and (II.56) for the shear stress and heat flux were used in equation (II.54) for the entropy generation. Recall this formulation of the constitutive relations, τ_{ij} and q_i, assumes only small deviations from equilibrium.

The viscosity, μ, and the thermal conductivity, K, are calculated using Sutherland's formulas, which are kinetic models based on a rigid sphere assumption and empirical constants (White, 2006: 28-30; Tannehill and others, 1997: 259, Vincenti and Kruger,

1967: 21-23). These formulas, while valid for simple gases at moderate temperatures, are probably not valid for highly non-equilibrium flows. The bulk viscosity term, μ_B, is assumed to be zero, which is the same as Stokes's hypothesis of $3\lambda + 2\mu = 0$[1]. Although this is a common assumption, it is likely invalid for the high-speed flows with shocks (White, 2006: 67). According to the work by Schrock, good CFD results for his normal shock calculations were obtained by assuming $\lambda = 0$, or $\mu_B = (2/3)\mu$.

DSMC–Based Analysis

Compared to Molecular Dynamics (MD), which models all molecules in a flow, DSMC reduces computational time by statistically representing a large number of actual particles with a single simulated particle. The velocity and internal energy state of each simulated particle is tracked as it moves about the flow, colliding with surfaces and other simulated particles. One important assumption made by DSMC is that the movement of the molecule and the effects of a collision can be uncoupled within a small time step. In other words, the position of each simulated molecule is calculated based on the time step and velocity, and afterwards collisions are allowed to take place within that same time step, independent of how the particles have moved. There are many sources available describing the mathematics involved in DSMC; one excellent reference is the book by Bird (1994).

The statistical modeling performed by DSMC also makes use of computational approximations. As noted by Bird (1994: 214), in a real gas it is possible a relative few

[1] $\lambda \equiv \mu_B - (2/3)\,\mu$.

molecules at extreme positions in a distribution may have significant effects on the flow. These few particles will not be represented in the distribution formed by the virtual particles, and the effects will not be captured in the flow. For this reason, it is best to keep the ratio of simulated particles to the number of actual particles as low as possible. Also, in order to capture collisions, the time step and cell size should tend toward zero, and be much less than the mean collision time and mean free path, respectively. Acceptable values of these parameters will vary depending on the flow.

This research makes use of the DSMC code MONACO created, developed, and maintained at the University of Michigan (Boyd and Wang, 2001). The simulations used the variable hard sphere model of Bird (1994: 40-41).

There are many parameters potentially affecting the accuracy of the results of a DSMC run. Some are specific to the gas species, such as those used to compute energy exchanges during collisions. Table 2 shows the species parameters for diatomic nitrogen. Schrock used these values in the previous work, and they appear in various sources in the literature (2005: 52). Schrock indicates the choice of these parameters may impact the solution. It is reasonable to assume that some of the discrepancy between the experimental data and the simulation results is due to one or more of these values. Future studies should investigate the effects these parameters may have on a solution.

Table 2. DSMC Diatomic Nitrogen Parameters

Molecular Weight (g/mol)	28.01
VHS Exponent ω	0.24
VHS Reference Diameter (pm)	407
VHS Reference Temperature (K)	273
Rotational degrees of freedom	2
Vibrational degrees of freedom	1.8
Θ_{rot} (K)	2.88
Θ_{vib} (K)	3371
Max Rotational Collision #	15.7
T_{ref} in Rot. Model (K)	80
Probability of Vibrational Exchange	0.01
Equilibrium Separation (pm)	109.769
Oscillating Frequency (Hz)	7.071×10^{13}

In addition to parameters corresponding to the flow species, the DSMC user must have knowledge of how time step size, the ratio of actual particles to virtual particles, grid cell size compared to the mean free path, number of particles per cell, and the total sampling time affect the solution. In order to gain an insight into the variation of these parameters, three studies were performed. The total runtime, the size of the grid, and the number of particles used were varied and compared.

Total Runtime Studies

One of the basic assumptions behind DSMC is the particle movement may be uncoupled from the collision occurring within a small time step. The validity of this assumption will depend on the ratio of the time step size to the mean time between collisions. For this reason, the time steps used in all simulations was adjusted so this ratio would be no greater than 0.1. Experience showed, at least for these cases, a lower ratio did not improve the solution, but did significantly increase the total computation time.

Unlike the residual convergence in CFD, DSMC has no simple parameter like residual convergence to determine how long to run a simulation. Statistically, the more particles sampled over time, the better the solution. In order to determine the total simulation time required for each run, data was taken for both runs 5 and 7 over a span of many time steps. The coefficient of pressure and Stanton number along the wall were used to calculate the RMS error between data sets. Convergence is indicated as the solution ceases to change with runtime, or as the RMS error approaches zero. For run 5, this is plotted on a log-log scale as shown in Figure 5 and Figure 6. The trend is logarithmic, meaning longer runtimes improve the solution less. A good cutoff is indicated when the RMS for the maximum C_p and St values reaches 10^{-3}, requiring about 30,000 time steps for run 5 and 50,000 time steps for run 7.

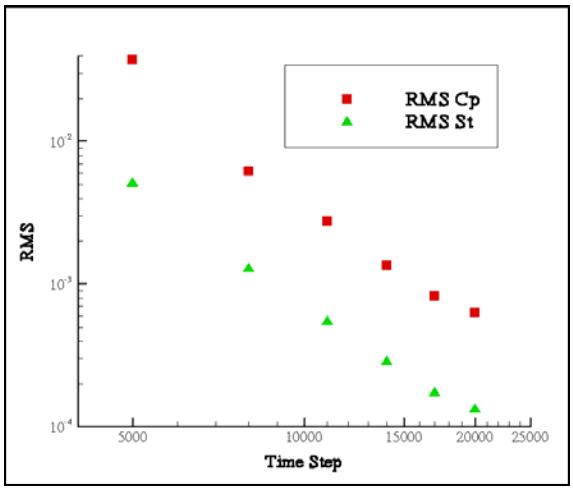


Figure 5. Run 5 RMS of Cp and St

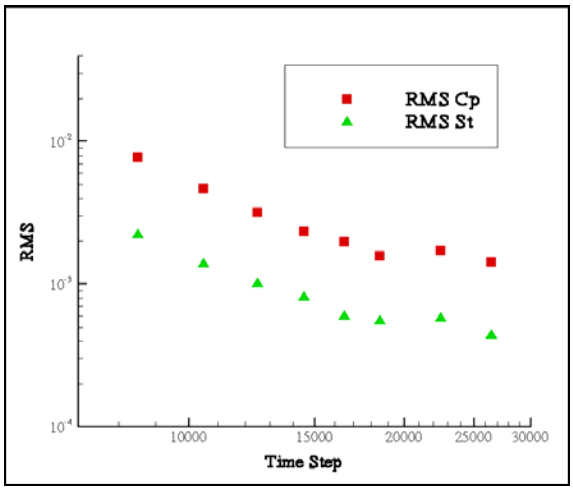


Figure 6. Run 7 RMS of Cp and St

Grid Studies

Three different grids were used for each run to observe the dependence of the solution on the cell sizes. The initial grids were uniformly coarse, the medium grids remained

uniform spacing but with a decreased cell size. The third grid was refined using the solution obtained on the medium grid. Cells are refined based on the mean free path to ensure the ratio of the cell size per mean free path is less than 1.0. Table 3 shows the number of cells for each grid, the processor hours to finish the solution to 200,000 time steps, and the average cell size per mean free path. Investigation of the solutions as compared to the experimental data revealed the average cell-size per mean free path needed to be less than 1.0.

Table 3. Grid Study

Grid	Number of Cells	Processor Hours	Ave cell-size/mfp
Run 5 Coarse	53,248	832	1.82
Run 5 Medium	332,743	969	1.71
Run 5 Refined	215,800	976	0.92
Run 7 Coarse	70,727	576	3.14
Run 7 Medium	442,102	736	1.18
Run 7 Refined	472,168	728	0.998

Particle Studies

The solutions are also dependent on the ratio of actual particles to virtual particles. A lower ratio gives a higher number of virtual particles used in the simulation, and theoretically better results. However, using more particles in the simulation significantly increases the processor hour demands. Four levels of variation were used for run 5 to determine the correct number of particles to use.

Table 4. Particle Ratio Study

Run	Particle Ratio	# of Particles	Processor Hours
Run 5 Least	8×10^{12}	2,100,000	88
Run 5 Less	5×10^{12}	3,356,000	176
Run 5 Middle	10^{12}	16,749,000	512
Run 5 More	8×10^{11}	20,930,000	618

Figure 7 and Figure 8 show the variation of the coefficient of pressure and the Stanton number with the number of virtual particles used in the simulation. The jump from 1×10^{12} and 8×10^{11} in particle ratio gives no visible improvement on the solution. However, there is a marked difference between 1×10^{12}, 5×10^{12}, and 8×10^{12}. This indicates that using 10^{12} actual particles per virtual particle is sufficient. A similar process for run 7 determined a sufficient particle ratio of 5×10^{12}.

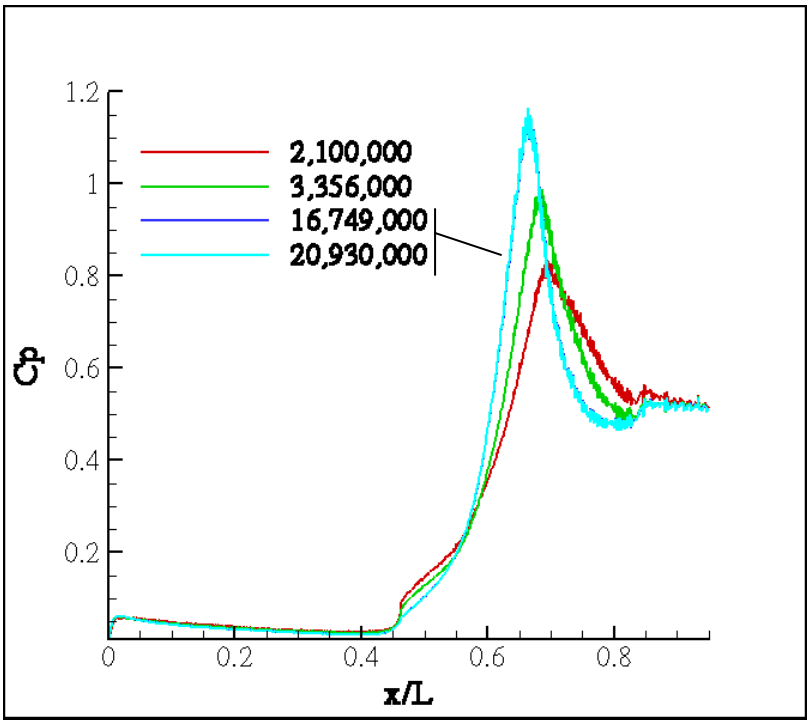


Figure 7. Run 5 - Variation of Cp with number of simulated particles

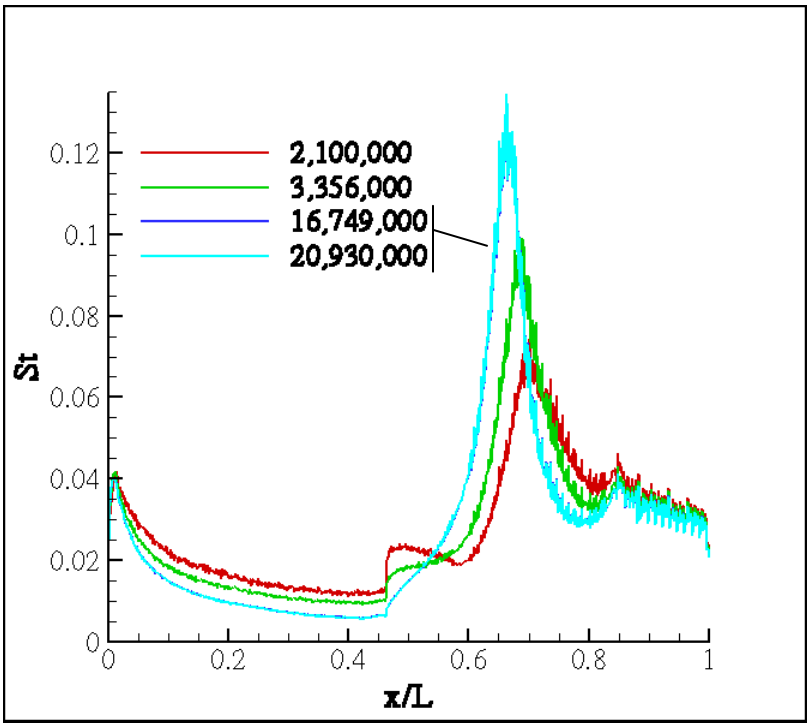


Figure 8. Run 5, Variation of St with number of simulated particles

Calculating Shear and Heating

Because the constitutive relations given in equation (II.55) and (II.56) are based on equilibrium assumptions, it is necessary to use other methods to compute entropy generation. A method such as DSMC, based on kinetic theory, does not make the same assumptions of near equilibrium made in the Navier-Stokes constitutive relations. When using DSMC the velocities (c_i) and internal energy state (ε_{rot}, ε_{vib}) of each virtual-particle is known. The shear tensor and heat flux vector for a diatomic gas with rotational and vibrational energy can be given in terms of the thermal velocities ($C_i = c_i - u_i$), where $u_i = < c_i >$, the average molecular velocity, and internal energies as (Vincenti and Kruger, 1967: 325-326):

$$\tau_{ij} = -[\rho < C_i C_j > - p\delta_{ij}] \tag{III.3}$$

$$q_i = \frac{1}{2}\rho < C_i C^2 > + n < C_i \varepsilon_{rot} > + n < C_i \varepsilon_{vib} > \tag{III.4}$$

The heat flux vector in equation (III.4) differs from the monatomic gas version from equation (II.16) by including the flux contributions from the rotational and vibrational energy modes. Entropy generation is found by substituting equations (III.3) and (III.4) into equation (II.54):

$$\dot{S}_{gen} = -\frac{(\rho < C_i C_j > - p\delta_{ij})}{T}\frac{\partial \mu_i}{\partial x_j} - \frac{(\rho < C_i C^2 > + n < C_i \varepsilon_{rot} > + n < C_i \varepsilon_{vib} >)}{T^2}\frac{\partial T}{\partial x_i} \tag{III.5}$$

Kinetic theory defines temperature, T, pressure, p, and density, ρ, as:

$$T = \frac{1}{3}\frac{m}{k} < C^2 > \tag{III.6}$$

$$p = \frac{1}{3}\rho < C^2 > \tag{III.7}$$

$$\rho = nm \tag{III.8}$$

where k is the Boltzmann constant, m is the mass of each particle, and n is the particle density, or number of particles per unit volume.

The remaining expectation values ($<C_iC_j>$, $<C_iC^2>$, $<C_i\varepsilon_{rot}>$, and $<C_i\varepsilon_{vibj}>$) are found by summing up the products within the brackets over all particles and dividing by the number of particles, N. A few examples of how this is done are helpful. The average velocity in the x-direction, is calculated as follows:

$$u_1 = < c_1 > = \frac{1}{N}\sum_{k=1}^{N} c_{1,k}\ ^2 \tag{III.9}$$

The DSMC program calculates the velocity and internal energy of each virtual-particle during each time step, and then stores summations like the above, allowing the user to extract expectation quantities based on these summations. Another example is the shear stress. The three-dimensional shear stress tensor, τ_{ij}, contains nine components. Because it is symmetrical only six are distinct. Each of the components contains the expectation value $<C_iC_j>$. For the example, let us look at $<C_1C_2>$:

[2] The k here is used to denote that the summation is performed over all particles, 1 to N. This notation will be dropped in the following equations.

$$
\begin{aligned}
< C_1 C_2 > &= \frac{1}{N}\sum (c_1 - u_1)(c_2 - u_2) \\
&= \frac{1}{N}\sum (c_1 c_2 - u_1 c_2 - c_1 u_2 + u_1 u_2) \\
&= \frac{1}{N}\sum c_1 c_2 - u_1 \frac{\sum c_2}{N} - u_2 \frac{\sum c_1}{N} + \frac{u_1 u_2}{N}\sum 1 \\
&= \frac{1}{N}\sum c_1 c_2 - u_1 u_2 - u_2 u_1 + \frac{u_1 u_2}{N} N \\
&= \frac{1}{N}\sum c_1 c_2 - u_1 u_2
\end{aligned}
\qquad \text{(III.10)}
$$

The expectation values within the heat flux vector ($<C_i C^2>$, $<C_i \varepsilon_{rot}>$, and $<C_i \varepsilon_{vib}>$) may be calculated in a similar fashion to equation (III.10). The first value, representing the translational contribution to the heat flux, expands to:

$$< C_i C^2 > = < C_i C_1^2 + C_i C_2^2 + C_i C_3^2 > = < C_i C_1^2 > + < C_i C_2^2 > + < C_i C_3^2 > \qquad \text{(III.11)}$$

Each of the individual expectation values separately are:

$$< C_i C_1^2 > = \frac{1}{N}\sum (c_i - \mu_i)(c_1 - u_1)(c_1 - u_1) = \frac{\sum c_i c_1^2}{N} - 2u_1 \frac{\sum c_i c_1}{N} + 2u_i u_1^2 - u_i \frac{\sum c_1^2}{N} \qquad \text{(III.12)}$$

$$< C_i C_2^2 > = \frac{1}{N}\sum (c_i - u_i)(c_2 - u_2)(c_2 - u_2) = \frac{\sum c_i c_2^2}{N} - 2u_2 \frac{\sum c_i c_2}{N} + 2u_i u_2^2 - u_i \frac{\sum c_2^2}{N} \qquad \text{(III.13)}$$

$$< C_i C_3^2 > = \frac{1}{N}\sum (c_i - u_i)(c_3 - u_3)(c_3 - u_3) = \frac{\sum c_i c_3^2}{N} - 2u_3 \frac{\sum c_i c_3}{N} + 2u_i u_3^2 - u_i \frac{\sum c_3^2}{N} \qquad \text{(III.14)}$$

The above values each represent a contribution to the i^{th} direction of the translational heat flux.

The internal energy contributions to the heating are calculated similarly:

$$< C_i \varepsilon_{rot} > = \frac{\sum c_i \varepsilon_{rot}}{N} - u_i \frac{\sum \varepsilon_{rot}}{N} \tag{III.15}$$

$$< C_i \varepsilon_{vib} > = \frac{\sum c_i \varepsilon_{vib}}{N} - u_i \frac{\sum \varepsilon_{vib}}{N} \tag{III.16}$$

The DSMC code was modified to calculate the above expectation quantities. These calculations match the method used by the original code to find macroscopic quantities such as temperature and pressure from microscopic particle data. As mentioned previously, this method is less computationally demanding and statistically cleaner than sorting particles into PDFs.

One interesting result of the above formulations is it is possible to separate the contributions to the entropy generation into translational, rotational, and vibrational components. The translational entropy generation formulation is given below:

$$\dot{S}_{gen-trans} = -\frac{(\rho < C_i C_j > - p\delta_{ij})}{T} \frac{\partial \mu_i}{\partial x_j} - \frac{(\rho < C_i C^2 >)}{T^2} \frac{\partial T}{\partial x_i} \tag{III.17}$$

The contributions from the internal energy are only found within the heat flux, and are expressed as follows:

$$\dot{S}_{gen-rot} = \frac{n < C_i \varepsilon_{rot} >}{T^2} \frac{\partial T}{\partial x_i} \tag{III.18}$$

$$\dot{S}_{gen-vib} = \frac{n < C_i \varepsilon_{vib} >}{T^2} \frac{\partial T}{\partial x_i} \tag{III.19}$$

MonacoGui

In addition to the above modifications made to the code, a program was written to help with the setup, submission, and monitoring of DSMC jobs. The inputs and feedback from the DSMC program are done entirely with text files. To handle the many input parameters needed to be set for each run, the MonacoGui program enables the user to use a graphical interface to quickly make modifications to the inputs. It also allows the user to submit jobs to the clusters, and monitor their progress based on a variety of output parameters. The details of this program are included in the appendix.

IV. Results

Run 5 Analysis

To validate the solutions, coefficient of pressure and Stanton number are compared to experimental data provided by the CUBRC hypersonic wind tunnel facility (Holden and Wadhams, 2004). The wall data in Figure 9 and Figure 10 reveals some difference in the DSMC and CFD solutions. The DSMC solution matches the data ahead of $x/L = 0.6$ better. Behind this value both solutions diverge from the experimental data. This region corresponds to a shock-boundary layer interaction that is very difficult to model. This region will be described in more detail in regards to run 7. The DSMC solution somewhat under-predicts the pressure rise in the interaction region. The cause of this is unknown.

The lack of perfect agreement between the experimental and computational wall data does not indicate the DSMC method is in error. Other authors have provided somewhat better solutions to these same cases (Harvey, 2003). Unfortunately, these solutions are often the result of a great deal of "knob turning". As was described in the Numerical Methods section of this document, there are many parameters the DSMC user must understand and wisely control. The amount of experience required may also depend on the level of sophistication of the code. According to one code developer, an elegant code allows a user with little basic knowledge of the exact mechanics of DSMC to obtain a good solution. The solution loses the dependence on the choice of the wide range of run parameters as the code becomes increasingly automated. Many DSMC codes are not yet developed to this level (Bird, 2006).

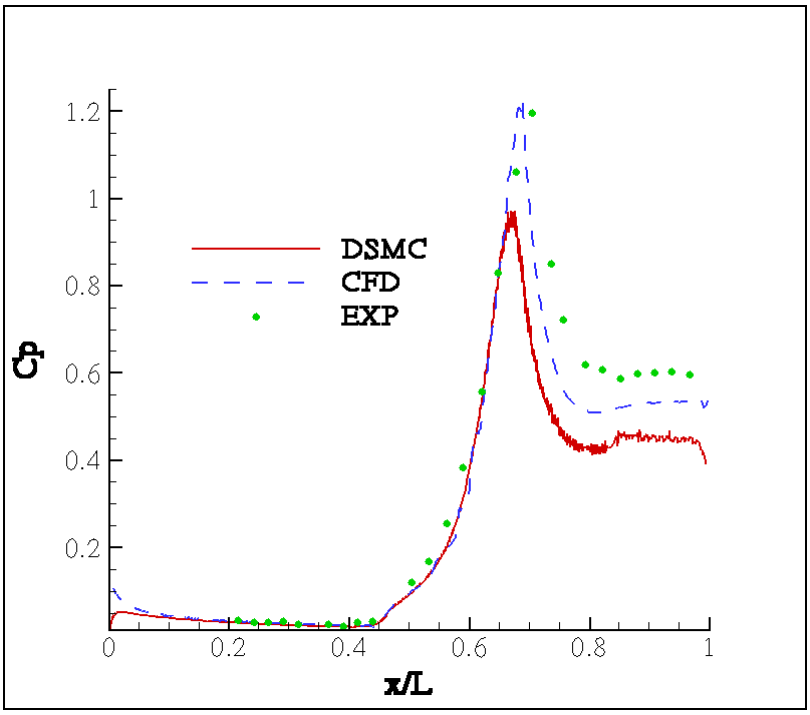


Figure 9. Run 5 Coefficient of pressure comparison

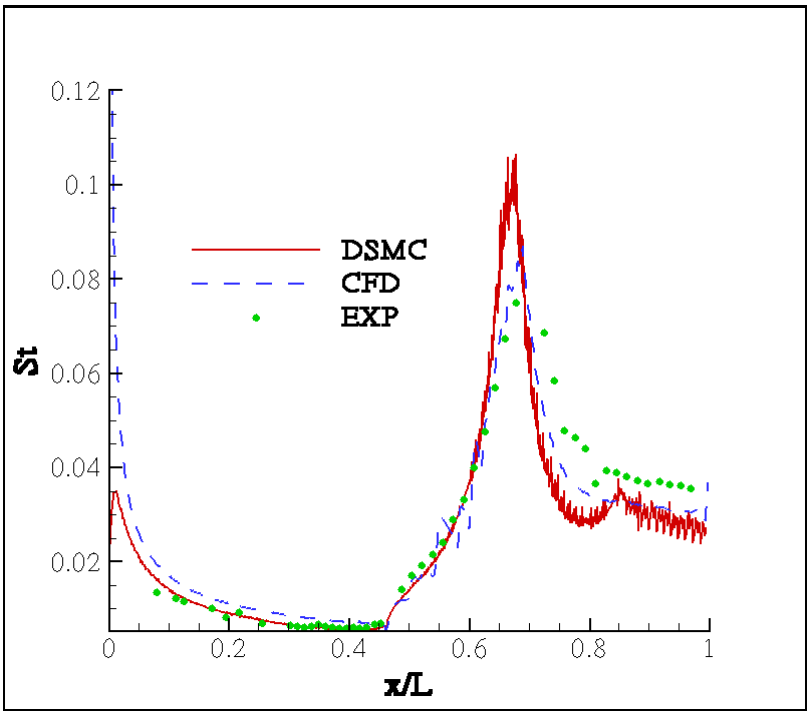


Figure 10. Run 5 Stanton number comparison

Figure 11 and Figure 12 show contour plots of temperature and entropy generation obtained by DSMC for case 5. The entropy generation shown in Figure 12 is non-dimensionalized to represent a ratio between microscopic and macroscopic time scales, similar to the Knudsen number from equation (I.1), a ratio between length scales.

$$Kn_S = \frac{S_{gen}}{\rho_\infty R \nu_\infty} \tag{IV.1}$$

Here, $S_{gen}/\rho R$, the entropy density generation divided by the gas constant and the density, can be interpreted as the macroscopic time scale, while ν, the molecular collision frequency, represents the microscopic time scale. Entropy-generation density is used (as opposed to simply entropy generation) as a direct result of the units in equation (II.54).

The contour plots offer a qualitative analysis of the flow. Standing off the horizontal surface of the cylinder is a shock somewhat like an oblique shock found standing off an inclined surface. The hypersonic boundary layer creates this shock. The high amount of kinetic energy in the freestream transfers to internal energy due to viscosity at the wall. This results in high temperatures, and thus lower densities. The boundary layer grows more rapidly in order to pass the required mass flow. As the displacement thickness in the boundary layer rapidly increases, the effective body seen by the incoming flow correspondingly increases, creating a shock standing off the leading edge of the cylinder.

Immediately following the leading edge, the boundary layer grows quickly due to feedback effects from the freestream. Because the viscous boundary layer and generated shock strongly affect the flow, researchers characterize this region as having strong

viscous interaction. Non-equilibrium effects dominate this region. These effects lessen farther along down the cylinder; the rate of boundary layer growth slows, affecting the freestream much less. The shock also ceases to curve as internal energy relaxation has had time to occur. This region has weak viscous interaction (Anderson, 1989: 15-16, 302-306).

Visual inspection of these two contours indicates the entropy generation occurs in the vicinity of the shocks, as expected. The non-equilibrium in the strong viscous interaction region at the leading edge is also seen.

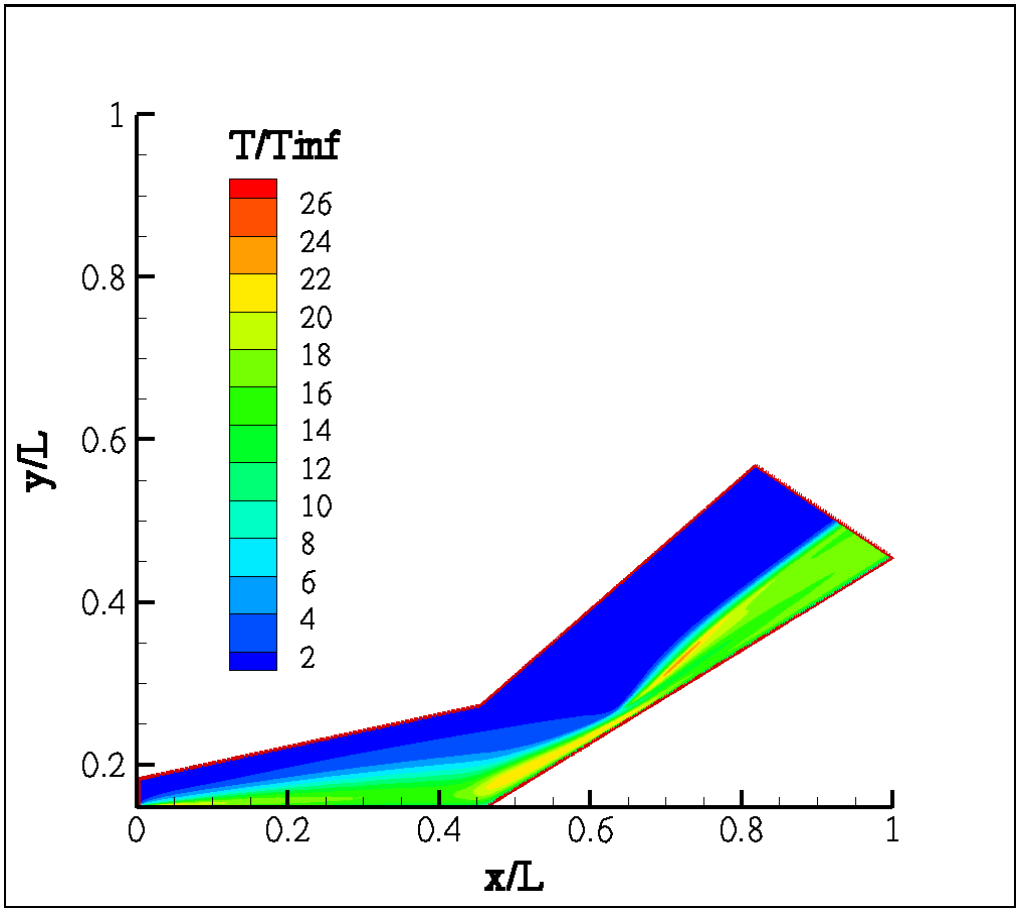


Figure 11. Run 5, Ratio of local temperature to freestream temperature contour

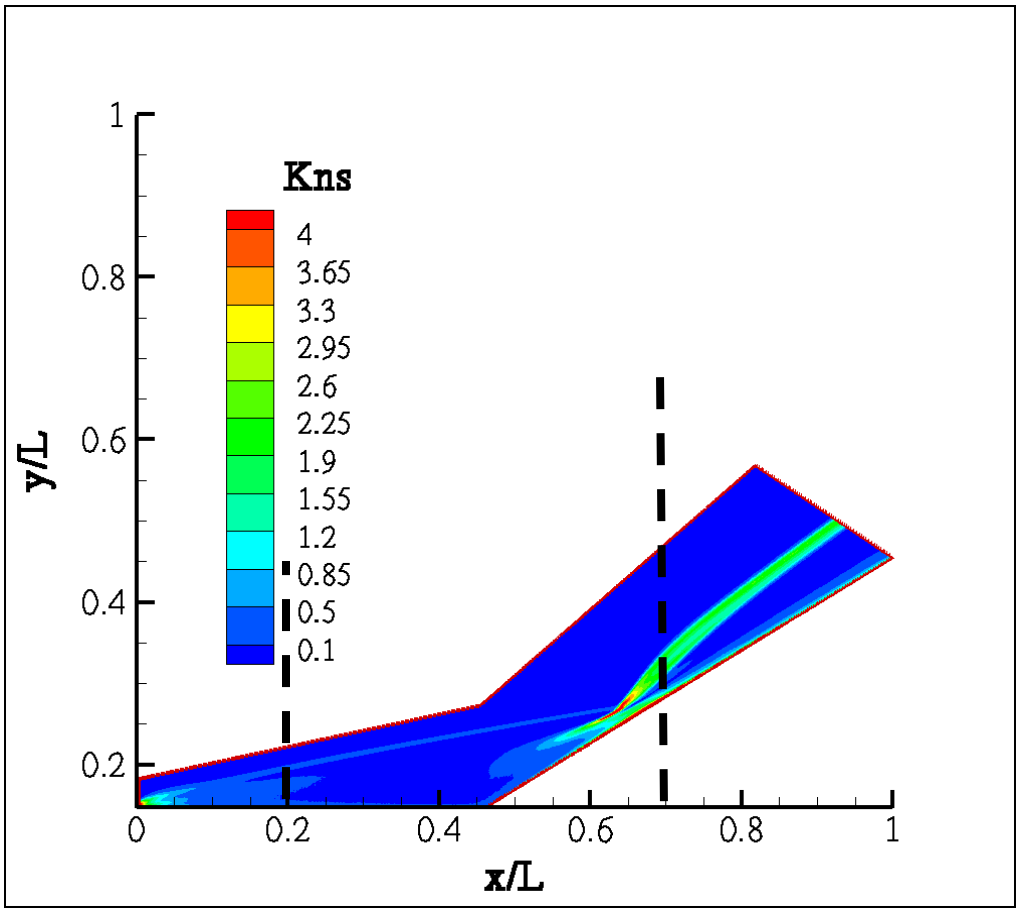


Figure 12. Run 5, Entropy generation contour

The viscous shock eventually impinges onto the surface of the flare resulting in a sharp increase in surface pressure and especially heating at the wall around $x/L = 0.65$ (Figure 9 and Figure 10). This impingement affects the boundary layer and shock forming on the surface of the flare. The oblique shock on the surface of the flare originates from this point of impingement. Run 7 shows an even stronger shock-boundary layer interaction and will be studied in more detail later in this document.

Data from the run 5 solutions was extracted vertically from the wall at $x/L = 0.2$ and $x/L = 0.7$ as shown by the dashed lines in Figure 12. Figure 13 shows the two temperature profiles.

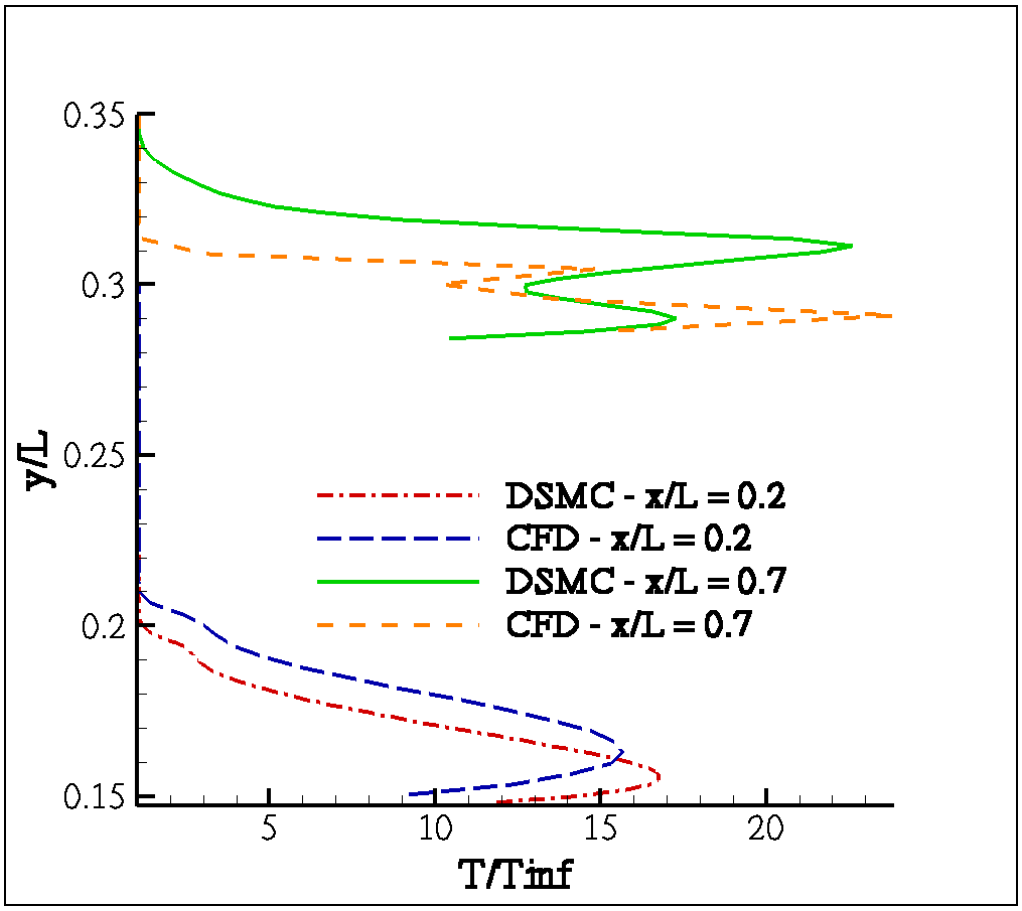


Figure 13. Run 5, Temperature profile comparison at *x/L* = 0.2 and *x/L* = 0.6

As seen in the figure, there are differences between the solution generated using CFD, and the solution generated using DSMC. Approaching the wall vertically down from the freestream at *x/L* = 0.2, the CFD solution predicts a rise in temperature before DSMC. However, the opposite is true at *x/L* = 0.7, where the DSMC solution predicts an earlier rise in temperature. The difference in these two locations is an oblique shock standing off of the flare (*x/L* = 0.7), while there is only a weak shock triggered by the viscous boundary layer standing off of the surface of the cylinder (*x/L* = 0.2).

Further insight can be gained by investigating the entropy generation profile. Figure 14 shows the profile at both *x/L* = 0.2 and *x/L* = 0.7. This figure shows the dramatic

difference between the two locations in magnitude of entropy generated. The entropy generated due to the oblique shock standing off the flare is much greater than the entropy generated by the weak viscous shock. This indicates the flow passing through the oblique shock experiences strong non-equilibrium, while the flow in the region of the fore-body does not. An enlarged view of the $x/L = 0.2$ profile shows the peak magnitude of the entropy generation here is roughly ten times less than for the oblique shock.

Figure 14 highlights the differences between the CFD and DSMC solutions. The DSMC solution predicts the oblique shock to stand further from the surface and also predicts a much thicker shock. These trends appear in other recent research (Lofthouse and others, 2006; Schwartzwelder and others, 2006; Schrock, 2005). The CFD peak entropy generation at $x/L = 0.7$ is much greater than the DSMC. This is probably because CFD predicts a thin shock, so the entropy must climb through steep gradients within the shock to satisfy jump conditions. To achieve this, CFD predicts a very high peak of entropy generation in the thin shock.

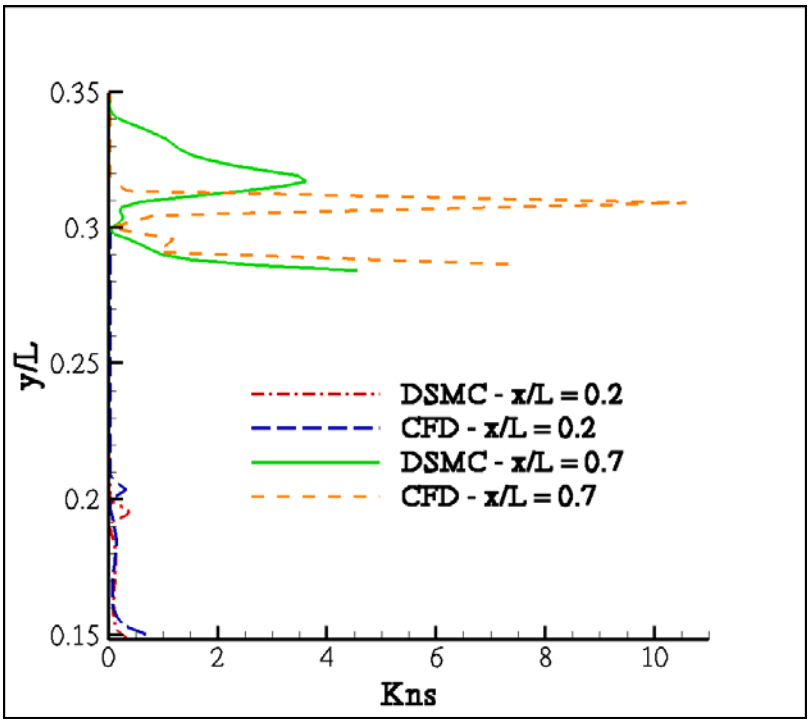


Figure 14. Run 5, Entropy generation profile comparison for x/L = 0.2 and x/L = 0.7

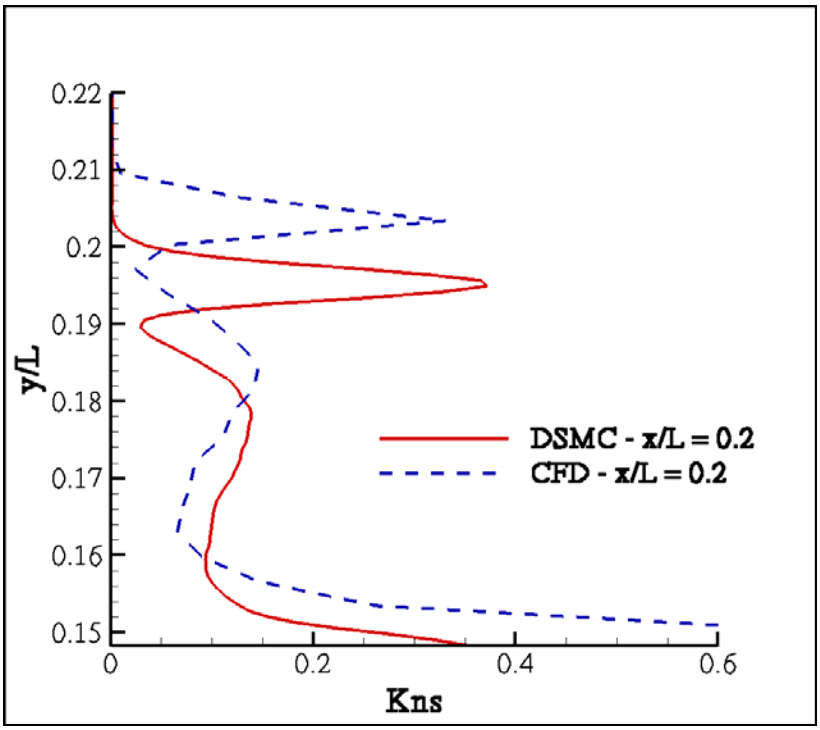


Figure 15. Run 5, Entropy generation profile comparison for x/L =0.2

The previous figures all compared a solution obtained using DSMC with a solution obtained using CFD. It is interesting to compare these solutions, but it is not correct to state all differences are due to non-equilibrium effects modeled in the DSMC solution. Although it is tempting to say this, there are too many differences between the two solution methods to make such a general statement.

However, the goal is to understand how non-equilibrium might affect a solution; particularly how non-equilibrium affects the values of the shear and heating and where the non-equilibrium effects are manifest. Unfortunately comparing the values of shear and heating obtained with the CFD solution with the values obtained by the DSMC solution is a bit like comparing apples to oranges. The flow fields themselves are distinct between the two solutions, so the values of the constitutive relations, dependent on flow field parameters, will thus be correspondingly different and it will be impossible to see where the constitutive relations differ in terms of non-equilibrium.

For this reason, another method of comparison is proposed. It is possible to calculate the shear stress and heat flux from parameters obtained via the DSMC solution using the Navier-Stokes constitutive relations, equation (II.55) and equation (II.56). These then could be compared to the shear and heating values generated by the kinetic theory method in equations (III.3) and (III.4). To state it simply: use the same solution field (from DSMC), use different constitutive relations (continuum and kinetic). This will enable a look at the differences between the two methods of computing the constitutive relations. Because the shear stress tensor has six distinct components and the heat flux

vector has three, the entropy generation, a convenient scalar quantity which is made up of both shear and heating, will be used to make this comparison.

Continuum and kinetic Kn_S profiles are shown in Figure 16 and Figure 17 for the two vertical extractions. These figures display large spikes in entropy generation coinciding with shock locations. The most obvious trend is that in the shock, the kinetic method predicts a higher peak value of entropy generation with a thicker shock. The thicker shock signifies the kinetic method predicts a larger region of non-equilibrium than predicted by the constitutive relations. Also, the higher peak entropy generation indicates a greater degree of non-equilibrium. The continuum method underestimates the total amount of entropy generated. These results corroborate observations by Schrock that DSMC in general predicts that non-equilibrium effects are more widespread in the flow than when calculated by the Navier-Stokes equations. The constitutive relations used in the Navier-Stokes equations to calculate shear and heating break down in the presence of non-equilibrium.

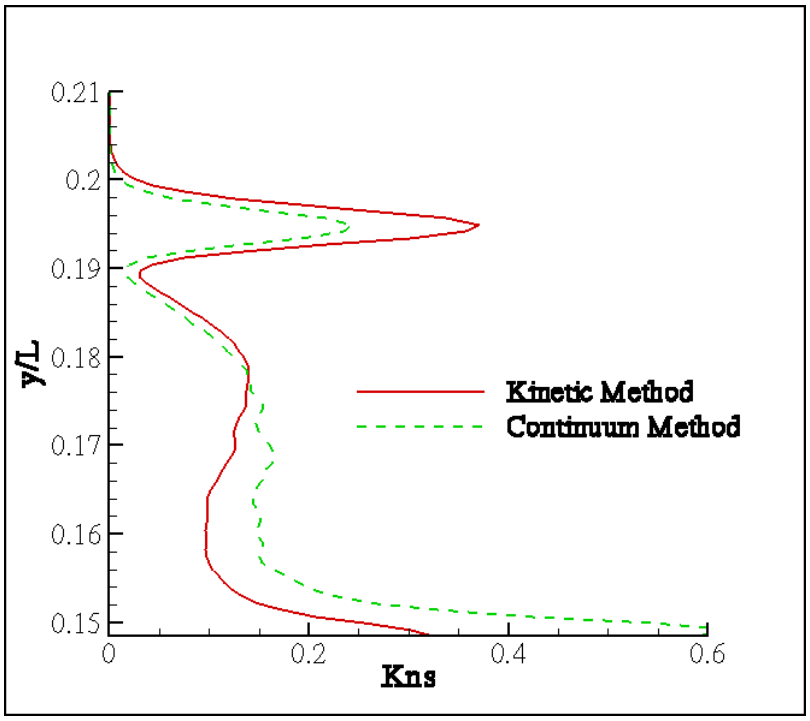


Figure 16. Run 5, Entropy generation by kinetic and continuum methods for x/L = 0.2

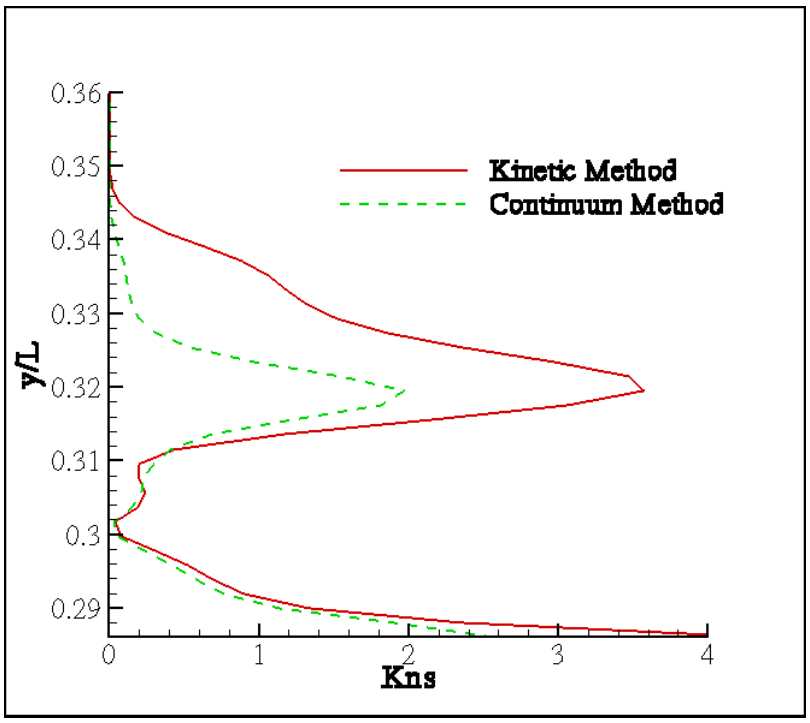


Figure 17. Run 5, Entropy generation by kinetic and continuum methods for x/L = 0.7

It is also possible to investigate the contributions of the rotational and vibrational modes to the entropy generation. Figure 18 and Figure 19 show these contributions for both locations. As would be expected, the translational entropy generation contributes the most to the total. The peak value of the translational entropy generation is greater than the value predicted by the continuum approach as seen in Figure 16 and Figure 17. This difference is due purely to non-equilibrium in the translational energy. The vibrational mode contributes almost nothing, as the characteristic temperature of vibration is 3,390 K, well above any temperature in this flow.

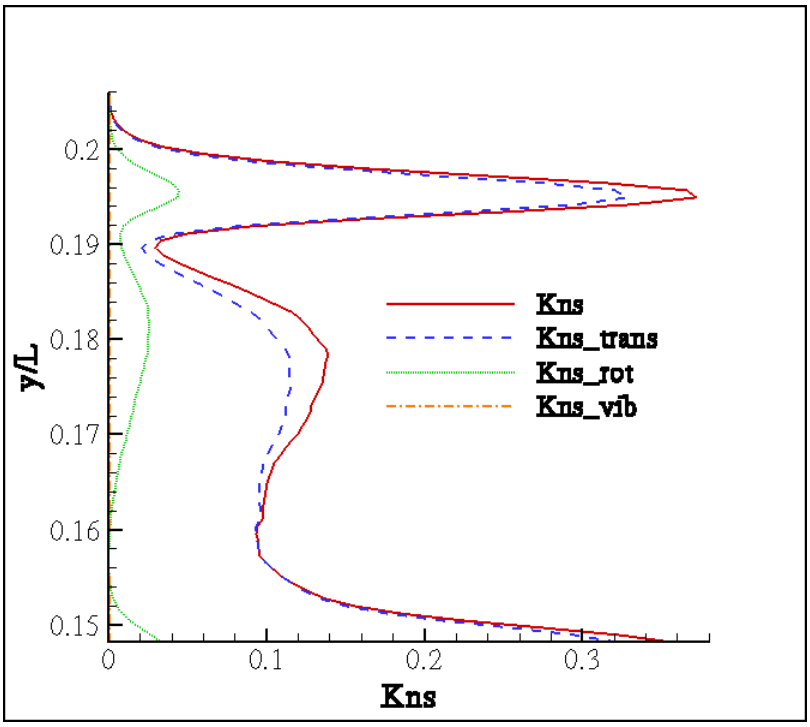


Figure 18. Run 5, Internal energy contributions to entropy generation, x/L = 0.2

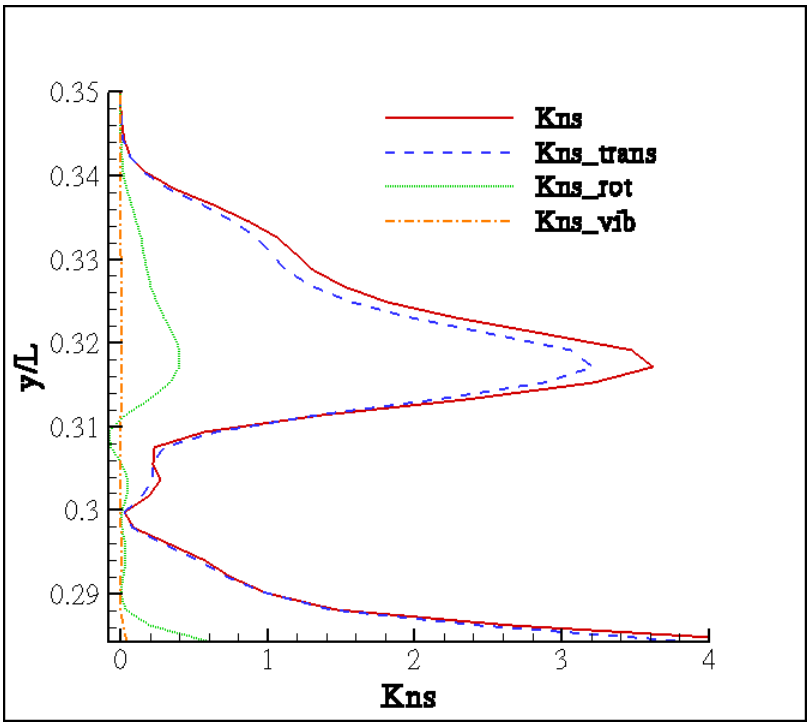


Figure 19. Run 5, Internal energy contributions to entropy generation, x/L = 0.7

The internal energy contributions to entropy generation may be a useful tool to extend the usefulness of the constitutive relations. The bulk viscosity, sometimes known as the dilatation viscosity because of its connection with expanding or contracting gases, could provide a means to do this. As previously mentioned, the bulk viscosity makes no contribution for a dilute monatomic gas. However, non-equilibrium in the distribution of internal energy can cause the bulk viscosity to become non-zero (Vincenti and Kruger, 1967: 407- 412). With knowledge of non-equilibrium for both rotation and vibration, it may be possible to develop a formulation for the bulk viscosity which can be applied to the constitutive relation for shear. An example of how the inclusion of bulk viscosity can improve the shear is given later in this chapter.

Run 7 Analysis

The wall data comparisons between DSMC, CFD, and experimental data are displayed for run 7 in Figure 20 and Figure 21. Again, both solvers have difficulty calculating the region where the shock from the fore-body impinges upon the boundary layer of the rear cone.

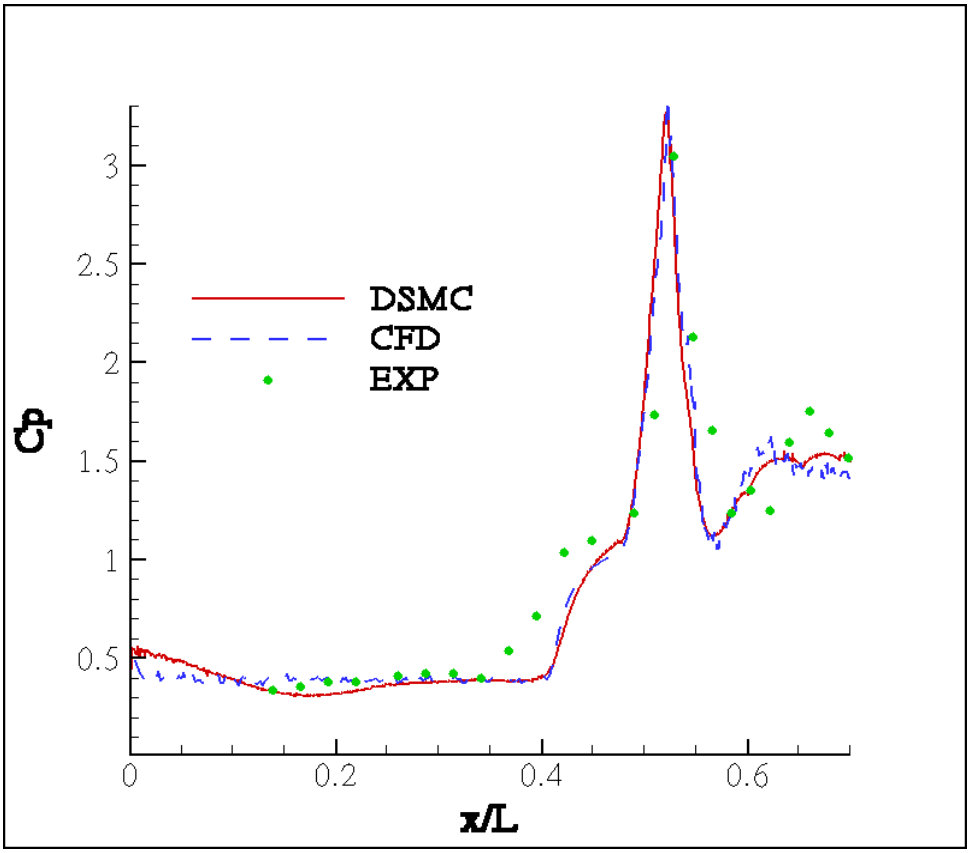


Figure 20. Run 7, Coefficient of pressure comparison

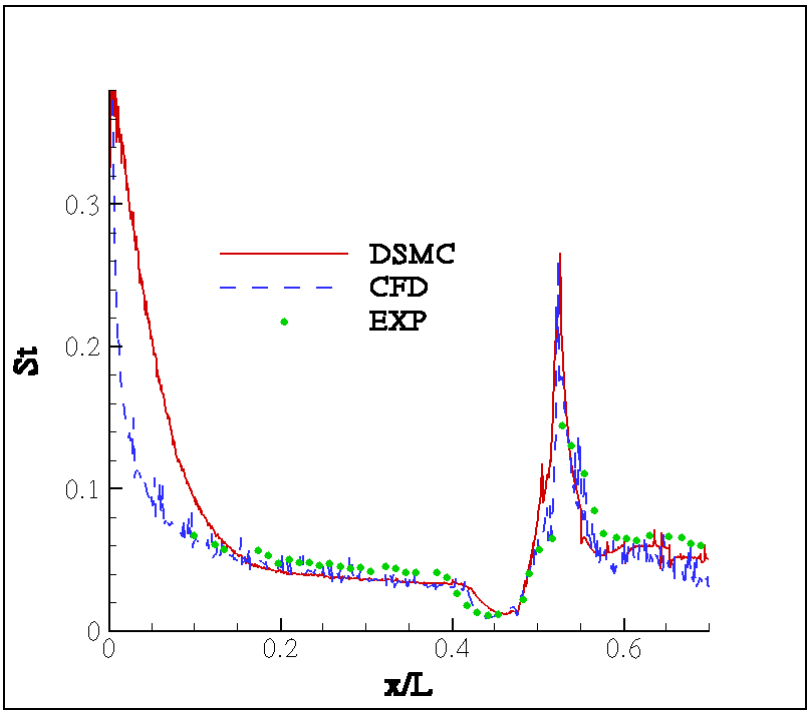


Figure 21. Run 7, Stanton number comparison

Temperature and entropy generation contours of run 7 are displayed in Figure 22 and Figure 23. The geometry here is similar to case 5; however, the oblique shock standing off the 25-degree angle fore-body of the cone for case 7 is stronger than the viscous induced shock standing off the surface of the cylinder in run 5. Again, a pocket of entropy generation exists at the leading edge due to transfer of kinetic energy to internal modes. This curves the shock at the leading edge because internal energy modes are activated and must relax as the flow travels downstream. The angle of the shock at the leading edge should correspond to a theoretical "frozen" prediction, meaning the reaction rates are essentially zero. Further downstream, the shock angle will correspond to an equilibrium prediction, meaning the reaction rates are infinite. The actual shock angle curves from the frozen to the equilibrium value. This same behavior appears in the oblique shock standing off the second angle of the cone.

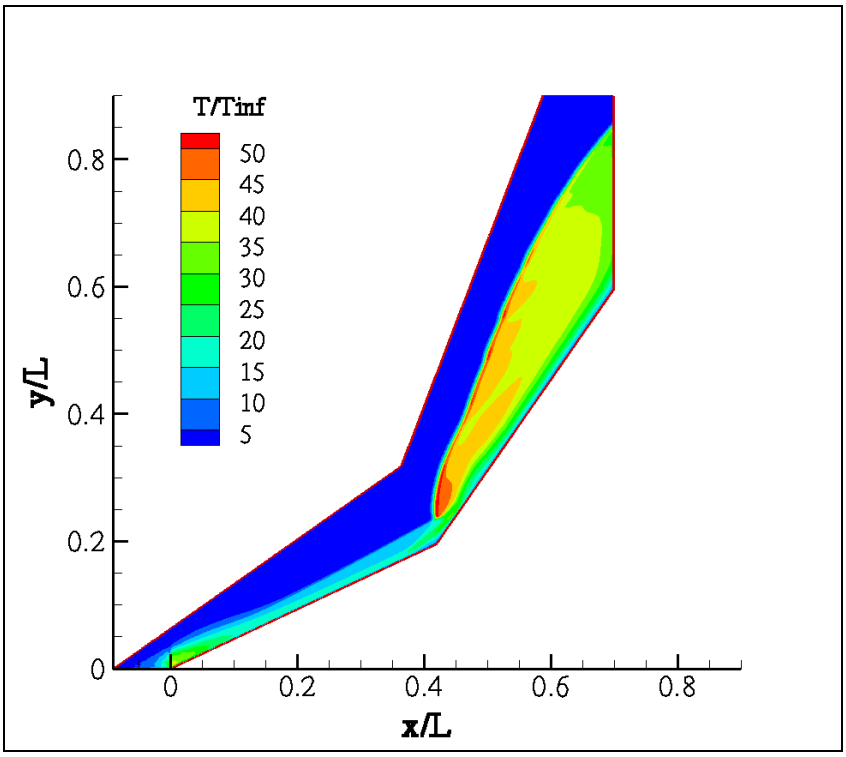


Figure 22. Run 7, Non-dimensional temperature contours

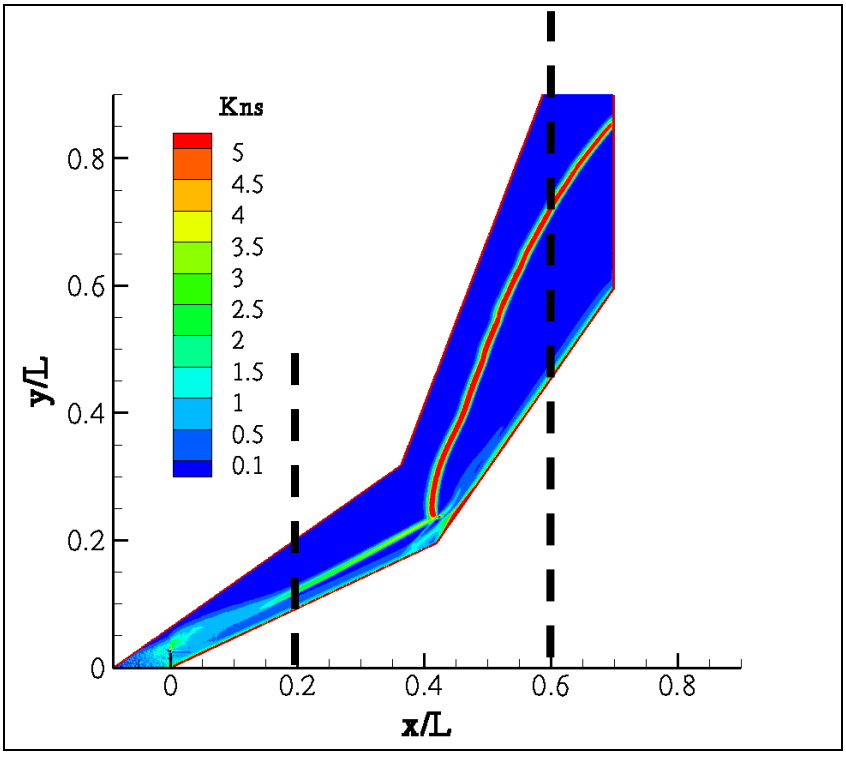


Figure 23. Run 7, Entropy production (Kn_S) contour lines

Another interesting feature of this flow is the impingement of the oblique shock from the 25-degree fore-body cone upon the wall of the second 55-degree cone. The incident shock wave interacts with the viscous boundary layer. The abrupt pressure change from the shock is an adverse pressure gradient potentially causing the flow to separate from the surface of the cone, occurring ahead of the impingement site. This phenomenon is observed in run 7 in the magnified picture in Figure 24.

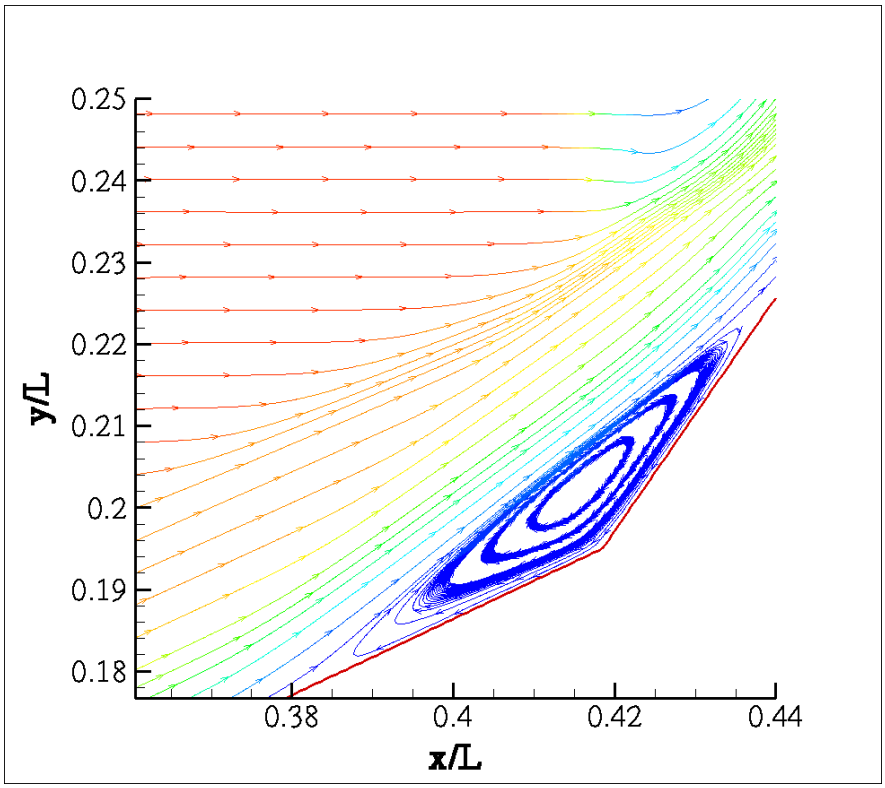


Figure 24. Streamlines showing separation caused by shock impingement on a boundary layer

The separation induces a shockwave, which combines with a reattachment shock wave to form a single oblique shock standing off the surface of the second cone. This flow scenario is very complicated, but important to understand due to the high thermal loads

transferred to the wall at the point of impingement as seen in the wall $x/L = 0.5$. This type of impingement has caused structural damage to hypersonic vehicles in the past (Anderson, 1989: 321-322).

Data was extracted vertically at $x/L = 0.2$ and at $x/L = 0.6$ as shown by the dashed lines in Figure 23. Figure 25 and Figure 26 show temperature and entropy generation profiles for the two extractions.

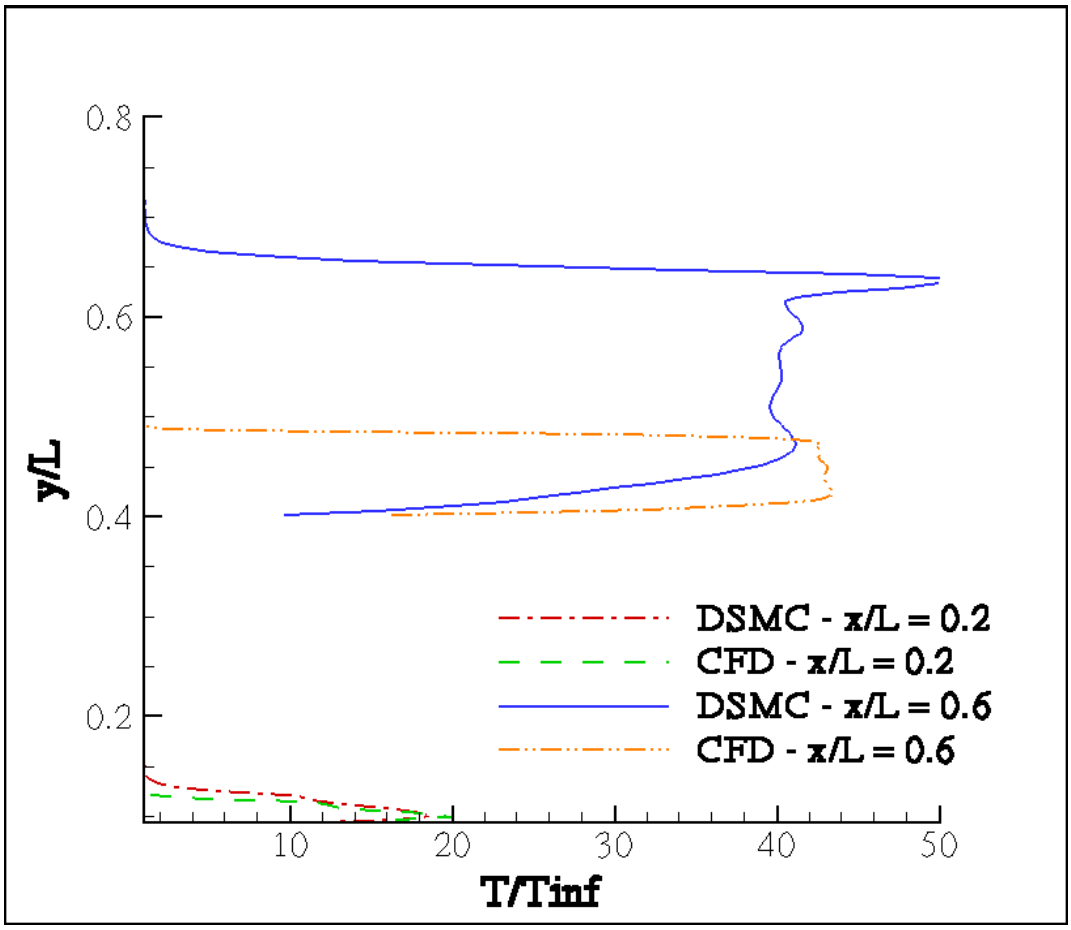


Figure 25. Run 7, Temperature profile at x/L = 0.2 and x/L = 0.6

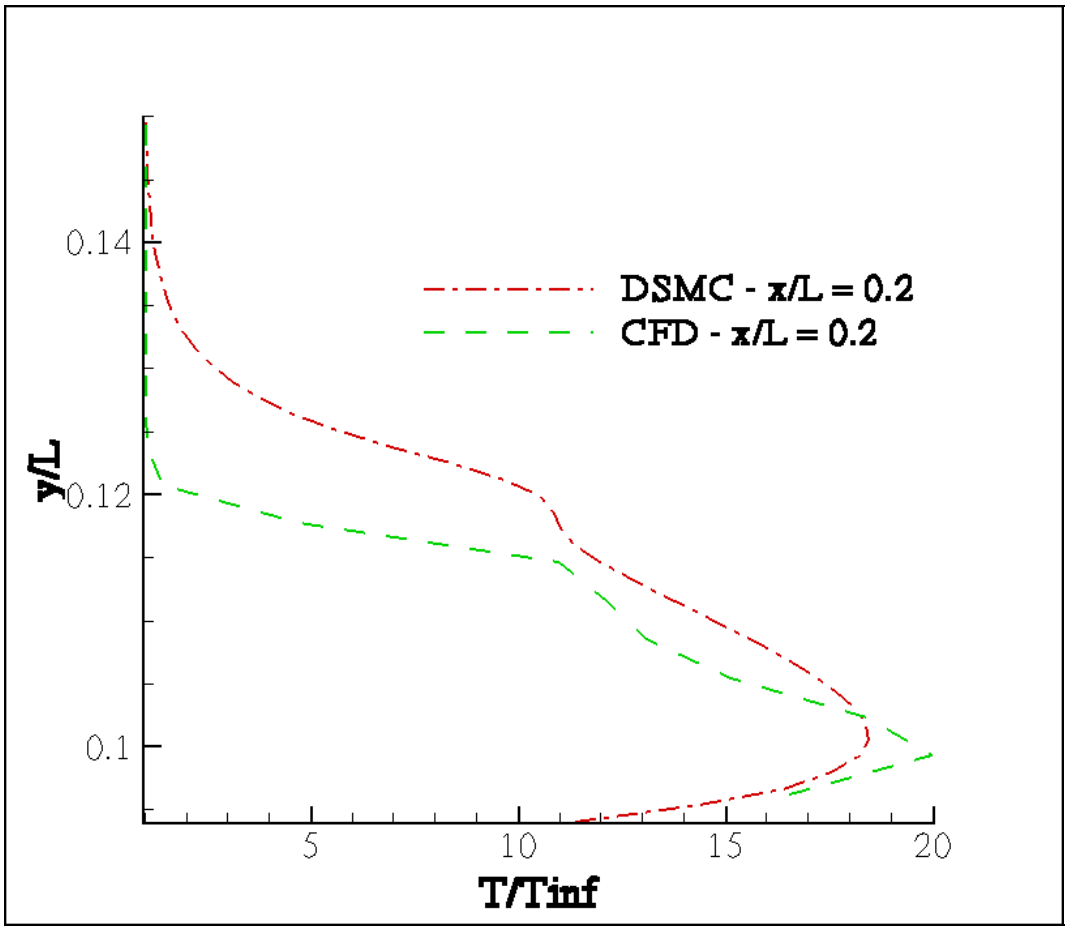


Figure 26. Run 7, Magnified temperature profiles for x/L = 0.2

The temperature profiles suggest the shock generated by DSMC stands further from the surface of the double cone. Also, the CFD shock is much thinner at $x/L = 0.6$. Figure 27 shows the entropy generation profiles for $x/L = 0.2$ and $x/L = 0.6$, while Figure 28 shows an enlarged view of the profile at $x/L = 0.2$.

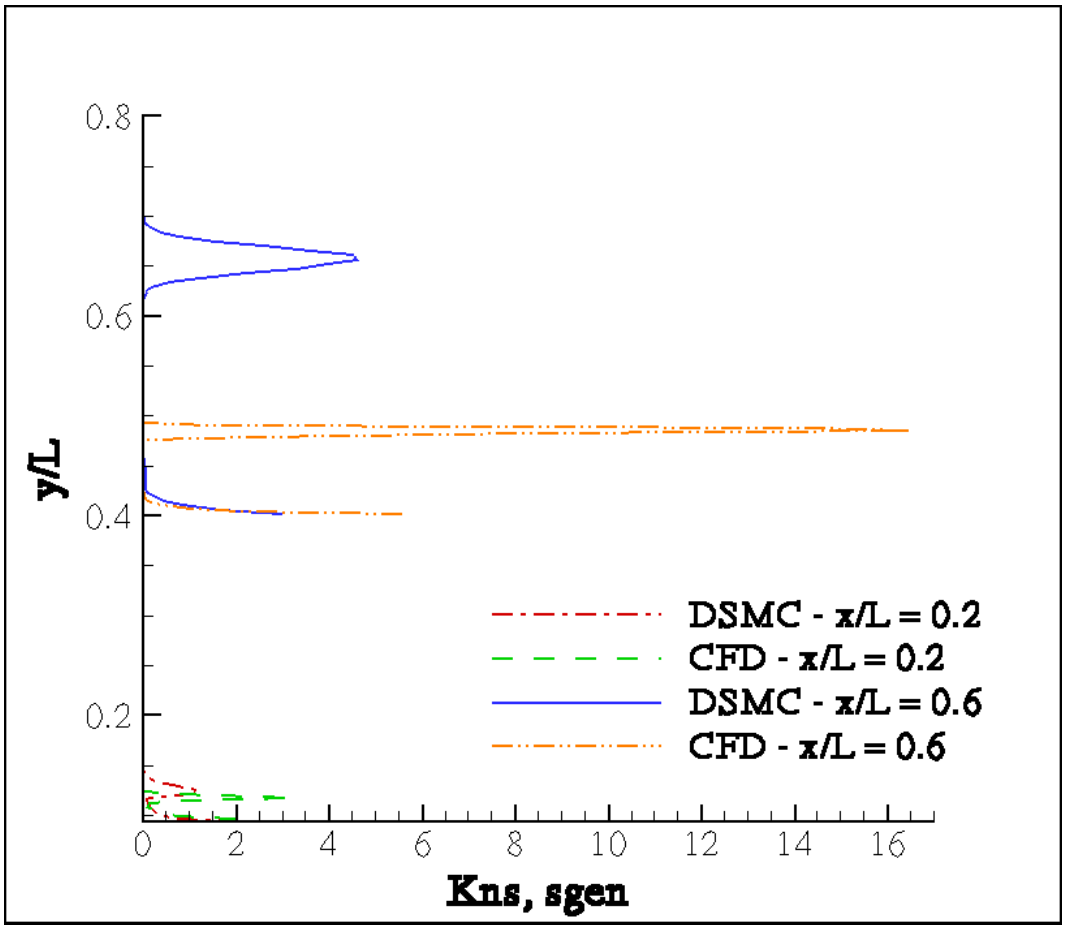


Figure 27. Run 7, Entropy generation profiles for x/L = 0.2 and x/L = 0.6

The DSMC shock stands further from the surface than the CFD shock for both profiles; however, the difference is much greater for the case of $x/L = 0.6$. This corresponds to a greater peak value of entropy generation. The amount of difference in the standoff from the surface between the DSMC and CFD solutions increases with x/L, showing the shock angles are different. The DSMC code predicts a greater angle between the shock and the wall of the cone. Again, the DSMC shock is much thicker and the high entropy generation peak in the CFD data is due to the thin shock. Even if the two shocks centered

in the same *y/L* location, the DSMC solution would predict non-equilibrium before the CFD, as evidenced in the difference in widths.

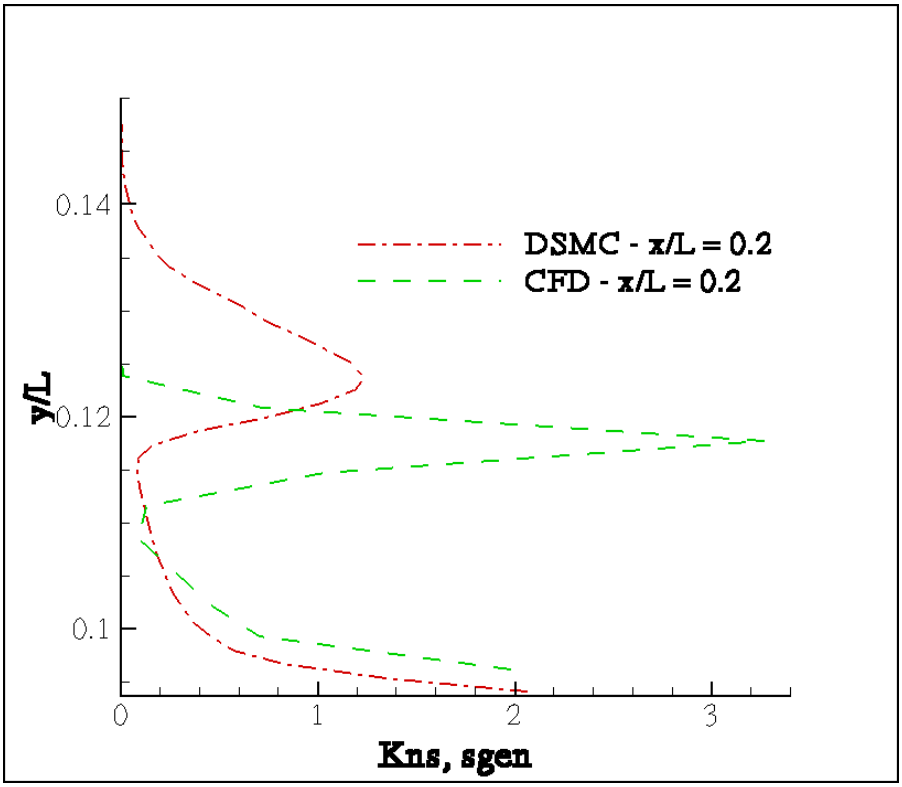


Figure 28. Run 7, Magnified entropy generation profiles for x/L = 0.2

Similar to the analysis of run 5, the same flow field (DSMC) is used to compare the kinetic and the continuum formulations of calculating the entropy generation,. The resulting entropy generation profiles for *x/L* = 0.2 and *x/L* = 0.6 are shown in Figure 29 and Figure 30. Again, note the shock thickness calculated by the kinetic method is greater than the shock thickness calculated by the continuum method, even when using the same solution field. It is likely that the equilibrium limitation in the continuum constitutive relations leads to the thin shocks in the CFD solution.

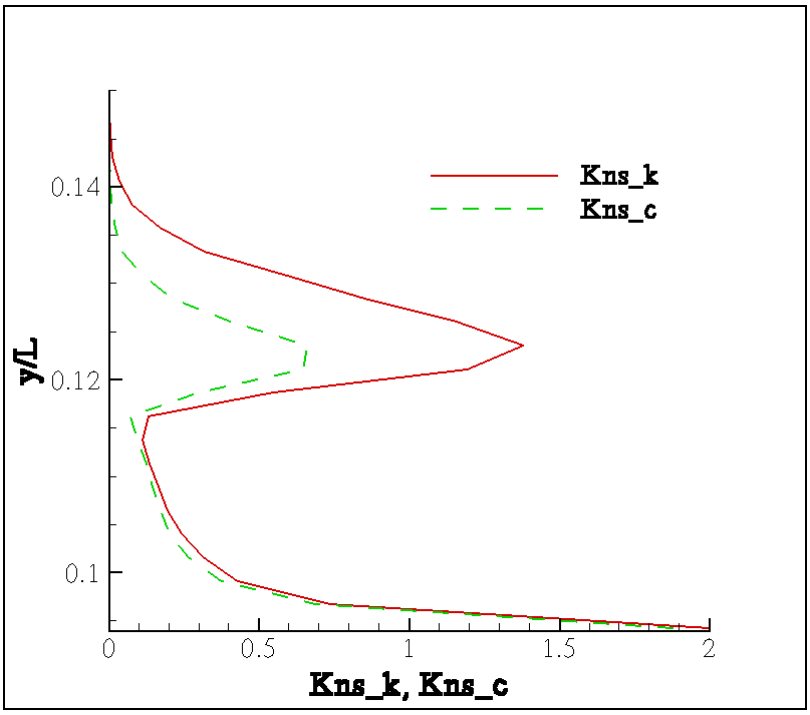


Figure 29. Run 7, Comparison of kinetic and continuum entropy generation, x/L = 0.2

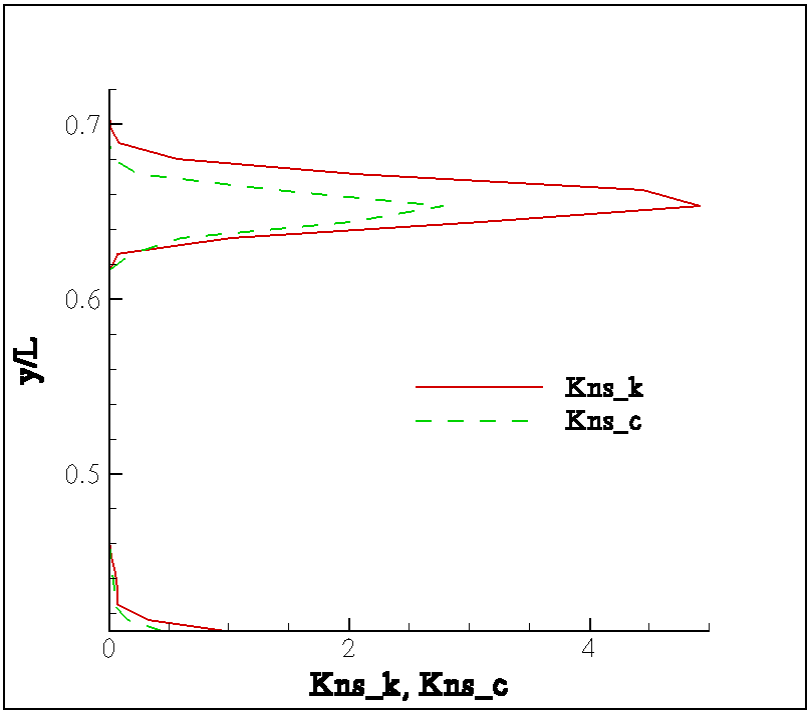


Figure 30. Run 7, Comparison of kinetic and continuum entropy generation, x/L = 0.6

It is also possible to investigate the effect that each of the energy modes has on the entropy generation. For $x/L = 0.2$, Figure 31 indicates that the translation entropy generation (including the shear terms) contributes the most to the overall total and again the vibrational mode is barely activated. Figure 32 shows the contributions for $x/L = 0.6$. These figures give a unique perspective on the extent to which the internal energy modes activate. Any formulation using the perfect gas assumption could not be expected to model the effects from the internal energy modes.

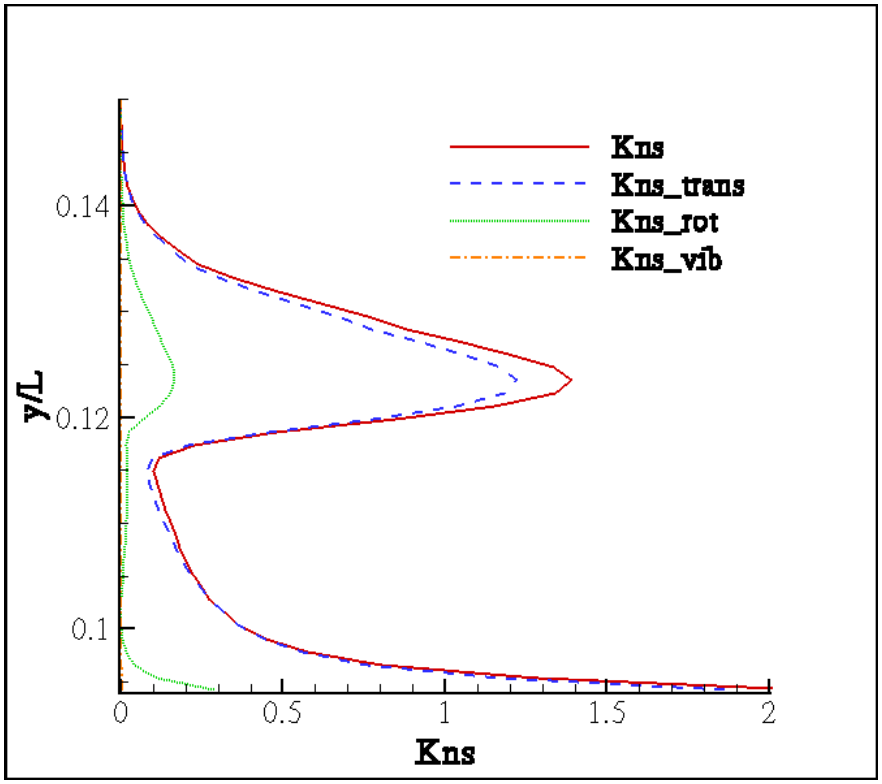


Figure 31. Comparison of internal energy contributions to the entropy generation, x/L = 0.2

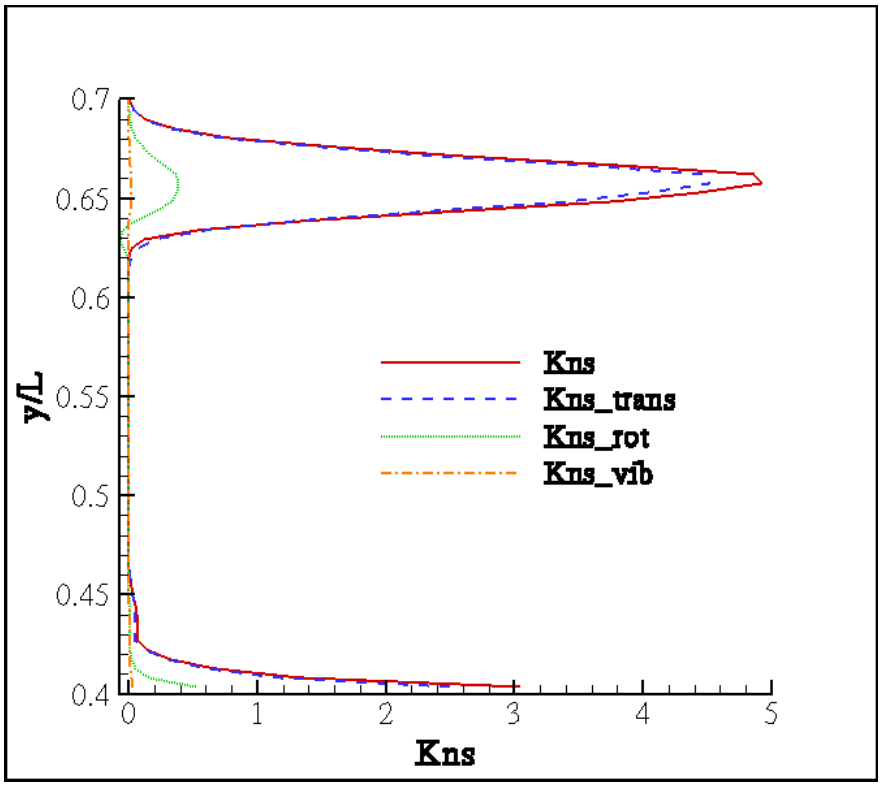


Figure 32. Internal energy contributions, x/L = 0.6

Modifications to Viscosity, Thermal Conductivity, and Bulk Viscosity

The difference between the continuum constitutive relations and the kinetic model results partially from assumptions made in modeling the viscosity and thermal conduction coefficients. Although the limits with these models are normally associated with high temperature activation of internal energy states, it is conceivable that different approximations need to be stated at low temperature. Sutherland's viscosity model given in equation (II.32) assumes hard spheres with an attraction potential. The Lennard-Jones model offers increased temperature flexibility by introducing a strong repulsive potential at short distances. This introduces an extra parameter into Sutherland's Law (Chapman

and Cowling, 1952: 227-229; Bird, 1994: 43; Bird and others, 1960: 22; Hirschfelder and others, 1954: 22):

$$\mu = \mu' \left(\frac{T}{T'} \right)^{\frac{3}{2}} \frac{S + T'^{\frac{\nu-3}{\nu-1}}}{S + T^{\frac{\nu-3}{\nu-1}}} \qquad \text{(IV.2)}$$

The term ν gives an extra knob to turn, allowing a larger range of temperatures. The S term has taken on a slightly new meaning, now being a measure of both attractive and repulsive potential. Thermal conductivity is modeled similarly based on its relation to viscosity given in equation (II.31). For $\nu = \infty$, this equation returns the Sutherland's viscosity given by equation (II.32). This model has been implemented for both viscosity and thermal conductivity.

For both run 5 and 7, the Lennard-Jones model for the viscosity coefficient leads to a better fit with the kinetic data. The choice of ν and S vary depending on the temperature and type of gas in the flow. It was found that, as compared to Sutherland's law, an improved fit to the kinetic data was found using $\nu = 4.0$ and $-32 < S < -12$. The optimal value of S is slightly lower for run 7 than it is for run 5. This is not surprising because although the geometry and the freestream conditions of the two runs are similar, there are substantial differences in the flows. The temperature range of run 7 is about twice that of run 5, as observed by compariring Figure 13 and Figure 25.

It is interesting that the best fit was found for negative values of S. Differing from Sutherland's model, this constant has become a measure of the attractive-*repulsive* force between molecules. The negative values indicate the repulsive force has become

important in the flow. Figure 33 and Figure 34 show these fits at $x/L = 0.6$ for run 7 for a single component of the shear stress tensor and the heat flux vector, respectively.

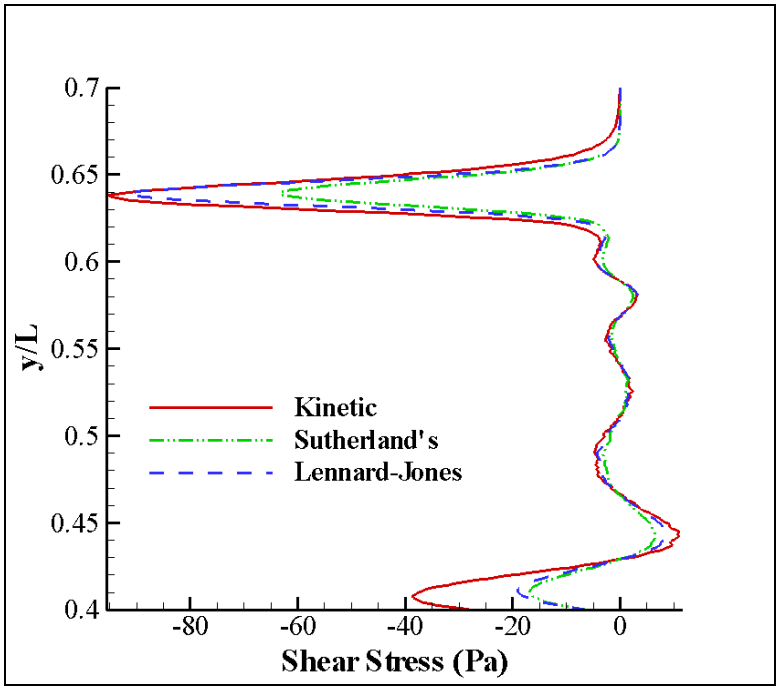


Figure 33. Run 7, comparison of kinetic, Sutherland's Law, and Lennard-Jones models for the shear stress in the *x*-*x* direction at *x/L* = 0.6

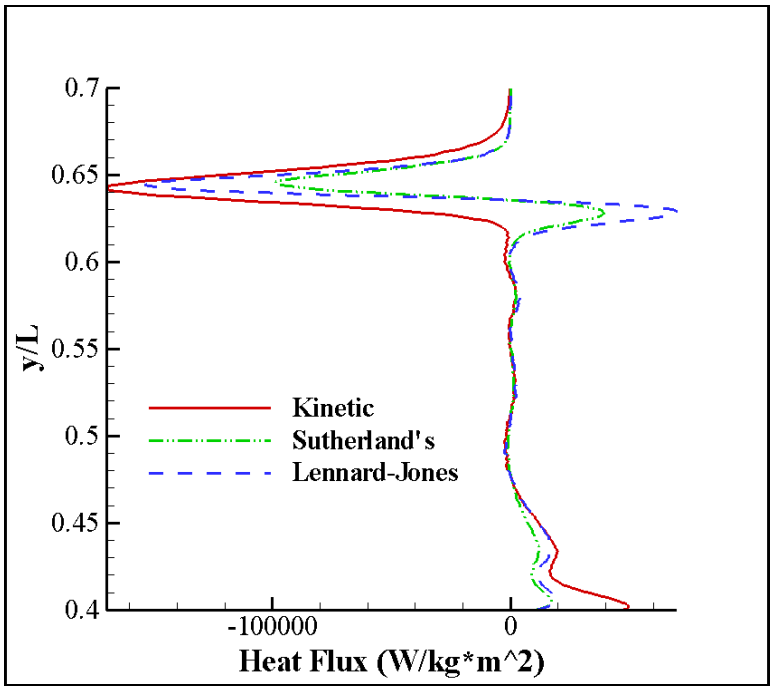


Figure 34. Run 7, comparison of kinetic, Sutherland's Law, and Lennard-Jones models for the heat flux in the *x*-direction at *x/L* = 0.6

The fit has been improved substantially by the use of the Lennard-Jones model. Other combinations of *v* and *S* may also yield good results, indicating the need for more trial iterations using this model. Inspection of Figure 34 shows a positive, dispersion-like second peak in heat flux just downstream of the shock. This can be damped out by addition of the internal energy contribution to heat flux as acquired by the kinetic formulation (as well as improving the fit in the main shock portion). In other words, this effect is due to not properly including internal energy effects into the continuum model.

In addition to quantifying non-equilibrium, entropy generation combines all of the components of the shear and heating into a single convenient scalar. A final look at the entropy generation in Figure 35 shows the Lennard-Jones model does indeed improve the continuum solution. The improvements are not as marked as what is seen for the

individual components of shear and heating. This is because the dispersion wave seen in the heating terms seems to decrease the Lennard-Jones entropy generation peak and shift the entire shock toward the wall. This shift indicates that failing to adequately predict internal energy in the thermal conductivity delays the prediction of non-equilibrium in the flow. It was mentioned that the dispersion could be damped somewhat by including the internal energy contributions found from the kinetic solution to the heat flux. This damping improves the entropy generation fit. Unfortunately, the rotational energy activation is not available when computing continuum data. Additional research incorporating internal energy effects in the thermal conductivity calculation may yield a better match.

The next proof of the validity of the fit found here is to use the Lennard-Jones model in a CFD simulation. The same CFD grids used in this document could be used for the initial run of the simulations. The grid could then be re-adapted to the temperature gradient. It is expected that the new viscosity and thermal conductivity relations will result in a closer fit to the DSMC solution. It is expected that this fit could be used in a CFD model to analyze any flow with similar geometry and freestream conditions.

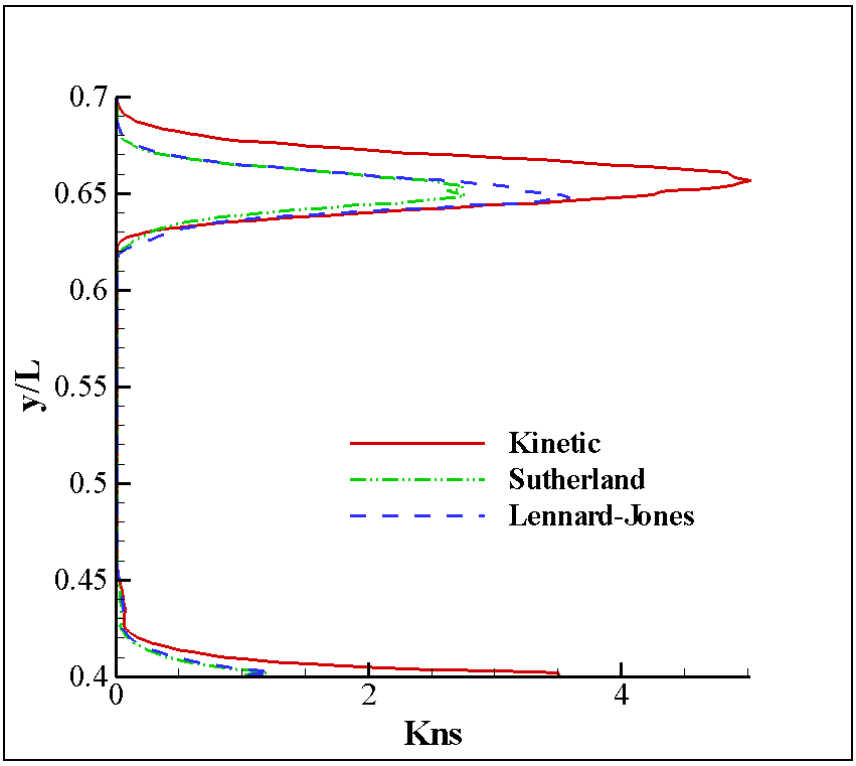


Figure 35. Run 5, Entropy generation comparison between kinetic, Sutherland, and Lennard-Jones models

Recall that both the models for calculating viscosity and thermal conductivity were based upon the first Sonine approximation to the Chapman-Enskog solution. By including more terms in the Sonine expansion, it is possible to improve upon the viscosity and thermal conductivity predictions, especially for high or low temperature cases (Chapman and Cowling, 1952: Appendix C).

Further improvement can be made to the continuum shear relation by inclusion of the bulk viscosity. This term is contained in the traditional N-S formulation for the shear stress tensor given in equation (II.55). Stoke's Hypothesis dealt with the bulk viscosity term by assuming that $3\lambda + 2\mu = 0$, where again, $\lambda \equiv \mu_B - (2/3)\,\mu$. This is equivalent to the Chapman-Enskog solution to the Boltzmann equation that for a monatomic gas near

equilibrium $\mu_B = 0$. This fact naturally leads one to theorize that non-equilibrium may result in a non-zero value of bulk viscosity due to activation of internal energy modes. A small value of bulk viscosity may result in a better fit to the kinetic data, as shown in Figure 36.

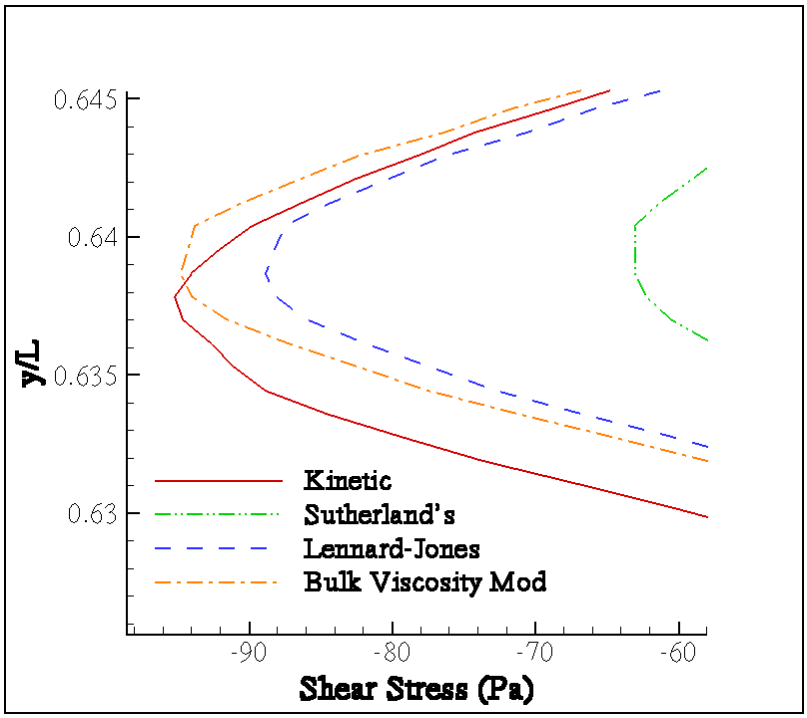


Figure 36. Run 7, Comparison showing improvement to the shear stress in the x-x direction, only the shock peak is shown as the bulk viscosity only has effect in regions of compression

This figure demonstrates how the proper amount of bulk viscosity (here about 7% of the kinematic viscosity) can improve the solution. The changes are only applied inside the shock. This is a consequence of the divergence term associated with the bulk viscosity in the shear stress tensor (equation (II.55). The divergence term is often associated with the conservation of mass and is a measure of how much the fluid is expanding or contracting. Therefore, the bulk viscosity only has an effect inside the shock where there is a sudden compression of the flow. The physical mechanism of bulk viscosity is due to the increase

in translational energy due to density changes. The subsequent internal energy increase comes only after a certain relaxation time. The ratio of translational energy to the total amount is greater than it normally would be at equilibrium, effectively introducing a pressure to oppose the contraction of the gas. The opposite is true of expansion. Bulk viscosity is a way of keeping track of this non-equilibrium imbalance between translational and internal energy; essentially a compression/expansion damping term (Chapman and Cowling, 1952; 396). Future study should seek a formulation for bulk viscosity as a function of internal energy activation. Suggestions for this relation are found in the literature (Hirschfelder and others, 1954: 503; Vincenti and Kruger, 1965: 407 – 412).

It should be noted that the kinetic formulation for the shear stress and heat flux obtained via DSMC is not perfect. As described earlier, the DSMC method simulates flow by modeling molecules as they interact with each other and with boundaries such as walls. The validity of the DSMC method is only as good as the physics used to model the interaction of the molecules. The value of DSMC is that the user is mostly limited by his or her ability to apply good models. It is probable that the physical models used in any particular DSMC code do not offer perfect agreement with reality. This would have an effect on the second order moments used to calculate shear, and certainly the third order moments found in the heat flux would suffer due to imperfect collision models. For this reason, it is recommended that future study focus not only on improving values used within the continuum constitutive relations, but also on implementing better molecular

models within the DSMC process itself. Other codes may be chosen which demonstrate closer fits to experimental data (see Harvey, 2003).

V. Conclusions

In order to gain a deeper understanding of non-equilibrium in a flow, it is useful to examine the fluid as a collection of particles, rather than as a continuous mass. Kinetic theory models the physics of particles and thus is not overly constrained by equilibrium. The Chapman-Enskog solution to the Boltzmann equation links kinetic theory with the traditional equations used in CFD, like the Navier-Stokes. This solution also demonstrates the constitutive relations for shear and heating used in the Navier-Stokes equations are invalid for highly non-equilibrium situations. Because these constitutive relations are so commonly used in modeling, it is important to understand why and when they become invalid.

Entropy generation is an indicator of non-equilibrium. Boltzmann's *H*-theorem supports this assertion, showing that entropy is produced when a PDF is disturbed from equilibrium. This research theorizes entropy generation can serve to indicate exactly how much the velocity distribution has deviated, or in other words, how much non-equilibrium is present in the flow.

To measure entropy generation the method being used must be able to capture non-equilibrium effects. For this reason, Direct Simulation Monte Carlo was used in comparison with a standard CFD package. Entropy generation was derived based on the Gibbs equation in conjunction with the conservation of mass, momentum, and energy. Equilibrium assumptions only enter when forming the constitutive relations for shear

stress and heat flux. However, it is possible to express these two using kinetic theory, and reliably model non-equilibrium flows.

Two cases computed by CFD and DSMC were compared. Variations between the two solution methods were discussed in the context of non-equilibrium effects. In general, an oblique shock predicted by DSMC tends to stand further away from the surface of the body and is much thicker than the shock predicted by CFD. Additionally, because the shock predicted by CFD is much thinner, it tends to over-predict the peak height of the entropy generation in order to meet entropy conditions on the downstream side of the shock.

In order to understand where the traditional constitutive relations become invalid, shear stress and heat flux have been calculated based on the same solution field using two methods. In the first, the shear and heating were calculated using the traditional continuum method. This method uses the coefficients μ, for viscosity, and K, for thermal conductivity. Kinetic theory gives formulations for the viscosity and thermal conductivity based on empirical constants. One such method, known as Sutherland's law, was used here. Sutherland's law is inadequate for certain regions of these flows, and an alternate formulation is proposed for future study.

The second method of calculating the shear stress and heat flux benefits from the DSMC simulation, where the velocity and internal energy state of each particle are known. The constitutive relations were calculated based on their kinetic definitions as expectation quantities of various moments of velocity and internal energy. The relationships developed were free from equilibrium assumptions. For this reason, a comparison of the

two methods highlights the equilibrium limitations in the continuum constitutive relations.

The kinetic method predicts a larger shock region than the continuum method. There was a region just upstream and downstream of the continuum shock where non-equilibrium conditions exist, but not captured by the continuum constitutive relations. Because the continuum shock is thin, and the entropy must increase across the shock regardless of the width of the shock, the peak entropy generation is very high in the continuum solution.

The main source of discrepancy between the kinetic method and the continuum method of computing the shear and heat flux results from using Sutherland's law. This model includes only attractive forces.The Lennard-Jones model can be applied over a broader range of temperatures by representing both attractive and repulsive forces between molecules. An improved fit to the kinetic data is found. The thermal conductivity formulation can be improved by inclusion of internal energy contributions.

The Chapman-Enskog solution and evidence from the literature suggest the bulk viscosity may be used to account for some amount of non-equilibrium due to compression or expansion of a flow. A small amount of bulk viscosity was added to the continuum shear stress resulting in an improved match with the kinetic prediction for shear. A formulation for bulk viscosity may be found to further improve the continuum shear formulation.

Once satisfactory formulations for the viscosity, the thermal conductivity, and the bulk viscosity are found, the next step is to apply them to a CFD code and obtain a solution,

rather than simply calculating their values post-process. The solution would then be compared with the DSMC solution and available experimental data. Tuning the constitutive relations to account for non-equilibrium effects would effectively extend the useful range of the Navier-Stokes equations. Unlike previous methods introducing new variables and equations, this method is relatively simple, conceptually an empirical fit based on DSMC data rather than experiment. Once a single DSMC solution is generated and a suitable fit is found, the same parameters may be applied to calculate viscosity and thermal conductivity for a variety of flows with similar geometries and freestream conditions. The limits of this method are still unexplored. While this method of extending the valid range of the continuum equations may not be quite as elegant as some of the complicated versions mentioned in this document, its simplicity and effectiveness is compensation enough to merit further study.

References

Anderson, J.D., *Modern Compressible Flow*, 3rd edition, McGraw-Hill, New York, 2003.

Anderson, J.D., *Hypersonic and High Temperature Gas Dynamics*, McGraw-Hill, New York, 1989.

Baganoff, D., "A Glimpse of Hydrodynamics Beyond the Navier-Stokes Equations", *Physics of Fluids*, Volume 14, Number 10, October 2002.

Barr, Larine. "Multi-national Agreement to Advance High-Speed Flight." News Article in *news@afrl*, the official Air Force Research Laboratory newsletter. December 2006. http://www.afrl.af.mil/news/dec06/features/agreement_flight.PDF

Bertin, J.J., *Hypersonic Aerothermodynamics,* AIAA Education Series, American Institute of Aeronautics and Astronautics, Inc., Washington D.C., 1994.

Bird, R.B., Stewart, W.E., Lightfoot, E.N., *Transport Phenomena*, John Wiley & Sons, Inc., New York, 1960.

Bird, G.A., *Molecular Gas Dynamics and the Direct Simulation of Gas Flows*, Oxford University Press, New York, 1994.

Bird, G.A., Emeritus Professor, The University of Sydney, Australia. Personal conversation. November 2006.

Boyd, I.D., Wang., W.-L., "Monte Carlo Computations of Hypersonic Interacting Flows", Paper, A01-16820, AIAA, January 2001.

Burnett, D., 1935. "The Distribution of Velocities in a Slightly Non-uniform Gas. Proc." Lond. Math. Soc., Ser. 2, vol. 39, p. 385 (As reference in Vincenti and Kruger).

Camberos, J.A., "On the Construction of Entropy Balance Equations for Arbitrary Thermophysical Processes", Paper, A01-16633, AIAA, January 2001.

Camberos, J. A., and Chen, P. H., "Continuum Breakdown Parameter Based on Entropy Generation Rates". Paper, 2003-157, AIAA, January 2003.

Carr, R.C., Branam, R. D., Camberos, J.A., "Quantifying Non-Equilibrium Using Entropy Generation". Paper 2006-7967, AIAA, November 2006.

Chapman, S., and Cowling, T.G., *The Mathematical Theory of Non-Uniform Gases*, Cambridge University Press, London, 1952.

Chen, Xinzhong, Hongling Rao, and Edward A. Spiegel. “Continuum description of rarefied gas dynamics. I-III,” *Physical Review E*, Volume 64, 046308, 046309, 046310.

Comeaux, K.A., *An Evaluation of the Second Order Constitutive Relations for Rarified Gas Dynamics Based on the Second Law of Thermodynamics*. PhD dissertation, Stanford University, CA, April 1995.

EasyGui. Version 0.72, internet download. Stephen Ferg, http://www.ferg.org/easygui/index.html, 2004.

Eyring, H., Henderson, D., Stover, B.J., Eyring, E.M., *Statistical Mechanics and Dynamics*, John Wiley and Sons Inc., New York, 1964.

Fiscko, K.A. and Chapman, D.R. (1989). “Comparison of Burnett, Super-Burnett and Monte Carlo solutions for hypersonic shock structure”. Prog. In Aero. and Astro. 118, 374-395.

FLUENT, Software Package, Ver. 6.2.16, Centerra Resource Park, Lebanon, NH, 2004.

Grad, H. “Asymptotic Theory of the Boltzmann Equation”. *Physics of Fluids*, Volume 6, Number 2, Pages 147-181.

Harvey, J. K., “A Review of a Validation Exercise on the use of the DSMC Method to Compute Viscous/Inviscid Interactions in Hypersonic Flow”. Paper, 2003-3643, AIAA, June 2003.

Hirschfelder, J.O, Curtiss, C.F., Bird, R.B, *Molecular Theory of Gases and Liquids,* John Wiley & Sons, Inc., New York, 1954.

Holden, M. S., and Wadhams, T. P., “CUBDAT Database v4.0”, Database of Hypersonic Flows [CD-ROM], v4.0, Casplan Universisty of Buffalo Research Center, 2007.

“Honest Broker for Science and Technology, The,” *Defense AT&L*, January-February 2007.

Lofthouse, A.J., Boyd, I.D., Wright, M.J., “Effects of Continuum Breakdown on Hypersonic Aerothermodynamics”, Paper, 2006-993, AIAA, January 2006.

Pham-Van-Diep, G.C., Erwin, D.A., and Muntz, E.P. (1991). “Testing continuum descriptions of low Mach number shock structures”. J. Fluid Mech. 232, 403-413.

Python. Version 2.3.4, internet download. Computer software. Python Software Foundation, Ipswich MA, 2006. http://www.python.org

Schrock, C.R., *Entropy Generation as a Means of Examining Continuum Breakdown*. MS thesis, AFIT/GAE/ENY/05/M20, School of Engineering and Management, Air Force Institute of Technology (AU), Wright Patterson AFB, OH, March 2005.

Schrock, C.R., McMullan, R.J., Camberos, J.A., "Calculation of Entropy Generation Rates via DSMC with Application to Continuum/Equilibrium Onset". Paper 2005-4830, AIAA, January 2005.

Schrock, C.R., McMullan, R.J., Camberos, J.A., "Continuum Onset Parameter Based on Entropy Gradients Using Boltzmann's H-Theorem". Paper 2005-967, AIAA, January 2005.

Schwartzentruber, T.E., Scalabrin, L.C., Boyd, I.D., "Hybrid Particle-Continuum Simulations of Non-Equilibrium Hypersonic Blunt Body Flow Fields". Paper 2006-3002, AIAA, June 2006.

Tannehill, J.C., Anderson, D.A., and Pletcher, R.H., *Computational Fluid Mechanics and Heat Transfer, Taylor and Francis, Philadelphia*, PA, 1997.

Tecplot. Version 11.0–1-125. Software Package, CD-ROM Computer software. Tecplot Inc. Bellevue, WA, 2006.

Vincenti, W.G., and Kruger, C.H., *Introduction to Physical Gas Dynamics*, Wiley & Sons, Inc., New York, 1967.

White, F.M., *Viscous Fluid Flow*, 3rd ed., McGraw Hill, New York, 2006.

Appendix

Changes to the MONACO Source Code

File: *src/PHYS/count.c*

This file is used to calculate summations. These are stored in the *sums* structure as defined in the file *src/PHYS/cellphys.h*. The entire file has been included. Additions were made to calculate $<C_iC_j>$, $<C_iC^2>$, $<C_i\varepsilon_{rot}>$, and $<C_i\varepsilon_{vib}>$ and have been commented in the code with ***.

```
/*********************************************************************
 *                    MONACO Version 3.0                    *
 *                                                          *
 * Copyright (c) 1999-2004 University of Michigan                *
 *                                                          *
 * count.c : Sample particle properties in a cell, sorted by species*
 *          Density is always sampled for collision selection.     *
 *          Other properties only when macro props are desired.    *
 *                                                          *
 *********************************************************************/

#include <string.h>

#include "../KERN/constants.h"
#include "particle.h"
#include "../KERN/global.h"
#include "../KERN/misc.h"
#include "cellphys.h"

#define BLOCKSIZE 1000

void count(int nobj,
          particle_type particles[MAXNOBJ],
          int sample,
          float sums[MAXNSPEC][MAXNSUMS],
          int nobjspec[MAXNSPEC])
{
  int iobj,istrips,nup,nlow,ispec,n;
  int iobjX[MAXNSPEC][BLOCKSIZE],nspecblock[MAXNSPEC];
  float obju,objv,objw,objrot,objvib;
  float usum,uusum,vsum,vvsum,wsum,wwsum,rotsum,vibsum;
/*********************************************************/
  /* Additions by Ryan Carr to calculate shear and heating */
  float
uvsum,uwsum,vwsum,uuusum,uvvsum,uwwsum,vuusum,vvvsum,vwwsum,urotsum,uvibsum,vrotsum,vvib
sum;
  /*********************************************************/
```

```
/* Reset number of objects per species */

  memset(nobjspec, 0, MAXNSPEC*sizeof(int));

  if (!sample)
  {
    /* If no sampling necessary just count number of objects per species */

    for (iobj = 0; iobj < nobj; ++iobj)
      ++nobjspec[particles[iobj].spec];
  }
  else
  {
    /* Sample properties for each species */

    nup = 0;

    /* Process in stripes for better data locality */

    for (istrips = 0; istrips < (nobj-1+BLOCKSIZE)/BLOCKSIZE; ++istrips)
    {
      nlow = nup;
      nup  = MIN(nlow+BLOCKSIZE,nobj);

      /* Reset blockcounter for spec */

      memset(nspecblock, 0, MAXNSPEC*sizeof(int));

      /* Calculate species number of objects and sort into groups */

      for (iobj = nlow; iobj < nup; ++iobj)
      {
        ispec = particles[iobj].spec;
        iobjX[ispec][nspecblock[ispec]] = iobj;
        ++nspecblock[ispec];
      }

      /* Calculate the sums for the number of particles */

      for (ispec = 0; ispec < MAXNSPEC; ++ispec)
        nobjspec[ispec] += nspecblock[ispec];

      /* Calculate the momentum of the distribution functions */

      for (ispec = 0; ispec < MAXNSPEC; ++ispec)
      {
        usum   = 0.0;
        uusum  = 0.0;
        vsum   = 0.0;
        vvsum  = 0.0;
        wsum   = 0.0;
        wwsum  = 0.0;
        rotsum = 0.0;
        vibsum = 0.0;
```

```
/*************************************************************/
          /* Additions by Ryan Carr to calculate shear and heating */
          uvsum  = 0.0;
          uwsum  = 0.0;
          vwsum  = 0.0;
          uuusum = 0.0;
          uvvsum = 0.0;
          uwwsum = 0.0;
          vuusum = 0.0;
          vvvsum = 0.0;
          vwwsum = 0.0;
          urotsum = 0.0;
          uvibsum = 0.0;
          vrotsum = 0.0;
          vvibsum = 0.0;
          /*********************************************************/

      for (n = 0; n < nspecblock[ispec]; ++n)
      {
       iobj = iobjX[ispec][n];

       obju    = particles[iobj].Vx;
       usum   += obju;
       uusum  += obju*obju;

       objv    = particles[iobj].Vy;
       vsum   += objv;
       vvsum  += objv*objv;

       objw    = particles[iobj].Vz;
       wsum   += objw;
       wwsum  += objw*objw;

       objrot  = particles[iobj].Erot;
       rotsum += objrot;

       objvib  = particles[iobj].Evib;
       vibsum += objvib;

/*************************************************************/
          /* Additions by Ryan Carr to calculate shear and heating */
          uvsum += obju*objv;
          uwsum += obju*objw;
          vwsum += objv*objw;
          uuusum += obju*obju*obju;
          uvvsum += obju*objv*objv;
          uwwsum += obju*objw*objw;
          vuusum += objv*obju*obju;
          vvvsum += objv*objv*objv;
          vwwsum += objv*objw*objw;
          urotsum += obju*objrot;
          uvibsum += obju*objvib;
          vrotsum += objv*objrot;
```

```
            vvibsum += objv*objvib;
            /* note that I have not put in wuusum and wvvsum, this */
            /* must be done before the code can be used in 3D */
            /*********************************************************/
        }

        sums[ispec][SUM_N]   += nspecblock[ispec];
        sums[ispec][SUM_U]   += usum;
        sums[ispec][SUM_UU]  += uusum;
        sums[ispec][SUM_V]   += vsum;
        sums[ispec][SUM_VV]  += vvsum;
        sums[ispec][SUM_W]   += wsum;
        sums[ispec][SUM_WW]  += wwsum;
        sums[ispec][SUM_ROT] += rotsum;
        sums[ispec][SUM_VIB] += vibsum;

/*********************************************************/
            /* Additions by Ryan Carr to calculate shear and heating */
            sums[ispec][SUM_UV]  += uvsum;
            sums[ispec][SUM_UW]  += uwsum;
            sums[ispec][SUM_VW]  += vwsum;
        sums[ispec][SUM_UUU]  += uuusum;
            sums[ispec][SUM_UVV]  += uvvsum;
            sums[ispec][SUM_UWW]  += uwwsum;
            sums[ispec][SUM_VUU]  += vuusum;
            sums[ispec][SUM_VVV]  += vvvsum;
            sums[ispec][SUM_VWW]  += vwwsum;
            sums[ispec][SUM_UROT]  += urotsum;
            sums[ispec][SUM_UVIB]  += uvibsum;
            sums[ispec][SUM_VROT]  += vrotsum;
            sums[ispec][SUM_VVIB]  += vvibsum;
            /*********************************************************/
      }
    }
  }
}
```

End of file count.c

File: src/PHYS/cellphys.h

This file is used to define the *sums* structure which is part of the cell structure.

```
/*********************************************************************
 *                    MONACO Version 3.0                             *
 *                                                                   *
 * Copyright (c) 1999-2004 University of Michigan                    *
 *                                                                   *
 * cellphys.h : Definition of subset of cell structure which is      *
 *              physics related, including all sampled data.         *
 *                                                                   *
 *********************************************************************/

#ifndef CELLPHYS_H
#define CELLPHYS_H
```

```
enum { SUM_N = 0,
      SUM_U,
      SUM_UU,
      SUM_V,
      SUM_VV,
      SUM_W,
      SUM_WW,
      SUM_ROT,
      SUM_VIB,
/*************************************************/
      /* Additions by Ryan Carr to calculate shear and heating */
      SUM_UV,
      SUM_UW,
      SUM_VW,
      SUM_UUU,
      SUM_UVV,
      SUM_UWW,
      SUM_VUU,
      SUM_VVV,
      SUM_VWW,
      SUM_UROT,
      SUM_UVIB,
      SUM_VROT,
      SUM_VVIB,
      /*******************************************************/
      MAXNSUMS
};

typedef struct cell_phys
{
  float sums[MAXNSPEC][MAXNSUMS];     /* Sample values for cell */

} cell_phys;

#endif /* CELLPHYS_H */
```

End of file cellphys.h

File: src/OXFD/getvars.c

This file is used to read in the post-process variables requested by the user. The entire file is too long to include here. This segment is simply added to the *if* statement inside the file.

```
…   else if ( !strcasecmp(var,"MACH") )
    {
      if (normval[nvars]==1.0)
            sprintf(var_list[nvars],"%s", "Ma");
      else
            sprintf(var_list[nvars],"%s", "Ma/Ma0"); /* this would be strange */

      if (species[nvars]>0)
            strcat(var_list[nvars], specnum);

      eval_list[nvars] = eval_mach;
```

```
    nvars++;
   }
/*****************************************/
/* Changes added by Ryan Carr */

   else if ( !strcasecmp(var,"Q1") )
   {
    strcpy(var_list[nvars],"q1");

    if (species[nvars]>0)
     strcat(var_list[nvars], specnum);

    eval_list[nvars] = eval_q1;

    nvars++;
   }

   else if ( !strcasecmp(var,"Q2") )
    {

          strcpy(var_list[nvars],"q2");

          if (species[nvars]>0)
            strcat(var_list[nvars], specnum);

          eval_list[nvars] = eval_q2;

          nvars++;
    }

     else if ( !strcasecmp(var,"TAU11") )
    {
          strcpy(var_list[nvars],"tau11");

          if (species[nvars]>0)
      strcat(var_list[nvars], specnum);

    eval_list[nvars] = eval_tau11;

    nvars++;
   }

   else if ( !strcasecmp(var,"TAU12") )
    {
          strcpy(var_list[nvars],"tau12");

          if (species[nvars]>0)
      strcat(var_list[nvars], specnum);

    eval_list[nvars] = eval_tau12;

    nvars++;
   }
```

```
 else if ( !strcasecmp(var,"TAU22") )
   {
         strcpy(var_list[nvars],"tau22");

         if (species[nvars]>0)
    strcat(var_list[nvars], specnum);

   eval_list[nvars] = eval_tau22;

   nvars++;
 }

else if ( !strcasecmp(var,"QROT1") )
 {
  strcpy(var_list[nvars],"qrot1");

  if (species[nvars]>0)
    strcat(var_list[nvars], specnum);

  eval_list[nvars] = eval_qrot1;

  nvars++;
 }

 else if ( !strcasecmp(var,"QROT2") )
   {

         strcpy(var_list[nvars],"qrot2");

         if (species[nvars]>0)
           strcat(var_list[nvars], specnum);

         eval_list[nvars] = eval_qrot2;

         nvars++;
   }

else if ( !strcasecmp(var,"QVIB1") )
 {
  strcpy(var_list[nvars],"qvib1");

  if (species[nvars]>0)
    strcat(var_list[nvars], specnum);

  eval_list[nvars] = eval_qvib1;

  nvars++;
 }

 else if ( !strcasecmp(var,"QVIB2") )
   {

         strcpy(var_list[nvars],"qvib2");
```

```
        if (species[nvars]>0)
          strcat(var_list[nvars], specnum);

        eval_list[nvars] = eval_qvib2;

        nvars++;
    }
/*********************************/
```

End of changes to getvars.c

File: src/OXFD/oxford.h

Functions are defined in this file. The functions used to calculate the shear and heating components are added to the list.

```
/* added by Ryan Carr to calculate heat and shear */
extern float eval_q1(cell_type *, int);
extern float eval_q2(cell_type *, int);
extern float eval_tau11(cell_type *, int);
extern float eval_tau12(cell_type *, int);
extern float eval_tau22(cell_type *, int);
extern float eval_qrot1(cell_type *, int);
extern float eval_qrot2(cell_type *, int);
extern float eval_qvib1(cell_type *, int);
extern float eval_qvib2(cell_type *, int);
/************************************************/
```

End of changes to oxford.h

File: src/OXFD/eval.c

This file is a group of functions used to calculate post-process variables. Only the new functions to calculate shear and heating have been included here. They can simply be added to the overall list of functions.

```
/*********************************************************************/
/* Return Q1                         */
/*********************************************************************/
float eval_q1(cell_type *cell, int spec)
{
 int ispec;

 float vx,vy,vz,vxx,vyy,vzz,vxy,vxz,vyz,vxxx,vxyy,vxzz,vyxx,vyyy,vyzz;
 float q1 = 0.0,c1c1c1=0,c1c2c2=0,c1c3c3=0;

 /* Mole fractions of species in this cell */
 float *nfrac = molefrac[cell->cellid];

 if (spec==0)    /* All species */
 {
  for (ispec=0; ispec<nspec; ispec++)
  {
   vx = cell->phys.sums[ispec][1];
   vy = cell->phys.sums[ispec][3];
   vz = cell->phys.sums[ispec][5];
```

```
    vxx = cell->phys.sums[ispec][2];
    vyy = cell->phys.sums[ispec][4];
    vzz = cell->phys.sums[ispec][6];

    vxy = cell->phys.sums[ispec][9];
    vxz = cell->phys.sums[ispec][10];
    vyz =  cell->phys.sums[ispec][11];
    vxxx = cell->phys.sums[ispec][12];
    vxyy = cell->phys.sums[ispec][13];
    vxzz = cell->phys.sums[ispec][14];
    vyxx = cell->phys.sums[ispec][15];
    vyyy = cell->phys.sums[ispec][16];
    vyzz = cell->phys.sums[ispec][17];

    c1c1c1 += (vxxx-3.0*vx*vxx+2*vx*vx*vx)*nfrac[ispec];
    c1c2c2 += (vxyy-2.0*vy*vxy+2*vx*vy*vy-vx*vyy)*nfrac[ispec];
    c1c3c3 += (vxzz-2.0*vz*vxz+2*vx*vz*vz-vx*vzz)*nfrac[ispec];
  }
 }
 else
 {

    vx = cell->phys.sums[spec-1][1];
    vy = cell->phys.sums[spec-1][3];
    vz = cell->phys.sums[spec-1][5];

    vxx = cell->phys.sums[spec-1][2];
    vyy = cell->phys.sums[spec-1][4];
    vzz = cell->phys.sums[spec-1][6];

    vxy = cell->phys.sums[spec-1][9];
    vxz = cell->phys.sums[spec-1][10];
    vyz =  cell->phys.sums[spec-1][11];
    vxxx = cell->phys.sums[spec-1][12];
    vxyy = cell->phys.sums[spec-1][13];
    vxzz = cell->phys.sums[spec-1][14];
    vyxx = cell->phys.sums[spec-1][15];
    vyyy = cell->phys.sums[spec-1][16];
    vyzz = cell->phys.sums[spec-1][17];

    c1c1c1 = vxxx-3.0*vx*vxx+2*vx*vx*vx;
    c1c2c2 = vxyy-2.0*vy*vxy+2*vx*vy*vy-vx*vyy;
    c1c3c3 = vxzz-2.0*vz*vxz+2*vx*vz*vz-vx*vzz;

 }

 q1 = 0.5*eval_mdens(cell,spec)*(c1c1c1+c1c2c2+c1c3c3);

 return (q1);
}
/*****************************************************************/
/* Return Q2                                                     */
/*****************************************************************/
float eval_q2(cell_type *cell, int spec)
```

```
{
 int ispec;

 float vx,vy,vz,vxx,vyy,vzz,vxy,vxz,vyz,vxxx,vxyy,vxzz,vyxx,vyyy,vyzz;
 float q2 = 0.0,c2c2c2=0,c2c1c1=0,c2c3c3=0;

 /* Mole fractions of species in this cell */
 float *nfrac = molefrac[cell->cellid];

 if (spec==0)    /* All species */
   {
    for (ispec=0; ispec<nspec; ispec++)
          {
           vx = cell->phys.sums[ispec][1];
           vy = cell->phys.sums[ispec][3];
           vz = cell->phys.sums[ispec][5];

           vxx = cell->phys.sums[ispec][2];
           vyy = cell->phys.sums[ispec][4];
           vzz = cell->phys.sums[ispec][6];

           vxy = cell->phys.sums[ispec][9];
           vxz = cell->phys.sums[ispec][10];
           vyz =  cell->phys.sums[ispec][11];
           vxxx = cell->phys.sums[ispec][12];
           vxyy = cell->phys.sums[ispec][13];
           vxzz = cell->phys.sums[ispec][14];
           vyxx = cell->phys.sums[ispec][15];
           vyyy = cell->phys.sums[ispec][16];
           vyzz = cell->phys.sums[ispec][17];

           c2c2c2 += (vyyy-3.0*vy*vyy+2*vy*vy*vy)*nfrac[ispec];
           c2c1c1 += (vyxx-2.0*vx*vxy-vy*vxx+2*vy*vx*vx)*nfrac[ispec];
           c2c3c3 += (vyzz-2.0*vz*vyz-vy*vzz+2*vy*vz*vz)*nfrac[ispec];
          }
   }
 else
   {

    vx = cell->phys.sums[spec-1][1];
    vy = cell->phys.sums[spec-1][3];
    vz = cell->phys.sums[spec-1][5];

    vxx = cell->phys.sums[spec-1][2];
    vyy = cell->phys.sums[spec-1][4];
    vzz = cell->phys.sums[spec-1][6];

    vxy = cell->phys.sums[spec-1][9];
    vxz = cell->phys.sums[spec-1][10];
    vyz =  cell->phys.sums[spec-1][11];
    vxxx = cell->phys.sums[spec-1][12];
    vxyy = cell->phys.sums[spec-1][13];
    vxzz = cell->phys.sums[spec-1][14];
    vyxx = cell->phys.sums[spec-1][15];
```

```
    vyyy = cell->phys.sums[spec-1][16];
    vyzz = cell->phys.sums[spec-1][17];

    c2c2c2 = vyyy-3.0*vy*vyy+2*vy*vy*vy;
    c2c1c1 = vyxx-2.0*vx*vxy-vy*vxx+2*vy*vx*vx;
    c2c3c3 = vyzz-2.0*vz*vyz-vy*vzz+2*vy*vz*vz;
   }

  q2 = 0.5*eval_mdens(cell,spec)*(c2c2c2+c2c1c1+c2c3c3);

  return (q2);
}

/*********************************************************************/
/* Return tau11                                                      */
/*********************************************************************/
float eval_tau11(cell_type *cell, int spec)
{
  int ispec;

  float vx,vxx;
  float tau11,c1c1=0.0;
  /* Mole fractions of species in this cell */
  float *nfrac = molefrac[cell->cellid];

  if (spec==0)    /* All species */
    {
     for (ispec=0; ispec<nspec; ispec++)
            {
              vx = cell->phys.sums[ispec][1];
              vxx = cell->phys.sums[ispec][2];

              c1c1 += (vxx-vx*vx)*nfrac[ispec];
            }
    }
  else
    {
     vx = cell->phys.sums[spec-1][1];
     vxx = cell->phys.sums[spec-1][2];

     c1c1 = (vxx-vx*vx);
    }

  tau11 = -c1c1*eval_mdens(cell,spec)+eval_press(cell,spec);

  return (tau11);
}


/*********************************************************************/
/* Return tau12                                                      */
/*********************************************************************/
float eval_tau12(cell_type *cell, int spec)
{
```

```
  int ispec;

  float vx,vy,vxy;
  float tau12,c1c2=0.0;
  /* Mole fractions of species in this cell */
  float *nfrac = molefrac[cell->cellid];

  if (spec==0)    /* All species */
    {
     for (ispec=0; ispec<nspec; ispec++)
            {
              vx = cell->phys.sums[ispec][1];
              vy = cell->phys.sums[ispec][3];
              vxy = cell->phys.sums[ispec][9];

              c1c2 += (vxy-vx*vy)*nfrac[ispec];
            }
    }
  else
    {
     vx = cell->phys.sums[spec-1][1];
     vy = cell->phys.sums[spec-1][3];
     vxy = cell->phys.sums[spec-1][9];

     c1c2 = (vxy - vx*vy);
    }

  tau12 = -c1c2*eval_mdens(cell,spec);

  return (tau12);
}

/*********************************************************************/
/* Return tau22                                                      */
/*********************************************************************/
float eval_tau22(cell_type *cell, int spec)
{
  int ispec;

  float vy,vyy;
  float tau22,c2c2=0.0;
  /* Mole fractions of species in this cell */
  float *nfrac = molefrac[cell->cellid];

  if (spec==0)    /* All species */
    {

     for (ispec=0; ispec<nspec; ispec++)
            {
              vy = cell->phys.sums[ispec][3];
              vyy = cell->phys.sums[ispec][4];

              c2c2 += (vyy-vy*vy)*nfrac[ispec];
            }
```

```
  }
 else
  {
   vy = cell->phys.sums[spec-1][3];
   vyy = cell->phys.sums[spec-1][4];

   c2c2 = (vyy-vy*vy);
  }

 tau22 = -c2c2*eval_mdens(cell,spec)+eval_press(cell,spec);

 return (tau22);
}

/*********************************************************************/
/* Return Qrot1                          */
/*********************************************************************/
float eval_qrot1(cell_type *cell, int spec)
{
 int ispec;

 float vx;
 float qrot1,erotc1=0.0,erot;
 /* Mole fractions of species in this cell */
 float *nfrac = molefrac[cell->cellid];

 if (spec==0)    /* All species */
  {
   for (ispec=0; ispec<nspec; ispec++)
        {
         vx = cell->phys.sums[ispec][1];
         erotc1 = cell->phys.sums[ispec][18];
         erot = cell->phys.sums[ispec][7];
         qrot1 = (erotc1-vx*erot)*nfrac[ispec];
        }
  }
 else
  {
         vx = cell->phys.sums[spec-1][1];
         erotc1 = cell->phys.sums[spec-1][18];
         erot = cell->phys.sums[spec-1][7];

         qrot1 = (erotc1-vx*erot);
  }

 qrot1 = qrot1*eval_mdens(cell,spec)/species[spec].mass;

 return (qrot1);
}

/*********************************************************************/
/* Return Qrot2                          */
/*********************************************************************/
float eval_qrot2(cell_type *cell, int spec)
```

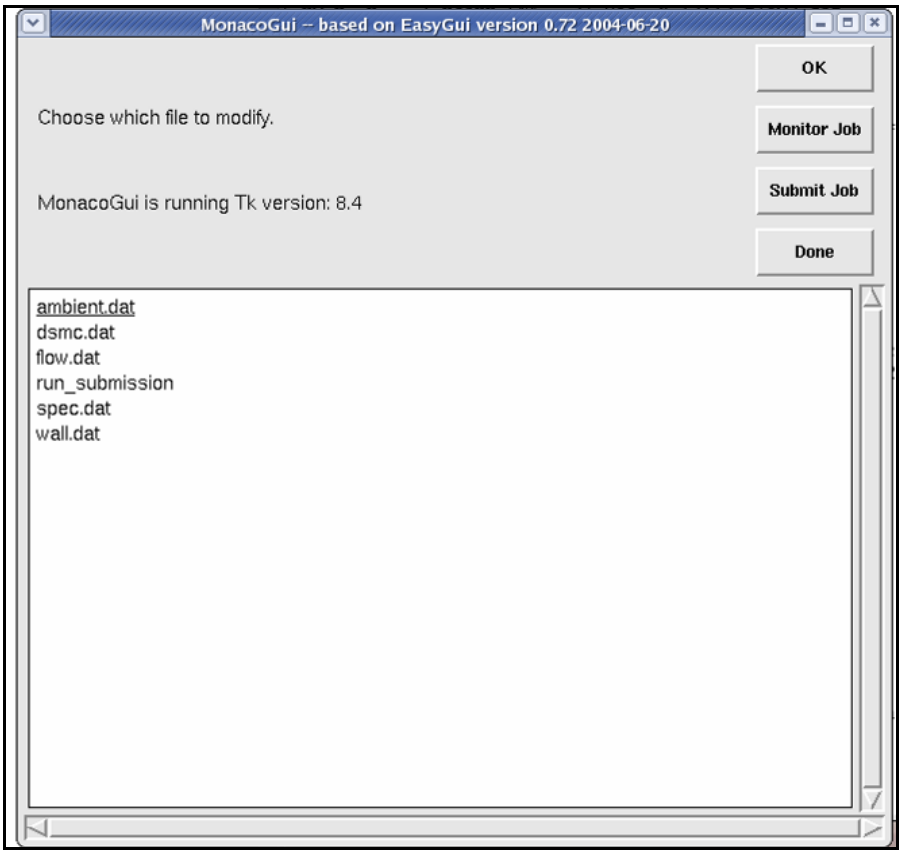
```
{
 int ispec;

 float vy;
 float qrot2,erotc2=0.0,erot;
 /* Mole fractions of species in this cell */
 float *nfrac = molefrac[cell->cellid];

 if (spec==0)    /* All species */
   {
    for (ispec=0; ispec<nspec; ispec++)
           {
            vy = cell->phys.sums[ispec][3];
            erotc2 = cell->phys.sums[ispec][20];
            erot = cell->phys.sums[ispec][7];

            qrot2 = (erotc2-vy*erot)*nfrac[ispec];
           }
   }
 else
   {
            vy = cell->phys.sums[ispec][3];
            erotc2 = cell->phys.sums[ispec][20];
            erot = cell->phys.sums[ispec][7];

            qrot2 = (erotc2-vy*erot);
   }

 qrot2 = qrot2*eval_mdens(cell,spec)/species[spec].mass;

 return (qrot2);
}


/*****************************************************************/
/* Return Qvib1                                                  */
/*****************************************************************/
float eval_qvib1(cell_type *cell, int spec)
{
 int ispec;

 float vx;
 float qvib1,evibc1=0.0,evib;
 /* Mole fractions of species in this cell */
 float *nfrac = molefrac[cell->cellid];

 if (spec==0)    /* All species */
   {
    for (ispec=0; ispec<nspec; ispec++)
           {
            vx = cell->phys.sums[ispec][1];
            evibc1 = cell->phys.sums[ispec][19];
            evib = cell->phys.sums[ispec][8];
```

```
            qvib1 = (evibc1-vx*evib)*nfrac[ispec];
           }
    }
  else
    {
            vx = cell->phys.sums[ispec][1];
            evibc1 = cell->phys.sums[ispec][19];
            evib = cell->phys.sums[ispec][8];

            qvib1 = (evibc1-vx*evib);
    }

  qvib1 = qvib1*eval_mdens(cell,spec)/species[spec].mass;

  return (qvib1);
}

/*********************************************************************/
/* Return Qvib2                          */
/*********************************************************************/
float eval_qvib2(cell_type *cell, int spec)
{
  int ispec;

  float vy;
  float qvib2,evibc2=0.0,evib;
  /* Mole fractions of species in this cell */
  float *nfrac = molefrac[cell->cellid];

  if (spec==0)    /* All species */
    {
     for (ispec=0; ispec<nspec; ispec++)
           {
            vy = cell->phys.sums[ispec][3];
            evibc2 = cell->phys.sums[ispec][21];
            evib = cell->phys.sums[ispec][8];

            qvib2 = (evibc2-vy*evib)*nfrac[ispec];
           }
    }
  else
    {
            vy = cell->phys.sums[ispec][3];
            evibc2 = cell->phys.sums[ispec][21];
            evib = cell->phys.sums[ispec][8];

            qvib2 = (evibc2-vy*evib);
    }

  qvib2 = qvib2*eval_mdens(cell,spec)/species[spec].mass;

  return (qvib2);
}
```

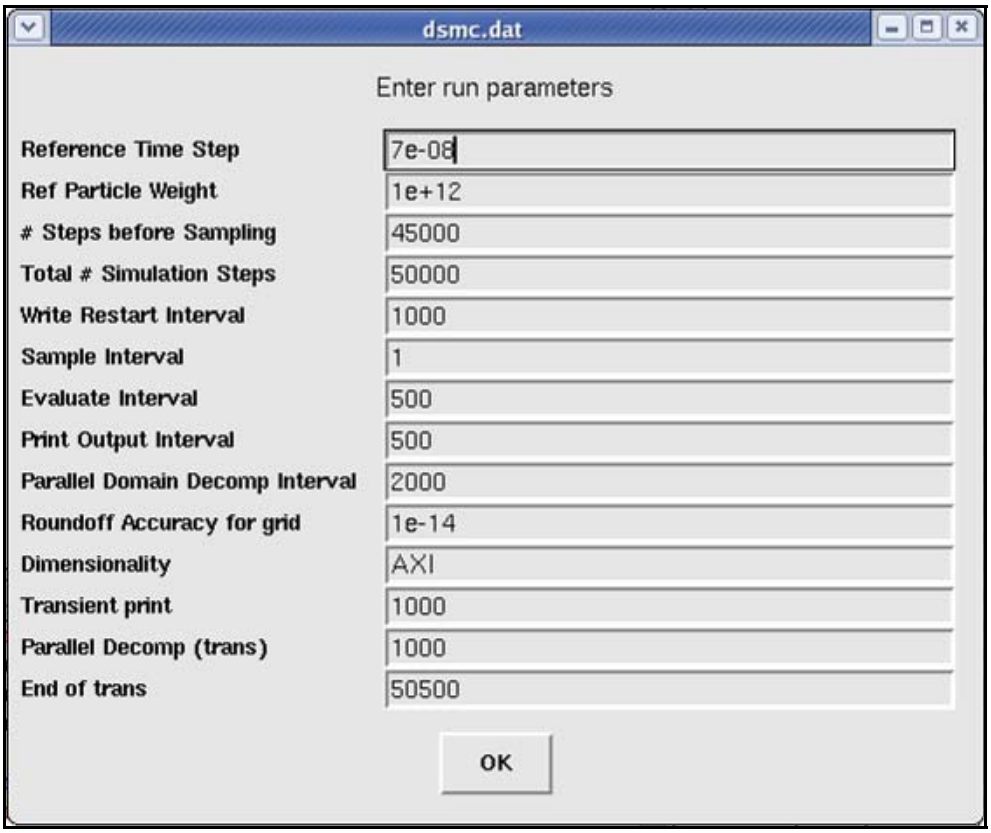
The MONACO Graphical Interface: MonacoGui

The MONACO program requires the user to create many input files containing run, species, collision model, flow, and wall parameters. Because there are so many parameters and input files it is easy for the user to make mistakes when creating and submitting a new run. For this reason, a graphical interface to MONACO was created by the author. The graphical user interface (GUI) was programmed in Python (2006), an open source license code widely used with Linux systems.

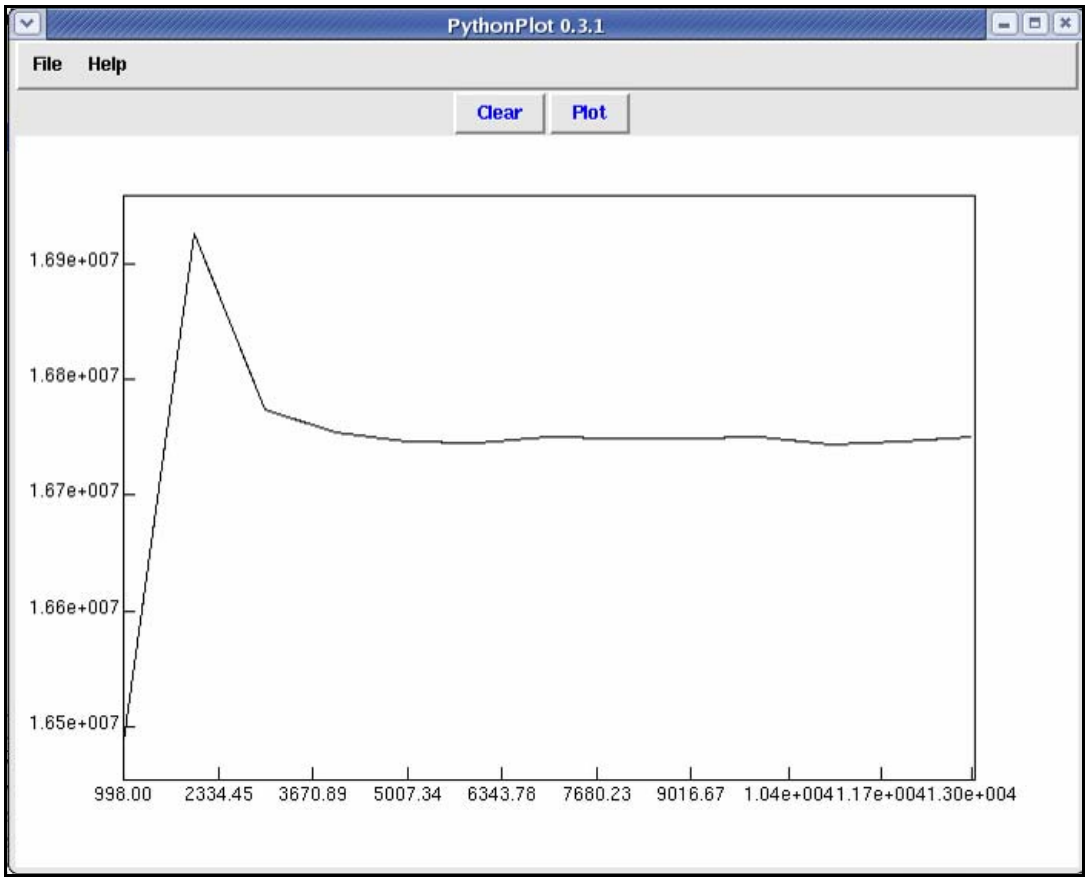
The base code was adapted from EasyGui, an open source program by Stephen Ferg (2004) available for download on the internet. Changes were made to enable MonacoGui to set up a run with all associated inputs, submit it to the queue, and monitor its progress. The picture below shows the opening screen.



The user is able to modify the input files by clicking on one of the files listed in the window. For example, to modifying the run parameter input deck *dsmc.dat* would take the user to the following screen:



Once the run is set up, by clicking the "Submit Job" button on the main window the user can submit a parallel job to the clusters. While the job is running, it may be monitored by using the "Monitor Job" button. Many different parameters may be monitored. Below is an example of the particle history for a standard run.



Code used to create MonacoGui is included below. This is by no means the entire code; however, it contains most of the modifications made by the author to the EasyGui program.

```
#-----------------------------------------------------
# routines defined by Ryan Carr
#-----------------------------------------------------
def dsmcdat():
        msg = "Enter run parameters"
        title = "dsmc.dat"
        fieldNames = ["Reference Time Step","Ref
Particle Weight","# Steps befo$
        fieldValues = []  # we start with blanks for
the values
        defaults=[0,0,0,0,0,0,0,0,0,0,0,0,0,0]


        f=open('dsmc.dat', 'r')
```

```
l = f.readline()
defaults[0] = float(l.strip())
l = f.readline()
defaults[1] = float(l.strip())
l = f.readline()
defaults[2] = int(l.strip())
l = f.readline()
defaults[3] = int(l.strip())
l = f.readline()
defaults[4] = int(l.strip())
l = f.readline()
defaults[5] = int(l.strip())
l = f.readline()
defaults[6] = int(l.strip())
      l = f.readline()
```

```
            defaults[7] = int(l.strip())
        l = f.readline()
        defaults[8] = int(l.strip())
        l = f.readline()
        defaults[9] = float(l.strip())
        l = f.readline()
        defaults[10] = l.strip('\n ')
        l = f.readline()
        defaults[11] = int(l.strip())
        l = f.readline()
        defaults[12] = int(l.strip())
        l = f.readline()
        defaults[13] = int(l.strip())

        f.close()

        fieldValues =
multenterbox(msg,title,fieldNames,defaults)
        f=open('dsmc.dat', 'w')
        for x in range(len(fieldValues)):
            f.write(fieldValues[x])
            f.write("\n")
        f.close()

def flowdat():
        msg = "Enter Flow Data"
        title = "field.dat"
        fieldNames = ["X-velocity (m/s)","Y-
velocity","Z-velocity","Temperature$
        fieldValues = []  # we start with blanks for
the values
        defaults=[0,0,0,0,0]
        f=open('flow.dat', 'r')
        l = f.readline()
        defaults = l.split()
### Assumes only one flow boundary condition!
        f.close()
fieldValues =
multenterbox(msg,title,fieldNames,defaults)

        l = "  ".join(fieldValues)

        f=open('flow.dat', 'w')
        f.write(l)
        f.close()
def specdat():
        msg = "Enter Species Data"
        title = "spec.dat"
        fieldNames = ["VHS Reference
Temperature (K)","VHS omega","Species
Name$
        fieldValues = []  # we start with blanks for
the values
        defaults=[0,0,0,0,0,0,0,0,0,0,0]
        f=open('spec.dat', 'r')

        T = f.readline()
        w = f.readline()
        l = f.readline()

        defaults[0] = T.strip()
        defaults[1] = w.strip()
        defaults[2:10]=l.split()

        f.close()

        fieldValues =
multenterbox(msg,title,fieldNames,defaults)

        l = "  ".join(fieldValues)

        f=open('spec.dat', 'w')
        f.write(T)
        f.write(w)
        f.write(l)
        f.close()
def walldat():
        msg = "Enter Wall Parameters"
        title = "field.dat"
        fieldNames = ["Wall Temperature
(K)","Wall Accomodation Coefficient"]
        fieldValues = []  # we start with blanks for
the values
        defaults=[0,0]
        f=open('wall.dat', 'r')
        l = f.readline()
        defaults = l.split()
        f.close()
        fieldValues =
multenterbox(msg,title,fieldNames,defaults)
        l = "  ".join(fieldValues)
        f=open('wall.dat', 'w')
        f.write(l)
        f.close()

def ambientdat():
        msg = "Enter Ambient Data"
        title = "ambient.dat"
        fieldNames = ["X-velocity (m/s)","Y-
velocity","Z-velocity","Temperature$
        fieldValues = []  # we start with blanks for
the values
        defaults=[0,0,0,0,0]
        f=open('ambient.dat', 'r')
        l = f.readline()
        defaults = l.split()
### Assumes only one flow boundary condition!
```

```
        f.close()
        fieldValues =
multenterbox(msg,title,fieldNames,defaults)
        l = "  ".join(fieldValues)
        f=open('ambient.dat', 'w')
        f.write(l)
        f.close()

def tahoesh():
        msg = "Enter Submission Data"
        title = "Job Submission Input File -
tahoe.sh"
        fieldNames = ["Job Name","# of
Nodes","Processors per Node (2)","'monac$
        fieldValues = []  # we start with blanks for
the values
        defaults=[0,0,0,0]


        f=open('tahoe.sh', 'r')

        l = f.readlines()

        defaults[0] = l[3].strip('#PBS -N\n')
        defaults[1:2] = l[7].strip('#PBS -l \n
nodes=').split(':ppn=')
        defaults[3] = l[12].strip('MONACO_DIR=
\n')

        f.close()

        fieldValues =
multenterbox(msg,title,fieldNames,defaults)

        l[3]=" ".join(['#PBS -N',fieldValues[0],'\n'])
        l[7]="".join(['#PBS -l
nodes=',fieldValues[1],':ppn=',fieldValues[2],'\$

l[13]="".join(['MONACO_DIR=',fieldValues[3],
'\n'])

        f=open('tahoe.sh', 'w')

        for x in range(len(l)):
                f.write(l[x])
        f.close()

def monitor(chooser):


        if chooser == '# of Particles':
                f = open('monaco.dat','r')
                content = f.readlines()
                f.close()
                time=range(len(content))
                part=range(len(content))

                for x in range(len(content)):
                        time[x] =
int(content[x].split('\t')[0])
                        part[x] =
int(content[x].split('\t')[1])
                master = Tk()
                plot = PythonPlot(master)
                plot.plotData(time, part)


        elif chooser == '# of Collisions':
                f = open('monaco.dat','r')
                content = f.readlines()
                f.close()
                time=range(len(content))
                coll=range(len(content))
                for x in range(len(content)):
                        time[x] =
int(content[x].split('\t')[0])
                        coll[x] =
int(content[x].split('\t')[2])

                master = Tk()
                plot = PythonPlot(master)
                plot.plotData(time, coll)

                master.mainloop()

                master.mainloop()

        ['Time Step','# of Particles','# of
Collisions','Particles per Processo$

#-----------------------------------------------------
----------
# monacogui driver code
#-----------------------------------------------------
----------
def _monaco():
        #============================end
of text
==================================
        intro_message = ("Choose which file to
modify.\n\n"
         + "" +
        "" +
        ""
        + "\n\nMonacoGui is running Tk version: "
+ str(TkVersion)
        )
```

```
#=================================
======== END DEMONSTRATION DATA
        while 1: # do forever
                choices = [
                        "dsmc.dat",
                        "flow.dat",
                        "spec.dat",
                        "wall.dat",
                        "ambient.dat",
                        "run_submission",
                        ]
                choice = choicebox(intro_message,
"MonacoGui " + MonacoGui, cho$
                if choice == None: return
                reply = choice.split()
                if   reply[0] == "dsmc.dat":
                        reply = dsmcdat()
                elif reply[0] == "flow.dat":
                        reply = flowdat()
                elif reply[0] == "spec.dat":
                        reply = specdat()
                elif reply[0] == "wall.dat":
                        reply = walldat()
                elif reply[0] == "ambient.dat":
                        reply = ambientdat()
                elif reply[0] == "run_submission":
                        reply = tahoesh()
                else:
                        msgbox("Choice\n\n" + choice +
"\n\nis not recognized",$
                        return
if __name__ == '__main__':
        _monaco()
```

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From – To)*
22 Mar 07	**Master's Thesis**	September 2005 – March 2007

4. TITLE AND SUBTITLE

Quantifying Non-Equilibrium in Hypersonic Flows Using Entropy Generation

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Carr, Ryan, W., 2nd Lieutenant, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GAE/ENY/07-M07

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AFRL/VASD
Jose Camberos
2210 Eighth St
WPAFB OH 45433

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The constitutive relations traditionally used for finding shear stress and heat flux in a fluid become invalid in non-equilibrium flow. Their derivation from kinetic theory only demonstrates they are valid only for small deviations from equilibrium. Because it is fundamentally linked to non-equilibrium, entropy generation is used to investigate the limits of the continuum constitutive relations. However, the continuum equations are inherently limited to near equilibrium conditions due to the constitutive relations; thus kinetic theory must be used as a basis for comparison. Direct Simulation Monte Carlo (DSMC), a particle method alternative to continuum methods, is based on kinetic theory and is used to develop a flow solution free from equilibrium assumptions.
Solutions were obtained for hypersonic flow over two axisymmetric geometries using both a continuum solver and DSMC. Formulations for entropy generation are presented for each method, and the two solutions are compared. The continuum solver fails to capture regions of non-equilibrium as evidenced by thicker shocks in the DSMC solution.
To extend the useful range of the continuum constitutive relations, the Lennard-Jones model is offered as an alternative to Sutherland's Law for calculating viscosity and thermal conductivity. The two are compared, and parameters offering a good fit for these flows are suggested for the Lennard-Jones model.

15. SUBJECT TERMS
Hypersonics, entropy generation, Direct Simulation Monte Carlo (DSMC), Computational Fluid Dyanamics (CFD), constitutive relations, non-equilibrium

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
REPORT	**ABSTRACT**	**c. THIS PAGE**			Richard D. Branam
U	**U**	**U**	**UU**	127	**19b. TELEPHONE NUMBER** *(Include area code)* (937) 255-6565, ext; e-mail: richard.branam@afit.edu