DSMC Modeling of Micromechanical Devices

by

Edward S. Piekos

B.S.E.(Aerospace E.), University of Michigan (1992)

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Master of Science in Aeronautics and Astronautics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 1995

Author . . . .
Department of Aeronautics and Astronautics
August 18, 1995

Certified by . . . .
Kenneth S. Breuer
Assistant Professor
Thesis Supervisor

Accepted by . . . .
Harold Y. Wachman
Professor
Chairman, Departmental Graduate Committee

DSMC Modeling of Micromechanical Devices

by

Edward S. Piekos

Submitted to the Department of Aeronautics and Astronautics
on August 18, 1995, in partial fulfillment of the
requirements for the degree of
Master of Science in Aeronautics and Astronautics

Abstract

Application of the direct simulation Monte Carlo method (DSMC) to flows related to micro-electro-mechanical systems (MEMS) is detailed. This effort is aimed at furthering the understanding of rarefied gas behavior and providing tools to facilitate the design and optimization of micro-devices. The code written for this work employs an unstructured grid and a trajectory-tracing particle movement scheme. It was applied to problems suited for traditional DSMC application as well as those which required its scaling constraints to be relaxed. The former problems included slip-flow and transition regime micro-channels and a supersonic micro-nozzle. These were selected to verify analytical models and to demonstrate DSMC's value for flows which are not amenable to other types of solution. Excellent agreement was obtained between the code and available analytic expressions. Rarefied gas phenomena, such as velocity and temperature slip at the wall, were also observed. The latter problems included a Rayleigh flow, a spatially-developing shear layer, and a channel with a circular obstruction (Ni's bump). These were selected to investigate the implications of sizing a DSMC calculation according to the flow gradients, rather than the molecular scales, in an effort to efficiently extend the method to smaller Knudsen numbers. Two implications of this scaling were observed: first, when it reaches a point where each particle suffers several collisions every time step, the method is modeling the Euler equations, not the Navier-Stokes equations, because the particles in each cell reach an equilibrium distribution every time step; second, the particles' thermal velocity, coupled with DSMC's disregard for cellular position when selecting collision partners, causes an increased diffusion of momentum through the domain, appearing as an artificial viscosity which varies with the cell size and quickly overtakes the physical value.

Thesis Supervisor: Kenneth S. Breuer
Title: Assistant Professor

Acknowledgments

I guess I'll start at the beginning: in the Beginning, God created fluids (OK, so maybe a couple of other things were supposedly done first, but this was the important one). Any topic which can have retiring professors still scratching their heads and saying "Hmm, that's strange" serves, I think, as evidence that there is a Creator with access to a bigger picture than our feeble efforts have been able to construct. I have enjoyed working in this area and with the curious group of people it attracts and am indebted to the grace of God for somehow getting me here.

On a more local scale, I'd like to thank my advisor, Kenny Breuer, for all his guidance along the path that led to this thesis. He is responsible for turning me from a wide-eyed undergrad into a, well, whatever I am now, who can actually program in C, work in UNIX, use LaTeX(though I do still have mixed feelings about this one), and keep his research notes organized (which was a great help in preparing this document, so I thank you especially for pointing me in this particular direction). I also thank you for putting me on this project and actually paying me to ask "why?" and "how?", two of my favorite pastimes in the world.

I must also thank my undergraduate advisor at the University of Michigan, Ken Powell, for encouraging me to pursue graduate studies. I have found research to be very rewarding and well-suited to my nature and I really can't say grad school would have occurred to me without you (don't worry, this doesn't mean I curse your name when I am up to my eyeballs in coding bugs and problem sets).

On a daily level, I'd like to thank everyone in the FDRL (including Matt and Becky, who aren't actually in the FDRL, but sit with us anyway and Errol, Aravind, and Stu who *are* actually with the FDRL but don't sit with us (confusing, huh?)) for providing a great atmosphere in which to work and putting up with all of my (numerous) idiosyncrasies. This thanks is also extended to the members of CASL/SPPL, my new place of residence (meant literally this last few weeks). I'd especially like to thank my coworker, friend, *and* roommate, D.J. Orr, for patiently listening to me babble about everything from from plants to programming and playing entertaining games like "name that grain" (and proofreading my thesis . ←missed one!

Similarly, I thank my good friend for all time (both directions), Christianne Holly, for discussing the especially bizarre, "big picture" topics with me, fearlessly sampling many of my off-the-wall culinary creations, and getting me out of the office every now and then. All my friends, in fact, scattered as they may be around the country and globe these days, deserve thanks for being the greatest (and strangest) group of people anyone could ever hope to know.

Of course, I wouldn't be the person I am today without my family, especially my mother, whose tireless efforts when my sister and I were growing up were nothing short of amazing. Though I'm not sure how the rest of the world feels about the result, I thank you for it. Mom, Dad, Laura, and now, Bruce, I love you all.

Finally, I'd like to thank the Air Force for making this work possible; both by funding it and providing a place for my parents to meet.

This work was supported by AFOSR grant number F49620-93-1-0376, monitored by Dr. J. McMichael.

"The whole of science is nothing more than a refinement of everyday thinking... He who can no longer pause to wonder and stand rapt in awe is as good as dead; his eyes are closed." - Albert Einstein

"The most interesting information comes from children, for they tell all they know and then stop." - Mark Twain

Contents

List of Figures

List of Tables

Nomenclature

Symbol	Description	Section
a	speed of sound	2.7.2
c'_m	most probable thermal velocity	2.1
$\bar{c}'$	mean thermal velocity	2.8.1
c_r	relative speed of colliding partners	1.4.3
F	tangential momentum accommodation coeff. (1=full accom.)	2.7.1
H	channel height	3.1
H_t	nozzle throat height	3.3
k	Boltzmann constant, $k = 1.380658 \times 10^{-23}$ J/K	2.2
Kn	Knudsen number	1.2
Kn_c	cell Knudsen number	4.1
Kn_o	outlet Knudsen number	3.1
L	channel length	3.1
$\mathcal{L}$	scale length of flow gradients	1.2
m	molecular mass	2.2
m_r	reduced mass of colliding pair, $m_r = m_1 m_2/(m_1 + m_2)$	2.2
Ma	Mach number	4.2.1
n	number density	1.4.3
$\tilde{n}$	reference simulation number density	4.1
N	total number of particles	1.4.2
N_c	number of particles in a cell	1.4.3
N_p	number of collision pairs to be sampled	2.1
p	pressure	2.7.2

$\mathcal{P}$	normalized pressure, $\mathcal{P} = p/p_{outlet}$	3.1
R	gas constant, $R = k/m$	4.2.1
R_f	random fraction, $0 < R_f < 1$	2.7.2
t	time	4.2.1
T	temperature	2.2
u, v, w	macroscopic (mean) velocity components	2.7.2
u', v', w'	microscopic (thermal) velocity components	2.7.2
u_n	macroscopic velocity component normal to I/O face	2.7.2
u_s	similarity speed	3.1
U_o	wall speed in Rayleigh problem	4.2.1
U_1, U_2	shear layer stream velocities	4.2.2
V_c	cell volume	4.1
x, y	spatial coordinates	3.1
γ	specific heat ratio	2.7.2
Δt	time step	4.1
Δx	typical cell dimension	4.1
η	exponent in the inverse power law model	2.2
θ	momentum thickness	4.2.2
λ	mean free path	1.2
μ	coefficient of viscosity	3.1
ν	kinematic viscosity coefficient	4.2.1
ν_p	particulate collision rate	2.8.1
ξ	similarity variable for Rayleigh flow, $\xi = \frac{y}{2\sqrt{\nu t}}$	4.2.1
σ	cross section	1.4.3
ω	exponent in the variable hard sphere model	2.2

Chapter 1

Introduction

The direct simulation Monte Carlo method (DSMC) is a particle-based numerical fluid modeling technique pioneered by G. A. Bird.[1] This particulate nature enables it to remain valid where traditional, continuum-based, computational fluid dynamics (CFD) techniques break down due to flow rarefaction. To date, most work using this method has centered on problems related to vehicles operating at high-altitudes, where continuum-based methods become inaccurate due to the low ambient density. With the advent of micro-electro-mechanical systems (MEMS), however, flows around devices with micron-scale features have also become important. Because their sharp gradients often cause continuum methods to fail, even at standard conditions, DSMC is an attractive tool for these flows as well.

This thesis explores DSMC's application to geometries related to MEMS devices. With this task in mind, the current chapter presents an introduction to MEMS, non-continuum flows, particle methods, and DSMC. The algorithm developed for this work is then outlined in Chaper 2. Non-continuum flows, the traditional realm of this method, are treated in Chapter 3 to evaluate theoretical models as well as explore regions for which no tractable models exist. Chapter 4 then examines scaling issues inherent to efficiently applying DSMC to general continuum flows, with the intention of expanding its useful range. Finally, Chapter 5 presents conclusions which may be drawn from this work and points to several features which remain to be explored in this area.

1.1 MEMS

Micro-electro-mechanical systems (MEMS), are very small (micron-scale) sensors and actuators manufactured with techniques similar to those used for integrated circuit (IC) chips. They are the subject of increasingly active research in a widening field of disciplines. Applications for these devices range from consumer products (airbag triggers, micro-mirror displays), to industrial and medical tools (microvalves, micromotors), to instrumentation (micro pressure sensors, micro shear-stress sensors). [2]

MEMS have many advantages over their macro-scale counterparts, where such counterparts even exist. First, because these devices are fabricated in a manner similar to IC chips, they are extremely inexpensive to manufacture in large quantities. Second, the technology for such production is quite mature. Very precise specification of the geometry, far beyond that possible with macro-scale fabrication techniques, is therefore routine and a high degree of control over material properties is available. Third, their small size and mass make them attractive where space is at a premium or weight is limited. Finally, their minimal inertia allows them to react very quickly, enabling the creation of actuators and sensors with frequency responses previously unthinkable for mechanical systems.

The small size of MEMS poses unique challenges in the design phase, however. While the mechanical properties of micromachined materials are reasonably well-studied, fluid effects at micron scales are not. These effects, such as film damping of resonant structures, heat transfer in mass flow sensors, and unsteady pressure fields around microvalves, for example, must be understood if the full potential of these devices is to be realized.

1.2 Non-Continuum Flows

The vast majority of computational and analytical tools for studying fluid behavior are based on the Euler or Navier-Stokes equations. An important underlying assumption of these equations is that the fluid may be treated as a continuum, rather than as a collection of discrete particles, as is done in the, more difficult, Boltzmann equation. This allows the transport terms to be calculated using macroscopic variables, such as temperature, rather than microscopic variables, such as molecular velocity distribution function, yielding an expression which is much more amenable to solution, both analytically and numerically. Unfortunately, this approximation becomes inaccurate as the scale length of the flow gradients ($\mathcal{L}$) approaches the average distance traveled by a particle between collisions (the mean free path, λ), which occurs for many MEMS-related flows. The ratio of these quantities is known as the Knudsen number ($Kn = \lambda/\mathcal{L}$) and is used to indicate the degree of flow rarefaction. The Navier-Stokes equations neglect rarefaction effects and are therefore only strictly accurate for vanishingly-small Kn.

As is the case for other non-dimensional numbers in fluid dynamics, such as the Mach and Reynolds numbers, the type of analysis appropriate for a particular flow is dictated by its Knudsen number. Consequently, the Kn domain ($0 < Kn < \infty$) is often divided into four flow regimes.[3] For $Kn < 0.01$, known as the 'continuum' regime, the Navier-Stokes equations, as commonly expressed, are applied. For $0.01 < Kn < 0.1$, known as the 'slip-flow' regime, the Navier-Stokes equations are applied with the usual no-slip wall boundary condition replaced by a slip-flow condition (detailed in Section 3.1). For $0.1 < Kn < 3$, known as the 'transition' regime, the flow is too rarefied for Navier-Stokes-based analysis, but not rarefied enough to apply the collisionless Boltzmann equation. The full Boltzmann equation is therefore prescribed. For $Kn > 3$, known as the 'free molecular' regime, the flow is sufficiently rarefied to allow molecular collisions to be neglected. The collisionless Boltzmann equation is therefore applied.

For many MEMS, Kn is driven from the continuum regime by their extremely small feature size, which is often comparable to λ, even at standard conditions. The gap between the sensing plate and the substrate on a floating element shear stress sensor, for example, is typically 1-2 microns.[4] The mean free path of air at standard conditions is approximately 60 nm. This places Kn in the slip flow regime, or even the transition regime for lighter gases or different conditions. Non-continuum effects in the gap, neglected in traditional analyses, may therefore have a significant impact on sensor operation. As a result, new models and techniques must be developed to correctly describe the behavior of the fluid in and around these devices.

1.3 Particle Methods

Particle methods, such as molecular dynamics (MD), particle-in-cell (PIC), and direct simulation Monte Carlo (DSMC) are attractive tools for the study of rarefied gas flows because they lack continuum assumptions. These techniques model gas behavior by tracking the interaction of computational particles, each with a position, a velocity, an internal energy, etc., mimicking the discrete molecular nature of the actual flow. This strategy differs considerably from that of traditional CFD, which numerically solves differential field equations formulated to describe fluid behavior in terms of macroscopic variables.

Particle methods are intuitively attractive because fully physical simulations would be devoid of assumptions and would therefore be valid for all flow regimes and geometries. Unfortunately, a fully-physical simulation, even for a simple problem, typically requires computational power several orders of magnitude greater than is currently available. In addition, it is arguable that a complete understanding of intermolecular and molecule-surface interactions is not yet reached, so some detail would necessarily be neglected if such a calculation were even attempted.

Fortunately, many features of molecular behavior have negligible influence on most engineering problems. The key to an efficient simulation is therefore to include the minimum level of complexity required to correctly reproduce the important features of a given flow. Consequently, all current particle methods make simplifications in the quantity, movement, and/or interactions of their computational particles. The form of these simplifications differentiates between the various techniques.

1.4 DSMC

DSMC is, by far, the most popular particle method for the analysis of collisional flows, ie. flows for which intermolecular collisions significantly affect fluid behavior. It is called a *simulation* (rather than a *solution*) scheme because it was originally formulated to capture the important physical features of the flow, not to solve a particular set of equations. Nevertheless, Nanbu later showed that the various techniques solve either the Kac Master equation or the Boltzmann equation.[5] As a result, some current algorithms were subsequently derived from these equations, rather than from physical arguments.

DSMC has undergone considerable development by many researchers for more than two decades. Consequently, a considerable number of variations from the original algorithm now exist. A number of features nonetheless remain common to most implementations. These defining features are associated with the simplifications made to the physical situation and differentiate DSMC from other particle methods.

1.4.1 Particle Quantity

The number of particles in the simulation must be reduced due to memory constraints as well as CPU time considerations. For example, a cubic centimeter of gas at standard conditions contains approximately 2.69×10^{19} molecules. To represent only position and velocity in 3-dimensional space for these particles with single-precision, floating point numbers would require a total memory of 6.5×10^{14} Mb, far beyond the capacity of even modern supercomputers.

The crucial assumption made by DSMC developers in the face of this problem is that, at any given time, a number of molecules are in virtually indistinguishable microscopic states. These molecules may therefore be collectively represented as a single 'computational particle', which is then a statistical, rather than a physical, entity because it represents a small volume in phase space, rather than an actual molecule.

The ratio of real to computational particles is known as the 'weight factor' and may be constant for a given computation or vary with position and/or time. Variable weight factors are useful in rapidly expanding and axisymmetric flows, where large changes in number density or cell volume occur across the domain, to hold the number of particles in each cell at a computationally efficient level.[6] These weighting schemes must be applied with caution because particles are usually 'cloned' when they move into a region with a smaller weight factor. This is problematic because cloned particles are not statistically independent. Introducing significant numbers of them will therefore cause a larger scatter to be observed in the results than is expected for the given number of particles. In addition, their lack of relative speed may cause a non-physical reduction of the local temperature.

The accuracy of the particle quantity approximation increases with the number of computational particles because each particle is required to represent a smaller region of phase space. In addition, the statistical scatter in a given sample of the flowfield is reduced through increased particle quantity. These factors are balanced by the computational cost and machine requirements of the simulation, which also increase with particle quantity. This issue is treated in detail by Chen and Boyd in Ref. [7].

It is notable that MD also uses a reduced number of computational particles but continues to treat them as physical entities. This is justified by noting that Kn is maintained (for flow similarity between the real and simulated cases) if the particle diameter is increased to hold the mean free path constant as the number density is reduced.[8] A large number of small particles may therefore be replaced with a small number of large particles without affecting the important flow characteristics. This argument fails, however, when the resulting particle diameters become too large compared to the geometric feature size or the inter-particle spacing.

1.4.2 Particle Movement

In a real gas, molecules move along their current trajectories until they strike another molecule or a boundary. This is the procedure used for movement in MD, consistent with its treatment of computational particles as physical entities. Unfortunately, each particle's next collision depends on its trajectory and position with respect to all other particles in the domain, as well as the boundaries. In the worst case, the computational work of this combined move/collide operation therefore scales as N^2, where N is the total number of computational particles. While clever ways of minimizing this work have been proposed, such as storing the time of next interaction for all particle pairs and recalculating only when it changes through the collision of one or both members[8], the computation quickly becomes unmanageable as the number of particles increases.

One of DSMC's defining assumptions is made in response to this problem: if particles are only allowed to move for a short time (some fraction of a molecule's mean time between collisions), then the movement and intermolecular collision steps may be decoupled. The calculation is therefore divided into 'time steps', each consisting of independent move and collide phases. Interactions between particles and the domain boundaries are still handled as they occur in the move phase, but all intermolecular collisions are performed in the collide phase. This reduces movement from an order N^2 to approximately an order N calculation. It may be noted that this also makes it possible for two molecules to simultaneously occupy the same position. This seemingly troublesome, non-physical event is not significant, however, because the computational particles each represent a *range* of positions due to their statistical nature. Their exact position on the grid is therefore not important.

1.4.3 Particle Collisions

In MD, interparticle collisions are physical events, with the collision partners and post-collision velocities determined according to the pre-collision particle trajectories and diameters. These diameters, as mentioned above, are scaled to maintain the proper collision rate for the reduced number of particles in the calculation.

For DSMC, however, the simplifications made above have eliminated the physical means for determining these collisions. An alternate approach must therefore be supplied. This involves both calculating the correct number of collisions to perform and choosing proper partners for each of them.

Quantity Calculation

Physical arguments from classical kinetic theory may be used to develop an expression for the number of collisions per unit time per unit volume of a gas, N_c:

$$N_c = \frac{1}{2} n^2 \overline{\sigma c_r} \tag{1.1}$$

where n is the number density, σ is the molecular cross-section (single-species flow), c_r is the relative speed of the colliding partners, and the overbar signifies a mean taken over all possible partners. Unfortunately, the computational work to evaluate this mean scales as the total number of particles squared. Several ways of eliminating this term have therefore been proposed, such as Bird's Time Counter (TC) and No Time Counter (NTC) schemes and Baganoff and McDonald's method.[9][10][11] The current code uses the NTC scheme, where the local maximum σc_r encountered in the calculation is used in Eq. 1.1 to determine the number of particle pairs to consider for collision. These pairs are then accepted with a probability proportional to their σc_r.

Partner Selection

Choosing potential collision pairs is best done according to some sort of 'close physical proximity' criterion. The most computationally convenient of these is to divide the domain into a number of 'cells', as is commonly done for traditional CFD calculations. Collision pairs are then chosen among particles in the same cell. For this to be a valid approximation, these cells must be 'small'. Rigorously, this means that their linear dimensions should be comparable to the mean free path, λ. The implications of relaxing this requirement to the 'negligible gradients across the cell' constraint used in continuum CFD, are investigated in Chapter 4.

1.4.4 Result Reporting

The cells required by the collision scheme are also used in reporting the results of a simulation. This is necessary because macroscopic variables, such as pressure and temperature, are typically the quantities of interest from a computation while the method functions in terms of microscopic variables, such as individual particle positions and velocities. To determine the former from the latter, the state of all particles in some small volume surrounding the point of interest must be sampled. In the current code, the particles in each cell are used to calculate the macroscopic variables which are reported at its centroid.

Due to the total particle quantity restrictions, imposed by computational considerations, and the cell size constraints, imposed by collision and sampling concerns, there may be as few as twenty computational particles in a given cell. Calculating the macroscopic variables from a single sample will therefore yield unacceptably large uncertainties. Several samples of each cell are therefore necessary. If steady-state information is required, these samples may be taken every few time steps[1] until acceptable statistical convergence is reached. If time-accurate information is required, ensemble averaging is performed, were the entire flow evolution is calculated repeatedly, sampling at the same temporal locations in each iteration.

A flowchart of a typical DSMC calculation is presented in Figure 1-1.

[1]the samples would not be statistically independent if they were taken every time step

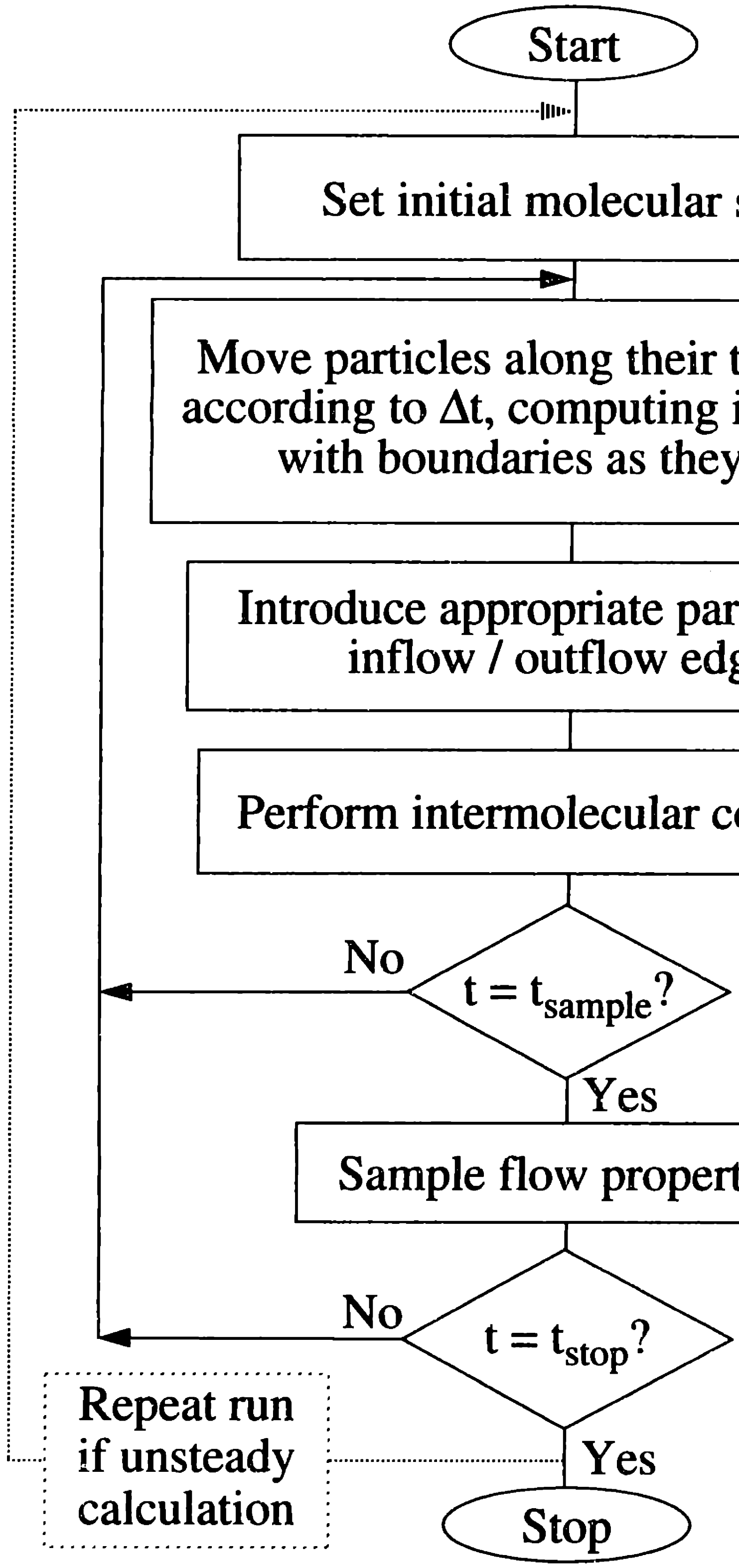
Start
Set initial molecular state
Move particles along their trajectories according to Δt, computing interactions with boundaries as they occur
Introduce appropriate particles at inflow / outflow edges
Perform intermolecular collisions
No
t = t_sample?
Yes
Sample flow properties
No
t = t_stop?
Yes
Repeat run if unsteady calculation
Stop

Figure 1-1: Flowchart for a typical DSMC calculation.

Chapter 2

Algorithm

Due to the wide variety of geometries and flow conditions present in MEMS devices, flexibility was a primary goal in constructing the algorithm for this investigation. DSMC has an inherent advantage over traditional CFD in this regard because its particulate nature makes it uniformly valid for all *Kn*, without regime-specific modifications. This is especially important for MEMS work because many devices contain mixed-regime flows.

Flexibility concerns also drove the particular implementation of DSMC developed for this work. First, C was chosen for the coding due to its flexible structure, dynamic memory allocation, and recursive function capability. In addition, the code was written to run in non-dimensional form, use unstructured grids, and store most data locally. The details of the resulting algorithm are discussed in the following sections.

2.1 Non-Dimensionalization

All quantities in the code are non-dimensionalized. This generalizes the construction of calculations and facilitates the interpretation of their results. The normalization factors selected for this purpose are relatively common in the DSMC community.

Due to the importance of *Kn* in the subject geometries, the mean free path, at some reference condition, is a logical choice for non-dimensionalizing length. Similarly, DSMC's particulate nature points to one of the molecular speeds (ie. mean, rms, or most probable) as the normalizing factor for velocity. These speeds differ by a constant factor near unity, so the choice is essentially arbitrary. The most probable molecular speed, c'_m, is used in this work. The quotient of the length and speed normalizations, λ/c'_m, $(2/\sqrt{\pi}$ times the mean collision time) is used to non-dimensionalize time. Temperature, pressure, and number density are non-dimensionalized by their values at a reference condition.

To non-dimensionalize the expression governing the number of collision pairs to be sampled in the NTC scheme:

$$N_p = \frac{1}{2} n^2 (\sigma c_r)_{max} \tag{2.1}$$

a normalizing factor must be chosen for the collision cross-section, σ. Here it should be noted that the probability of a particular molecule suffering a collision is proportional to the product of its cross-section, its path length, and the number density. For a non-dimensional flow representation to be similar to its dimensional counterpart, the collision probabilities for comparable particles must match. As a result, the scale factors for collision cross-section, length, and number density must have a product of unity. The proper cross-section scale factor is therefore $1/(n_{ref}\lambda_{ref})$.[12] This conclusion may also be reached through purely dimensional arguments if the units for σ are viewed as 'length-units squared per particle', rather than simply 'length-units squared'.

The non-dimensionalization / normalization factors used in the code are summarized in Table 2.1.

2.2 Molecular Model

An important defining feature of any DSMC code is its molecular model. This specifies the collision cross-section used in Eq. 2.1 as well as the post-collision scattering law. There is, again, a tradeoff to be made between physical realism and computational efficiency. The full physical description of intermolecular interaction is not

Variable	Scale Factor
Length	λ_{ref}
Velocity	$c'_{m_{ref}}$
Time	$\frac{\lambda_{ref}}{c'_{m_{ref}}}$
Temperature	T_{ref}
Pressure	p_{ref}
Number Density	n_{ref}
Collision Cross-Section	$\frac{1}{n_{ref}\lambda_{ref}}$

Table 2.1: Non-Dimensionalization / Normalization Factors

known and, for most flows, not necessary. As a result, many models have been proposed which attempt, with varying degrees of detail, to enable the computation to reproduce the important features of a collisional flow, such as the viscosity coefficient and its temperature dependence, without becoming unnecessarily complex. Examples include the Inverse Power Law (IPL), Hard Sphere (HS) and Maxwell models of classical kinetic theory, the Variable Hard Sphere (VHS) model of Bird, the Variable Soft Sphere (VSS) model of Koura and Mastumoto, and the Generalized Hard Sphere (GHS) model of Hassan and Hash.[13][14][15][16]

The VHS model is implemented in the current code. This model grew from the classical HS representation, which assumes a constant collision cross section and isotropic scattering. For real molecules, however, the effective cross-section is reasonably constant only at very low temperatures; at higher temperatures, it decreases with increasing translational kinetic energy and relative speed of the colliding partners.[1] In addition, post-collision scattering is decidedly non-isotropic. Bird noted, from numerical and analytical studies, that changes in collision cross-section have a strong influence on the gas behavior while changes in the scattering law do not. He therefore constructed a model with a c_r-dependent cross-section, but with isotropic scattering; essentially creating hard-sphere molecules with variable diameters (hence the name).

This retains most of the computational simplicity of the HS model, but more accurately reproduces the temperature dependence of the viscosity coefficient.

In this model, an empirical constant, ω, related to the exponent, η, of the inverse power law molecular force by

$$\omega = 2/(\eta - 1). \tag{2.2}$$

is supplied to establish the working fluid. The collision cross-section is then given by

$$\sigma_d = \sigma_{ref_d} \left(\frac{m_{r_d} c_{r_d}^2}{2(2-\omega)kT_{ref_d}} \right)^{-\omega}, \tag{2.3}$$

where k is Boltzmann's constant and m_r is the reduced mass of the colliding pair, $m_r = m_1 m_2/(m_1 + m_2)$, which is always $m/2$ for the single-species gases considered in this work (m is the molecular mass). Subscript "d" denotes a dimensional quantity.

Under the normalizations introduced in the previous section, using the VHS mean free path given by Bird:

$$\lambda_d = (T/T_{ref})^{\omega} / [\sqrt{2}(2-\omega)^{\omega}\Gamma(2-\omega) n_d \sigma_{ref_d}], \tag{2.4}$$

the non-dimensional collision cross-section becomes:

$$\sigma = \frac{2^{\omega}}{\sqrt{2}\,\Gamma(2-\omega)} c_r^{-2\omega}, \tag{2.5}$$

where $\Gamma()$ denotes the gamma function.

2.3 Grids

In response to the stated goal of maximum flexibility, the current code was written for unstructured grids. This enables it to treat geometries of arbitrary complexity without modification. In addition, several sophisticated generation and adaptation schemes are available for grids of this type. Though the code was written with a generalized cell geometry in mind, all calculations discussed in Chapters 3 and 4 use 3-sided cells generated as a Delaunay triangulation of points distributed in the domain and on the boundaries. This triangulation was performed using Watson's algorithm[17], which made possible the inclusion of 'point-and-click' node placement for pre-calculation local refinement.

A sample grid generated in this manner is presented in Figure 2-1. The geometry shown is a channel with a 25% bump, similar to the case run in Section 4.2.3. Grid refinement is demonstrated at the left edge of the bump.

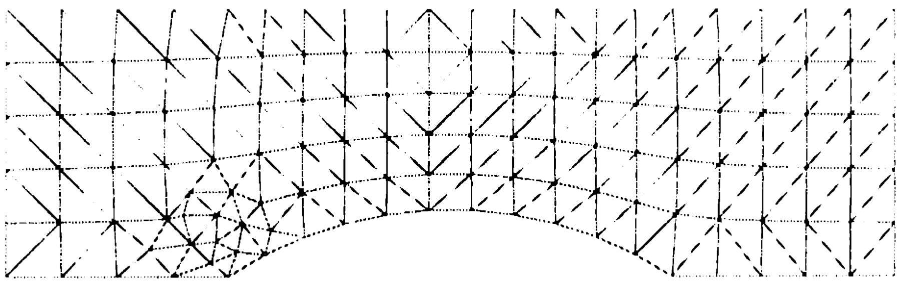

Figure 2-1: Example of an unstructured grid with local refinement generated with Watson's algorithm.

2.4 Particle Movement

It is common practice for DSMC codes on structured grids to displace all particles a full time step along their trajectories, then determine their resulting cell indices through a search or mathematical operation. By their nature, however, unstructured grids make this type of scheme very difficult. A solution to this problem was proposed (though for structured 3-D grids) by Dietrich[18]: perform particle movement and current-cell identification simultaneously by maintaining knowledge of a particle's current cell at all points in its trajectory. To accomplish this computationally, a particle is displaced until it contacts a face of its current cell. It is then passed to an adjoining cell, reflected from a solid boundary, or allowed to leave the calculation through an inflow/outflow edge, as specified by a 'neighbor identifier' stored for each face of the cell. This arrangement is extraordinarily flexible because the grid need only be composed of cells with valid neighbor identifiers on each of their faces. These cells can, strictly speaking, possess an arbitrary number of edges and be of any shape or orientation. From a computational viewpoint, however, this method is most convenient if the cells are required to be convex. This allows the faces to be treated as infinitely long and the impacted face to be found by selecting the line that the particle trajectory intersects at the earliest time.

This portion of the code proved to be the most sensitive to numerical accuracy issues. Due to the imperfect machine resolution of position and cell information, it is difficult to calculate the proper destination cell for particles which are close to faces or cross near nodes. A number of measures were implemented to combat this problem.

The first of these involves noting the face through which a particle entered its cell. This face is then excluded from consideration as a crossing site for the remainder of the current time step. Unfortunately, due to the unstructured nature of the grid, no mathematical means exist for identifying the entry face of the new cell from the (known) exit face of the old cell. A search for a face in the new cell with a neighbor identifier pointing to the particle's former cell is therefore performed.

From this effort, calculating a useless (and problematic) intersection time is avoided. This quantity is problematic because particles are displaced to their intersection point when passed to a new cell. The intersection time in the new cell for this face should therefore be zero, but will, in practice, be a small number due to machine resolution issues. If this number is positive, this face is likely to be selected as the next intersection because it has the smallest time to occurrence, causing the particle to be passed back to its original cell, which will return it, resulting in an infinite loop. The computation and memory cost of searching for and storing a particle's entry face is therefore justified.

Another test is included for crossings which occur very close to cell nodes, such as that shown in Figure 2-2. In this case, the particle will be transferred from cell one to two and then to three without difficulty. A problem is encountered when searching for intersections in cell three, however. The face between cells two and three is rejected because it was just crossed, but an intersection time near zero is calculated for the other face which shares the node skirted by the particle. This causes the particle to be (erroneously) passed to cell four without significantly moving it (due to the miniscule intersection time). In a similar fashion, cell four transfers the particle to cell one, which transfers it to cell two, and so on. This results in an unending cycle because each of these moves has a negligible duration, so the time step is never completed.

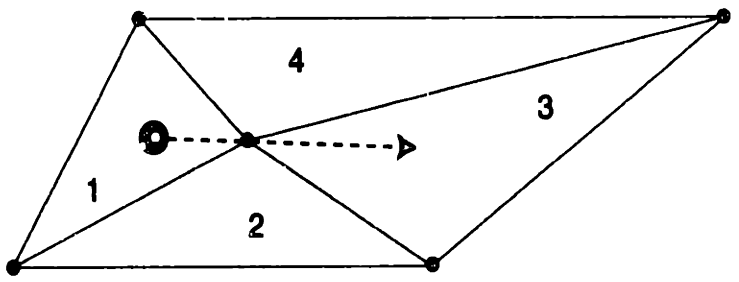

Figure 2-2: Sketch of a particle crossing near a grid node

To avoid establishing this cycle, a 'direction test' is performed on potential intersection faces. In this test, the particle trajectory is projected onto the inward-pointing normal of the face in question. If the projection has a positive sign, then the particle is not moving in the proper direction to strike this face and the intersection is rejected. To avoid encumbering routine cases with this test, a tolerance is defined. The test is then only performed on faces whose intersection times fall within this tolerance of zero.

A code listing of the movement function is provided in Appendix A.

2.5 Data Structure

Many DSMC codes use a single array to store the data for all particles in the simulation. A 'cross-reference vector' is then maintained by each cell, containing the indices of its particles in the central array. Moving a particle from cell to cell then consists of simply transferring its index from one cross-reference vector to another. This is a very efficient arrangement for machines with rapid access to all their memory, such as supercomputers, because very little data must be moved with the particle.

Dietrich and Boyd noted, however, that this scheme causes a significant loss of computational efficiency on 'workstation' computers.[19] These machines, unlike supercomputers, have a relatively slow main memory but a very fast cache. Before performing an operation, this cache is loaded with the segment of main memory containing the required data. If subsequent operations access only items already in the cache, they execute very quickly. Conversely, if the cache must be reloaded with new data, a condition known as a 'cache miss', a considerable amount of time is lost in the process. The central storage of particle data causes many cache misses when performing inherently cell-based operations, such as collisions and sampling, because the members of a given cell are scattered through a very large area of main memory.

The alternative is clear: physically store a particle's data in its current cell instead of in a large, centralized array. This increases the expense of moving a particle from cell to cell, but greatly decreases the number of cache misses suffered by cell-based operations.

An algorithm using the trajectory-tracing particle movement scheme outlined in the previous section is easily written with entirely cell-based operations, so considerable gains are possible from improved utilization of the cache. In addition, for an equilibrium gas with a properly chosen cell size and time step, less than 50% of particles typically leave their cell during a given time step, and, of these, less than 10% typically leave their new cell. The increased cost of inter-cell movement in this arrangement is therefore greatly overshadowed by the increased speed of cell-based data structures.

This data structure has the additional advantage of easy adaptability to a message-passing parallel environment. If a particle moves to a cell on a different processor, the inter-cell communication step is simply augmented with an inter-processor communication step. In addition, because a cell is now a complete data unit, containing its geometry, neighbors, and resident particles, it is easily passed as a whole to another processor as part of a dynamic load balancing scheme.

2.6 Inter-Cell Communication

Upon adopting a cell-based data structure, an efficient means for moving particle information between cells is required. The simplest method, which is similar to that used by Dietrich and Boyd for communication between parallel domains, is to maintain a ‘communication link’ for each cell. This is an array for particles waiting to be accepted into the cell. If a particle leaves its cell during the movement step, it is simply placed in the communication link of its destination cell. When a cell has completed the movement step for all of its particles, it begins to process its communication link: each of the incoming particles is checked to see if it will remain in the cell; if so, it is moved to the particle list for that cell, if not, it is moved to the communication link of its next cell. This process continues until all communication links are empty. This arrangement was found to be fast, but extremely demanding of memory; the communication links typically grew to be about half as large as the particle arrays for their cell. The total memory required for the program was therefore 50% larger than without links.

To circumvent this difficulty, a new scheme was devised whereby particles are simply marked with a flag signifying that they are not leaving the cell, the index of their destination cell, or a flag signifying that they have exited the cell and their position on its particle list is now vacant. Once the initial movement step is completed for all particles in all cells, a ‘communication phase’ is entered in which particles leaving their current cells, referred to as ‘travelers’, are moved to their destination cells. To accomplish this, the particle array in a traveler’s destination cell is checked for empty positions. If one is found, the particle is moved to its new cell, marking its former position as vacant. If no vacancies are found, the destination cell is searched for travelers. If a traveler is found, the communication function is recursively called to move this particle to its destination cell, then the original particle is moved into the space it left. If neither vacancies nor travelers are found in the particle array of the destination cell, the incoming particle is simply added to the end.

After careful optimization, this scheme was found to run nearly as fast as the communication link case, but with a 20% reduction in memory requirements for a half-million particle calculation.

A code listing of the communication function is included in Appendix B.

2.7 Boundary Conditions

As for any solution method, proper specification of the boundary conditions is critical to a successful DSMC simulation. The means of specifying these conditions in particle methods, however, differs considerably from continuum CFD. In continuum CFD, macroscopic variables, such as temperature and velocity, are imposed at given points based on their intended physical state. For particle methods, these conditions must be translated into rules for treating individual particles near these points.

2.7.1 Solid Walls

As noted for intermolecular collisions, many details of gas-surface interaction are still unknown. Again, models which reproduce the important features of the physical situation have been developed. The most common of these divides particle reflection from solid surfaces into two classes: specular and diffuse. A given interaction may be described completely by one of these classes or by some combination of the two. This description is quantified by the tangential momentum accommodation coefficient, F, which varies from 0, for no accommodation (specular reflection), to 1, for full accommodation (diffuse reflection). In the code, this is accomplished by treating F as the probability a given particle reflection will be treated diffusely.

Specular Reflection

In a specular reflection, the normal component of the impinging particle's velocity vector is simply reversed and the tangential component is left unchanged. No modification is made to the energy of the particle. This is intended to model an interaction with a perfectly smooth (ie. frictionless) surface. It may also be used to simulate a symmetry plane.

Diffuse Reflection

In a diffuse reflection, the impinging particle is emitted from the surface without regard to its incoming state. The outgoing velocity is randomly assigned according to a half-range Maxwellian distribution at the wall temperature. This is known as full thermal and momentum accommodation and may be viewed in terms of a particle that is absorbed, then re-emitted at equilibrium with the surface. This is intended to model an interaction with a completely rough surface, which is considered a valid description of most engineering materials. MEMS devices, however, often contain surfaces which are cut along the crystal planes of Silicon wafers. Tangential momentum accommodation coefficients considerably less than one are therefore possible on these extraordinarily smooth surfaces.

2.7.2 Inflow/Outflow Faces

Inflow/outflow (I/O) faces are considerably more difficult to treat than solid boundaries, particularly for low-speed cases, such as those presented in Sections 3.1 and 3.2. Ironically, this task appears to be straightforward and well-defined: simply introduce and remove particles to obtain the desired flow state. Upon closer inspection, however, it is found to be another sensitive compromise between physical detail and computational efficiency. In this case, an improper formulation leads to a strong flow adjustment (very similar to a shock) near the boundary. This effect is shown in Figure 2-3 for a channel with a specified pressure ratio of 4, which relaxed to approximately 3.65. Unfortunately, the strength of this relaxation is difficult to predict, frustrating efforts to model a specific geometry and flow condition.

In the current formulation, I/O faces are treated in two stages of the calculation: particle movement and boundary enforcement.

During the movement stage, particles are simply removed from the calculation if they encounter an I/O face. The data structure and communication schemes outlined previously make this a straightforward operation: the particle's position in its cell is marked vacant, as if it has moved to another cell, but it is not labelled as a traveler, effectively moving it 'nowhere'.

During the enforcement stage, particles are introduced at I/O faces to maintain user-specified boundary conditions, which are expressed in terms of macroscopic variables. These boundary conditions, combined with quantities calculated from the

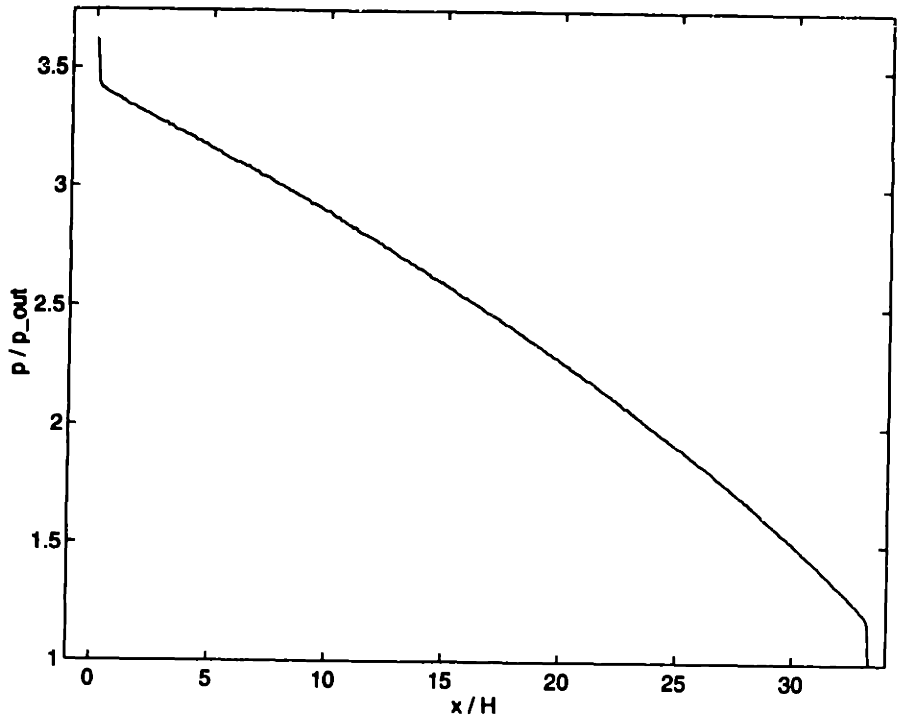

Figure 2-3: Pressure distribution for a channel with poorly-formulated IO treatment.

current flow state, determine the number of particles to introduce and their velocity distribution, as detailed below. It should be noted that, although the mechanism for enforcing boundary conditions differs considerably between particle methods and continuum CFD, the choice of which macroscopic variables to specify externally and which to calculate from the domain is determined in both techniques by the 'characteristic lines'.

Characteristics lines, or simply 'characteristics', are paths in space and time, derived from the Euler equations, along which certain flow variables remain constant.[20] They are therefore said to "carry" information from one place to another. Characteristics are used when formulating boundary conditions to prescribe how much information is communicated to the boundary from inside the domain and how much is communicated from outside. This determines which variables may be specified and which must be calculated from the flow itself.

There are four characteristic lines; one carries the entropy, one carries the transverse speed, and two carry the Riemann Invariants, J_+ and J_-, which are given by:

$$J_\pm = u \pm \frac{2a}{\gamma - 1} \tag{2.6}$$

where u is the flow speed, a is the speed of sound, γ is the ratio of specific heats, and all quantities are dimensional. The first two of these move through space with speed u and the second two with speed $u + a$ and $u - a$, respectively. Thus, three of the characteristics always point in the flow direction and the fourth points upstream for subsonic flow ($u < a$) and downstream for supersonic flow ($u > a$). A subsonic inlet therefore takes information from outside the domain on three characteristics and from inside the domain on one. A supersonic inlet, however, takes all of its information from outside the domain. Similarly, a subsonic outlet takes three characteristics from inside the domain and one from outside while a supersonic outlet takes all its characteristics from inside the domain. Over-constraining the boundary, by specifying too many variables for the number of incoming characteristics, leads, in DSMC, to strong local flow adjustments of the type shown in Figure 2-3.

Although the characteristics constrain the number of state variables that can be specified at the I/O faces, there is some freedom to choose their identity.[21] One possible arrangement is to specify the streamwise speed, transverse speed, and density, calculating the pressure from the domain. This is useful for modeling aerodynamic bodies at a given flight speed, angle of attack, and altitude, for example. A more common arrangement is obtained by substituting temperature for density in the above case so angle of attack and Mach number are the input parameters. Another possible arrangement was used for the calculations presented in Chapter 3. For these flows, the pressure, temperature, and transverse speed were specified and the streamwise speed was calculated from the domain at inflow faces. Only pressure was specified at the outflow faces of the subsonic cases and nothing was specified in the supersonic cases. These boundary conditions are intended to model fully-developed flows in channels and nozzles, (ie. flows which appear to represent a segment of a device which is far from its pressure reservoirs). This is accomplished by enabling the streamwise speed to self-adjust to a parabolic profile at the inlet and outlet which smoothly blends with the velocity profiles in the remainder of the channel.

DSMC's statistical nature complicates the boundary enforcement process, however. To determine a cell's macroscopic variables, its microscopic state must be sampled. Unfortunately, there may be as few as 20 particles in a given cell at a given time, so a single sample produces unacceptable statistical scatter. This leaves two options: neighboring cells may be included in the instantaneous sample, or it may be replaced with a time average. The former option involves the selection of neighboring cells with states 'close-enough' to that of the cell in question to yield a meaningful spatial average. The latter option involves choosing a time-averaging method that results in a sufficiently accurate estimate but still allows the flow to reach its steady state with reasonable speed. The latter option was implemented in the current code. The cell state was sampled after a movement step was completed, incoming particles were introduced at boundary enforcement, and collisions were performed. A weighted average was then taken between this result and a running value collected from previous time steps. The weighting of the instantaneous state may be varied to balance the accuracy of the running estimate with the convergence speed; a large weight makes the estimate sensitive to the statistical scatter inherent in the instantaneous average, causing error in the estimate, while a small weight causes prior information to decay slowly, retarding convergence to steady-state. A weight of 1/20 was chosen for the cases presented in this work.

Once the necessary macroscopic variables are obtained from either the user or the domain, the boundary enforcement process is identical for inflow and outflow faces: particles are introduced in sufficient quantity and with the proper velocity distribution to satisfy local constraints. It should be noted that a significant number of particles are introduced at both the inflow *and* the outflow faces in low-speed calculations, consistent with the existence of the backward-running characteristic discussed above.

First, the number of particles to introduce at an I/O face must be calculated. To accomplish this, a target number density for the cell containing the face is obtained from the specified/calculated macroscopic variables and the ideal gas relation, which is simply

$$p = nT \tag{2.7}$$

under the non-dimensionalizations of Section 2.1. A target particle quantity is then calculated through multiplication by the cell volume. The actual particle quantity is compared to this target to determine the number of particles to introduce. If the actual particle quantity is greater than or equal to the target, no action is taken; particles are never removed from a cell.

Velocity components perpendicular to the I/O face are then assigned to the incoming particles according to a Maxwellian distribution at the specified/calculated temperature. In non-dimensional form, the velocity distribution for a thermal velocity component in the transverse direction, say v', is given by:

$$f(v') = \frac{1}{\sqrt{\pi T}}\, \mathrm{e}^{-\frac{v'^2}{T}} \tag{2.8}$$

Values are selected from this distribution directly using a method presented by Bird in Ref. [1]: select two independent random numbers, R_{f_1} and R_{f_2}, then calculate v' from:

$$\begin{aligned} \theta &= 2\pi R_{f_1} \\ r &= \sqrt{-T \ln(R_{f_2})} \\ v' &= r \sin(\theta) \end{aligned} \tag{2.9}$$

At inflow faces, the specified transverse velocity (which is zero for all cases in this thesis) is added to this value. At outflow faces, the calculated transverse velocity is added. This operation is identical for all cases because one of the characteristics discussed above carries the transverse velocity specifically and always moves in the flow direction.

The velocity component normal to the I/O face is then assigned according to a fluxal distribution, ie. a distribution which is shifted based on the mean normal velocity through the face. The resulting (non-normalized) velocity distribution is given by:

$$f_{u'} = u' \, e^{-(u' - u_n)^2}, \tag{2.10}$$

where u_n is the normal component of the macroscopic (mean) velocity across the face and a positive value denotes motion into the cell (for both u' and u_n). This distribution is sampled via the acceptance-rejection method, where a randomly-selected value for u' is either accepted, with a probability equal to $f_{u'}(u')/(f_{u'})_{max}$, or rejected and another value chosen, repeating until an acceptance is made. Situations where $u' < 0$ are non-physical, because the particle could not enter the cell, and are therefore excluded from consideration.

A low aspect ratio channel with a pressure ratio of 3 and the fully-developed flow boundary conditions described above served as the test case during IO boundary development. This case was chosen because it produces strong gradients yet remains subsonic at the outflow, presenting a significant challenge to boundary enforcement. The channel geometry was 150x30 λ (referenced to the inlet) with 4000 uniform cells and approximately 180,000 particles at steady-state. A sample run is presented to demonstrate the effectiveness of the current boundary formulation.

Figure 2-4 contains the complete (ie. all cells were plotted) streamwise velocity distribution. End-effects are almost imperceptable at both the inlet and the outlet. The "long channel" boundary conditions were therefore successful and the flow in the entire domain can be considered fully-developed. The corresponding pressure distribution is shown in Figure 2-5, also with good results, exhibiting very little of the flow adjustment present in Figure 2-3.

A code listing of the boundary enforcement function is given in Appendix C.

2.8 Verification

Before treating more complicated cases, simple verification runs were made to test the algorithm formulation and coding. Cases with either an analytical solution or

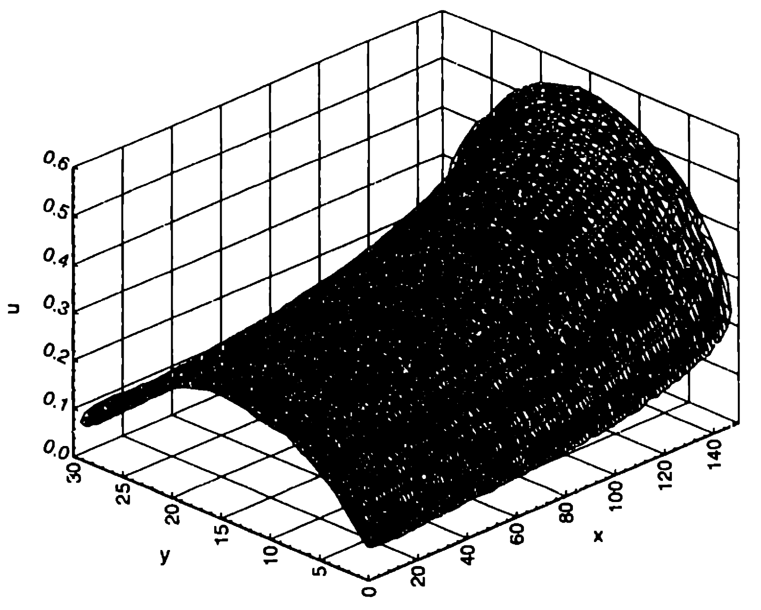

Figure 2-4: Complete streamwise velocity distribution for I/O test channel.

published results were chosen for this task to facilitate comparison with the current code. An equilibrium gas in a box and a 1-D shock wave are examined below.

2.8.1 Equilibrium Gas

All flows examined in this work have Knudsen numbers which place them comfortably in the collisional regime. The proper treatment of inter-particle encounters is therefore crucial to an accurate simulation. To test the mechanisms governing the collision frequency, the equilibrium collision rate of several different gases was calculated by simulating a resting fluid in a closed domain. The results were then compared to the theoretical prediction.

The theoretical equilibrium particulate collision rate, ν_p, represents the average number of collisions suffered per particle per unit time. It is given by the mean molecular speed, $\bar{c}'$, divided by the mean free path, λ. From Section 2.1, the non-dimensionalization factors for length and speed are λ and c'_m, respectively, both at a reference condition. Noting that $\bar{c}' = 2/\sqrt{\pi}\ c'_m$ the non-dimensional equilibrium collision rate is found to be:

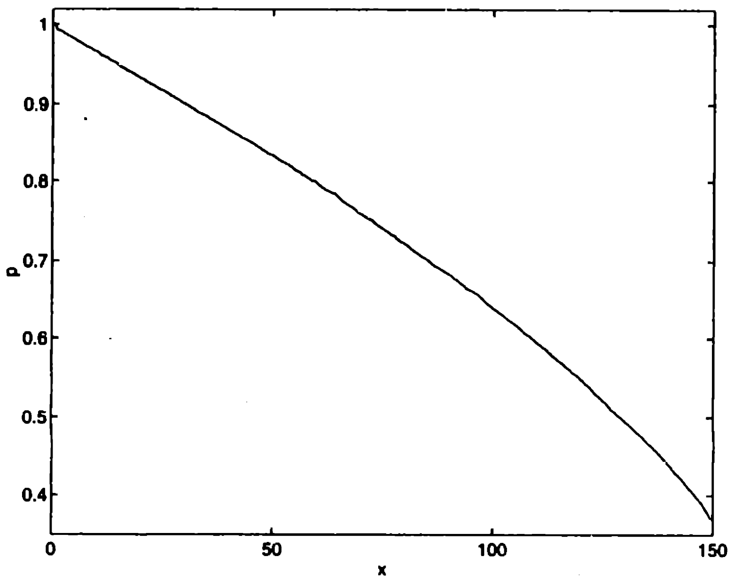

Figure 2-5: Complete pressure distribution for I/O test channel.

$$\nu_p = \frac{2}{\sqrt{\pi}} = 1.13 \tag{2.11}$$

To verify that the DSMC code correctly produces this rate, a 10x10 domain, composed of 100 cells and specularly-reflecting walls, was constructed. Five-thousand particles were distributed in this domain with velocities selected from a Maxwellian distribution at the reference temperature. The flow was then run for 1000 time steps, to allow any initial condition effects to die out, then sampled every four time steps until 1000 samples were obtained. This was performed for four gases with significantly different values of ω and the results are presented in Table 2.2.

Gas	ω	ν_p
Neon	0.16	1.15
Nitrogen	0.24	1.15
Argon	0.31	1.15
Carbon Dioxide	0.43	1.15

Table 2.2: Computed Equilibrium Collision Rates for Various Gases

First, it may be noted that the collision rate is independent of molecular species. This is supported by the theoretical prediction, as no gas constants appear in Equation 2.11. In addition, the computed rate is within 2% of the theoretical value. This error was found to be a function of the number of particles per cell, N_c, in accordance with the statements made in Section 1.4.1. This dependence is demonstrated in Table 2.3.

N_c	ν_p	$\%Error$
100	1.14	1.0
50	1.15	1.9
20	1.17	3.7
10	1.23	9.0
5	1.32	17.0

Table 2.3: Computed Equilibrium Collision Rates for Various Cell Particle Numbers

2.8.2 1-D Shock Wave

To assess the accuracy of the non-equilibrium collision rate and the post-collision scattering law, a 1-D shock wave was computed. This case was selected because analytical expressions exist for the state variable jumps across the shock and published DSMC results for the shock profile are available.

This calculation was performed on a 100x40 grid with 4000 uniform rectangular cells. The simplicity of this geometry allowed a particle's current cell to be determined by mathematical means, based on its position, the grid dimensions, and the number of cells. The trajectory-tracing movement scheme outlined in Section 2.4 was therefore not required. This enabled the shock to be created by moving the left side of the domain, compressing the entire grid with each time step to maintain its regularity. The grid was initialized with a resting fluid, and the wall was set in motion at t=0. Ensemble averaging was performed for a total of 50 ensembles.

The resulting jump in state variables across the shock may be compared to the analytical predictions derived from the 1-D flow equations.[20] This comparison is presented in Table 2.8.2 for a Mach 8 shock in Argon (u_{wall} = 5.39, ω = 0.31). An excellent agreement is obtained in all cases, with a maximum error of less than 2%.

Variable	Analytical	Computed	$\%Error$
Pressure	79.75	79.54	-0.26
Temperature	20.87	20.58	-1.39
Density	3.82	3.86	+1.05
Wave Speed	8.00	8.08	+1.00

Table 2.4: Analytical and Computed State Variable Jumps Across a Mach 8 Shock

The spatial profile of the shock may also be compared to other DSMC results, such as those presented by Baganoff and McDonald.[11] This comparison is shown in Figure 2-6 for a Mach 3 shock in a Maxwellian gas, which is equivalent to a VHS gas with $\omega = 0.5$. The agreement is, again, excellent, providing evidence that the code is indeed properly simulating the fluid behavior.

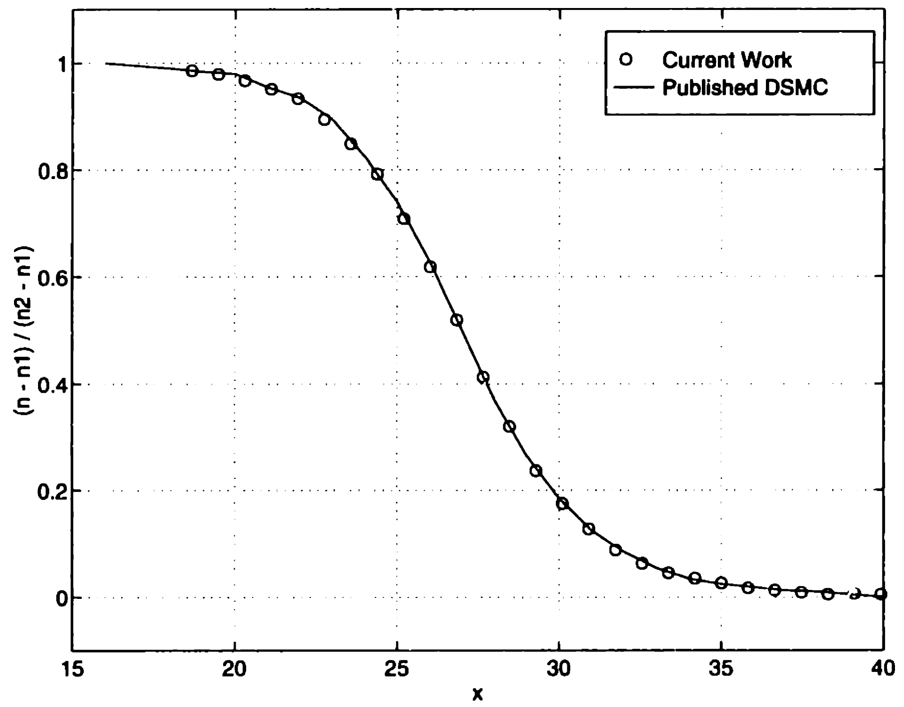

Figure 2-6: Comparison of Mach 3 shock profiles from the current code and published DSMC results.

Chapter 3

Non-Continuum Results

As discussed in Chapter 1, DSMC is primarily used for flows in non-continuum regimes, for which Navier-Stokes based methods break down. Formerly, most of these flows involved high-altitude flight. Now, MEMS devices provide a rich array of interesting flows in these regimes that have practical applications in a wide variety of areas. A sampling of these cases are examined in this chapter. This effort is intended to investigate DSMC's ability to accurately and efficiently model micro-flows as well as illuminate some of their unique features which are important to MEMS designers.

3.1 Slip Flow Regime Micro-Channel

The first case explored was a steady flow through a micro-channel with an outlet Knudsen number of 0.05, placing it in the slip flow regime. This geometry is similar to those investigated experimentally by Harley *et al.*[22] and Arkilic *et al.*[23] and numerically (with spectral element methods) by Beskok and Karniadakis.[24] This is, historically, an important canonical case for determining the effect of rarefaction on the transport terms in the Navier-Stokes equations. It is also useful as a representation of the flow along certain features common in MEMS devices, such as the space under the floating plate of a shear stress sensor or accelerometer, which is typically only one micron high but hundreds of microns in breadth and depth. In addition, it serves as an interesting calibration case to assess the accuracy of the numerical algorithm because analytical solutions have been developed for this geometry.

In one such effort, Arkilic *et al.* show that the Navier-Stokes equations may be solved analytically for a long, high aspect-ratio, isothermal channel in the slip flow regime if the boundary conditions are modified to include a *Kn*-dependent streamwise velocity (slip) at the wall, given by:

$$u_{wall} = \frac{2-F}{F} Kn \left. \frac{du}{dy} \right|_{wall} \tag{3.1}$$

where u is the streamwise velocity, Kn is the local Knudsen number, y is the transverse coordinate, which has its zero at the channel centerline, and F is the tangential momentum accommodation coefficient, discussed in Section 2.7.1.

Through this analysis, an expression may be obtained for the pressure distribution in a microchannel with diffusely-reflecting walls as a function of streamwise channel location and overall pressure ratio:

$$\mathcal{P}(x) = -6Kn_o + \sqrt{(6Kn_o + \mathcal{P}_i)^2 - \frac{x}{L}\left[(\mathcal{P}_i^2 - 1) + 12Kn_o(\mathcal{P}_i - 1)\right]} \tag{3.2}$$

where $\mathcal{P}(x)$ and $\mathcal{P}_i$ are the local and inlet pressures, respectively, normalized by the outlet value, Kn_o is the outlet Knudsen number, x is the streamwise coordinate, and L is the channel length. The distribution predicted by Equation 3.2 may be compared to a DSMC result as a test of both theory and code. Such a comparison is presented in Figure 3-1 for a $600 \times 20\lambda$ (referenced to the outlet) channel run with Nitrogen ($\omega = 0.24$) at a pressure ratio of 2.47 with the infinite channel boundary conditions discussed in Section 2.7.2. A good agreement is obtained (max error = 1.5%), including the nonlinear pressure distribution that occurs due to the large pressure drop down the length of the channel.

A theoretical expression for the streamwise velocity distribution was also developed by Arkilic *et al.*:

$$u = \frac{1}{2\mu}\frac{dp}{dx}\left(y^2 - \frac{H^2}{4} - H^2 Kn\right), \tag{3.3}$$

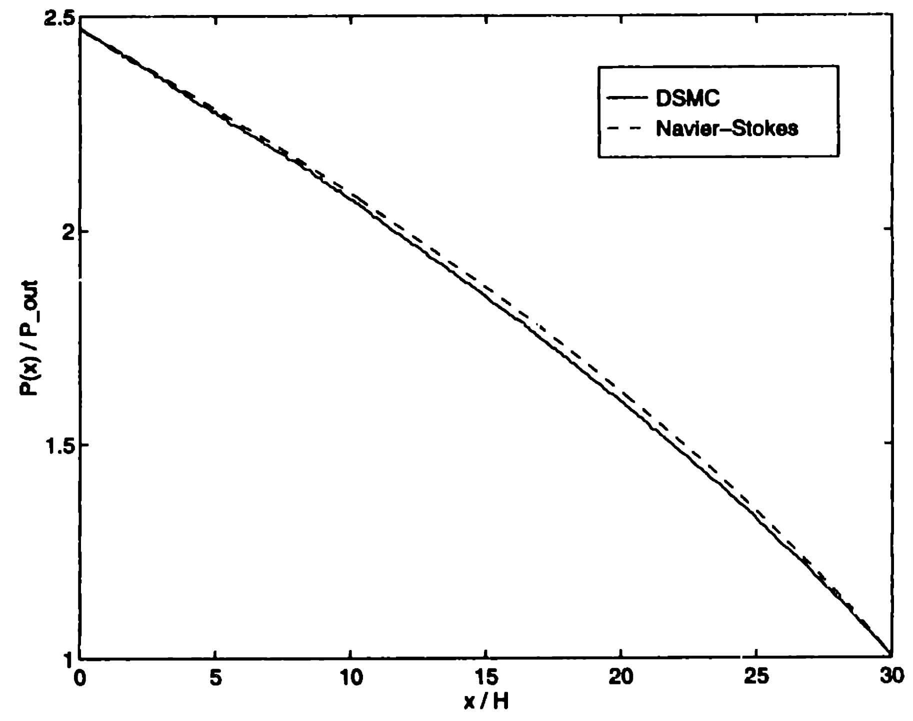

Figure 3-1: Comparison of computed and analytical pressure distributions for a microchannel in the slip flow regime.

where μ is the coefficient of viscosity, p is the pressure, and H is the channel height.

This equation is plotted in Figure 3-2 for the geometry and conditions used above. Several features unique to a flow of this type are visible. First, the fluid accelerates as it moves down the channel, unlike in the familiar Poiseuille result. This is a consequence of the density drop caused by the decreasing pressure in the streamwise direction (the flow is effectively isothermal). The mean streamwise velocity must therefore increase to maintain a constant mass flow. Second, the velocity at the walls is nonzero and increases with increasing x-coordinate. This is the aforementioned 'slip flow', which, by Equation 3.1, is essentially zero for continuum flows due to their very small Knudsen number. The increase in slip velocity down the channel is a result of growth in both Kn (from the decreasing pressure) and velocity gradient at the wall (from the accelerating flow).

The DSMC result for this configuration is presented in Figure 3-3. Comparing this to the previous figure, it may be concluded that the DSMC calculation qualitatively reproduces the mean flow acceleration and the increasing slip flow predicted

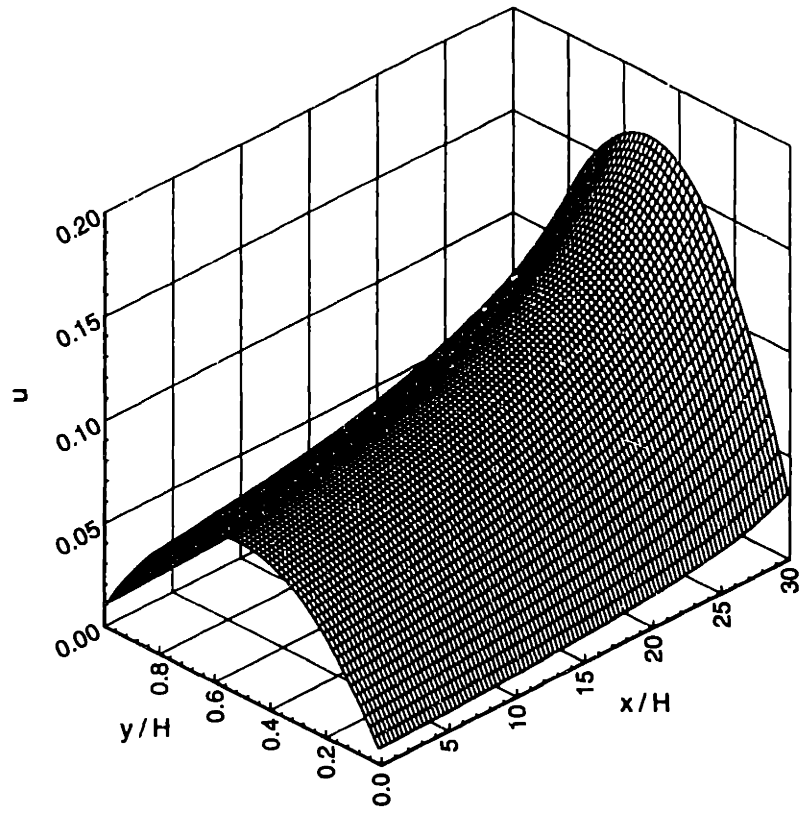

Figure 3-2: Theoretical streamwise velocity distribution for a micro-channel in the slip-flow regime.

by the theory. In addition, good quantitative agreement is obtained in the velocity distributions.

A further comparison with the theoretical analysis of Arkilic *et al.* may be found by normalizing the velocity distribution of Equation 3.3 by the average velocity at a given x-location, obtained by integrating u from the lower wall to the upper wall and dividing by H:

$$u_{ave} = -\frac{H^2}{12\mu}\frac{dp}{dx}(1 + 6Kn) \tag{3.4}$$

Rearranging the resulting expression yields:

$$-Kn + \left(\frac{1}{6} + Kn\right)\frac{u}{u_{ave}} = \frac{1}{4} - \left(\frac{y}{H}\right)^2 \tag{3.5}$$

The left side of this equation, which will be referred to as the 'similarity speed', u_s, is a function of x and y, while the right is a function of y only. Consequently, if

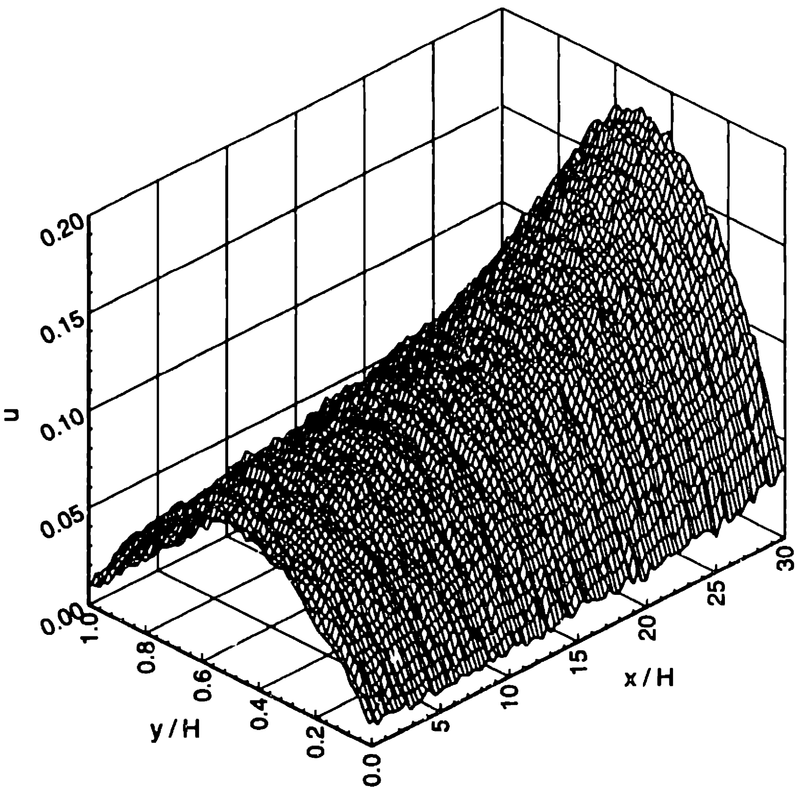

Figure 3-3: Computed streamwise velocity distribution for the micro-channel of Figure 3-2.

the slip-flow analysis holds, calculating the similarity speed using the local $Kn(x)$ and $u(x, y)$ will yield identical parabolas at all x-stations down the length of the channel.

This assertion was tested by computing a similarity speed distribution from the DSMC output. The maximum and minimum of the result at each x-location is shown in Figure 3-4. It should be noted that the upper theoretical line is not placed exactly at 0.25 because an even number of cells was used in the DSMC run, so there is no data point in the center of the channel. Also, the lower line is not at zero because the macroscopic quantities for a cell are assumed to be associated with its centroid, so there are no data points on the walls themselves.

It may be concluded from this figure that the similarity assertion indeed holds for the slip flow channel. As predicted by the analysis, the down-channel variation of the streamwise velocity profile seen in Figure 3-3 has given way to a constant similarity speed profile. In addition, the maximum and minimum similarity speeds compare well with the predicted values.

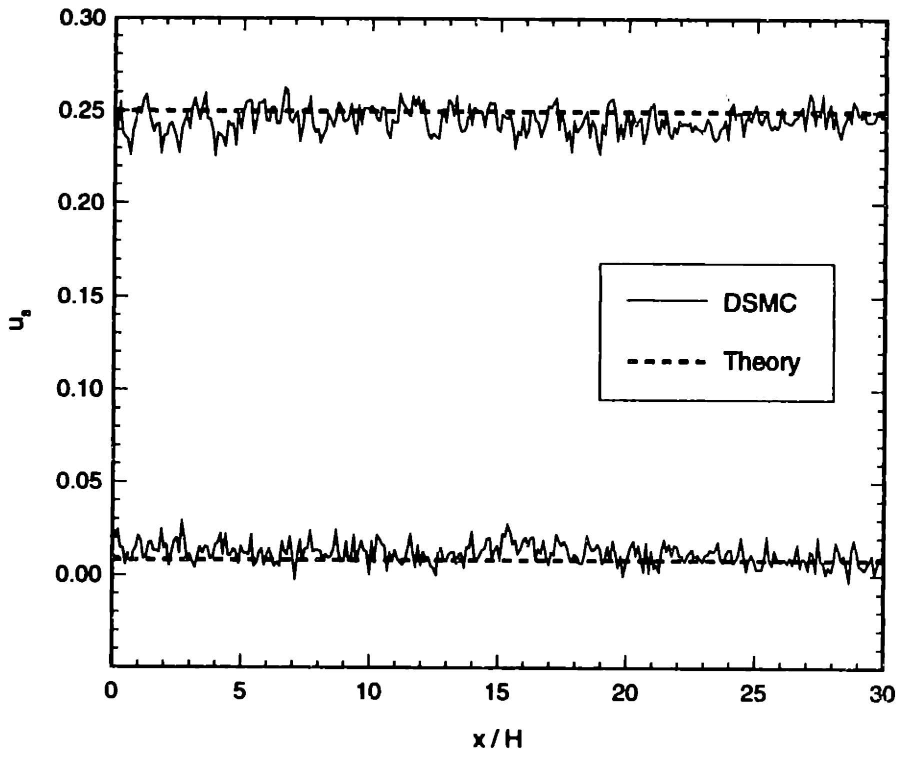

Figure 3-4: Comparison of computed and theoretical maximum and minimum similarity speeds for a micro-channel in the slip flow regime.

Overall, excellent agreement was obtained between the analytical solution of Arkilic *et al.* and the DSMC results. This supports the accuracy of both techniques. For the DSMC code, however, it is just the beginning; many more interesting flows, for which there are no reliable analytical solutions, may be easily treated with this method. The remaining cases presented in this chapter are intended to demonstrate this capability.

3.2 Transition Regime Micro-Channel

One of the great strengths of DSMC is its validity for dilute gases in all Knudsen number regimes. One of the most interesting of these is the transition regime, defined in Section 1.2 as $0.1 < Kn < 3$. Here the mean free path is comparable to the characteristic dimension of the flow. This makes analytical solution very difficult because the approximation of transport terms based on macroscopic quantities becomes inaccurate, precluding the use of the Navier-Stokes equations (even with the slip-flow boundary condition of Equation 3.1). Collisions are still important, however, so the collisionless Boltzmann equation is not yet an option. This leaves only the full Boltzmann equation; a very difficult expression to solve, either analytically or with numerical techniques.

DSMC is therefore a very attractive tool for investigating the transition regime. It is also one of the few tractable techniques which is uniformly valid in mixed Kn-regime flows. These are important features because, due to the aforementioned difficulties, relatively little is known about these cases. Such knowledge is critical, however, because many MEMS devices contain flows of this nature.

The micro-channel was again used as a sample case, this time to observe the failure of the slip-flow analysis as Kn enters the transition regime. The channel treated here is $136.4 \times 2.3\,\lambda$, with a pressure ratio of 4.2 and an outlet Knudsen number of 0.44. The working fluid is, again, Nitrogen.

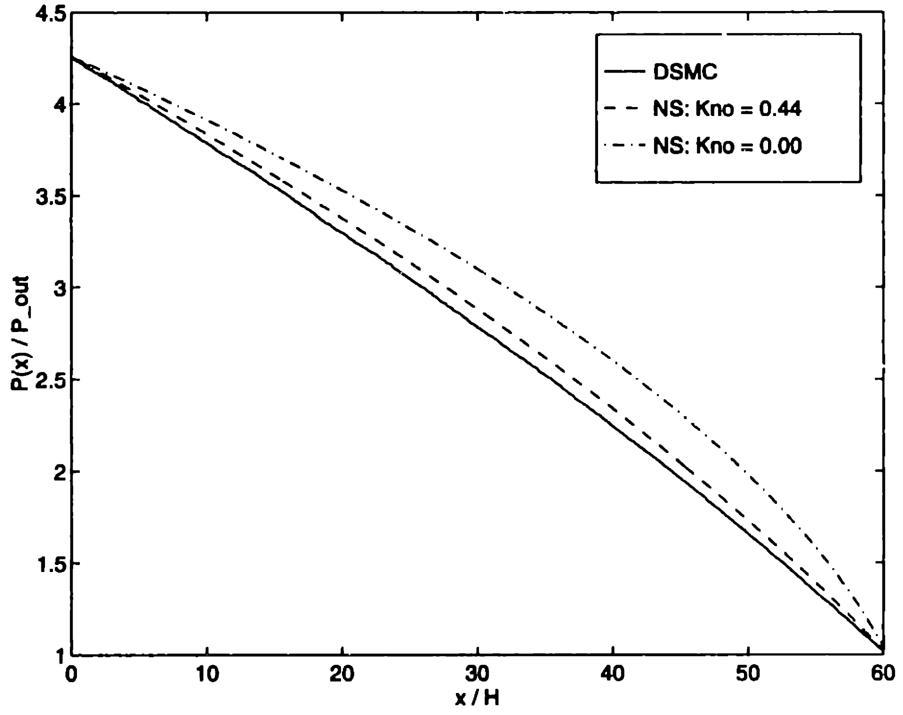

Figure 3-5: Comparison of computed, slip-flow, and continuum pressure distributions for a micro-channel in the transition regime.

Proceeding as in the slip-flow case, the computed pressure distribution is compared to the slip-flow prediction in Figure 3-5. The continuum curve ($Kn = 0.0$) is also shown for reference. As expected, the excellent agreement between DSMC and theory obtained for the slip-flow case (Figure 3-1) is no longer present. The error between the curves has grown from less than 2% to more than 4%. The form of this disagreement is also significant: the computed curve is more linear than its analytical counterpart. A trend of increasing pressure curve linearity with increasing rarefaction is therefore established by the relative shape of the continuum, slip-flow, and transition curves.

The analytical prediction for the streamwise velocity distribution in this channel is presented in Figure 3-6. Note that the theory predicts a flatter profile than for the previous case. This may be attributed to the last term in Equation 3.3, which is constant across the channel and proportional to *Kn*. The Knudsen number is an order of magnitude larger in this case, so this term has a much stronger influence on the shape of the distribution.

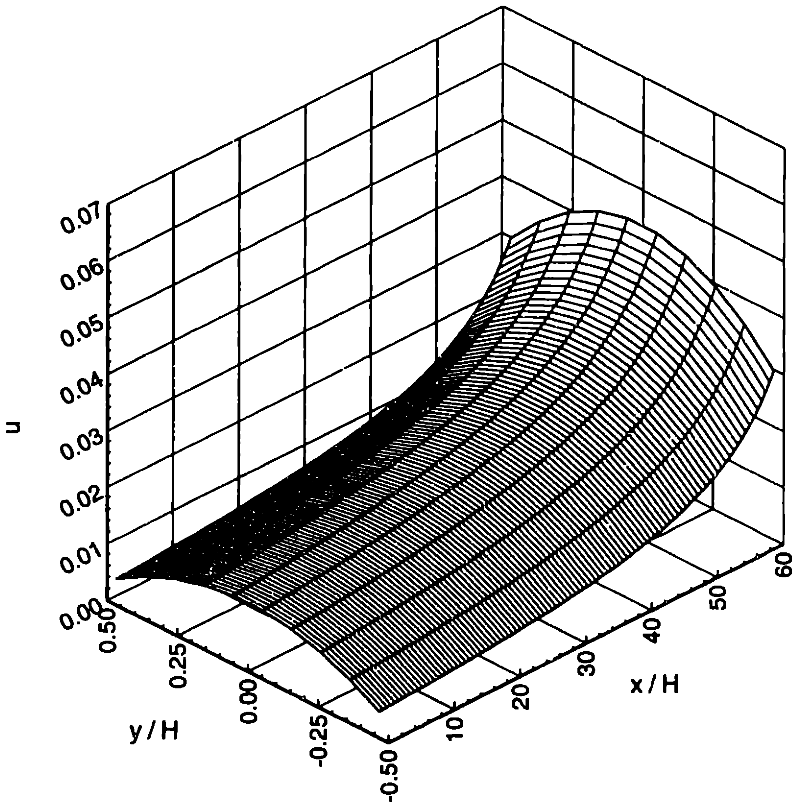

Figure 3-6: Theoretical streamwise velocity distribution for a transition-regime micro channel calculated with the slip-flow analysis.

Upon plotting the computed distribution for comparison (Figure 3-7), it becomes evident that the assumptions supporting Equation 3.3 are beginning to fail. Both

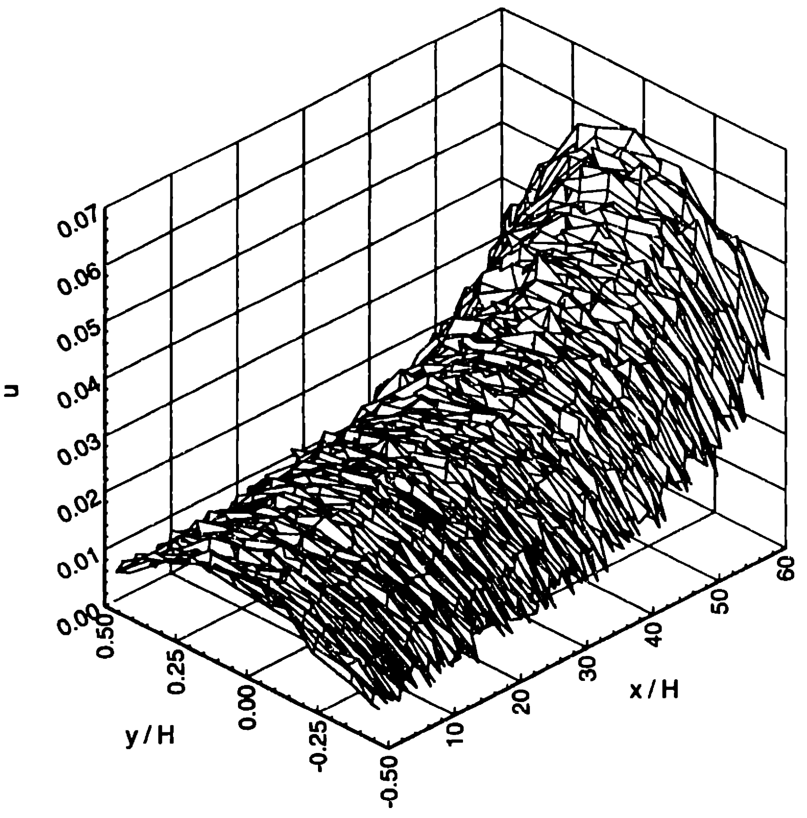

Figure 3-7: Computed streamwise velocity distribution for the channel of Figure 3-6

the slip flow and maximum speeds at a given x-location are higher than predicted by as much as 40%. This allows the channel to support a much larger mass flow than would be predicted by the slip flow theory, which, in turn, predicts a larger mass flow than the traditional Navier-Stokes analysis.

A final exposition of transition behavior may be made via the similarity analysis of the previous section. As discussed in Section 1.2, the transition regime is commonly considered to begin at $Kn = 0.1$. This assertion may be tested by noting that Kn increases with downstream position. The similarity profiles, which depend on the slip-flow solution, may then be computed for each position and compared to the analytical prediction. Because each position has a Knudsen number associated with it, the value for which the slip flow analysis fails may be determined by finding the point where the experimental and analytical curves begin to diverge. Toward this end, Figure 3-8 contains the computed maximum and minimum similarity speeds, plotted with the analytical prediction in a fashion identical to that of Figure 3-4. The Kn distribution was then overlaid to facilitate determining its value when the slip-flow analysis fails.

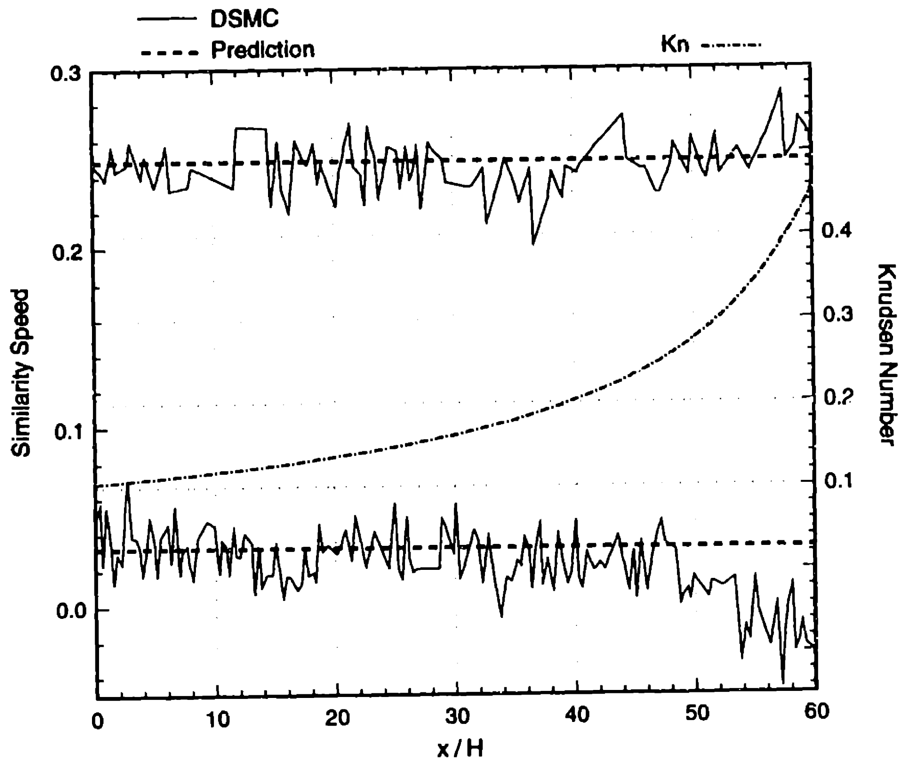

Figure 3-8: Computed and analytical similarity speeds with Knudsen number overlay.

It may be concluded from this figure that the slip flow analysis begins to fail at approximately $Kn = 0.15$. This supports the oft-used boundary for the transition region, $Kn = 0.1$. This limit may be understood if the slip boundary condition, Equation 3.1, is viewed as an expansion of the wall velocity in powers of Kn. The no-slip condition is then the zeroth-order solution, and Equation 3.1 is the first-order solution. It is therefore logical that the neglected higher-order terms would begin to significantly affect the result when Kn exceeds 0.1. A second-order accurate boundary condition in terms of the continuum variables is presented in Ref. [24], though it should be remembered that the Navier-Stokes equations themselves are only strictly valid to first order in Kn.

3.3 Supersonic Micro-Nozzles

The final cases presented in this chapter are supersonic micro-nozzles. These may be viewed as channels whose upper and lower walls form a parabolic contraction / expansion. Two such nozzles are discussed, one with an area ratio of 3.5 and sonic flow at the throat, and one with an area ratio of 2.0 and subsonic flow at the throat. The grid for the latter case is shown in Figure 3-9.

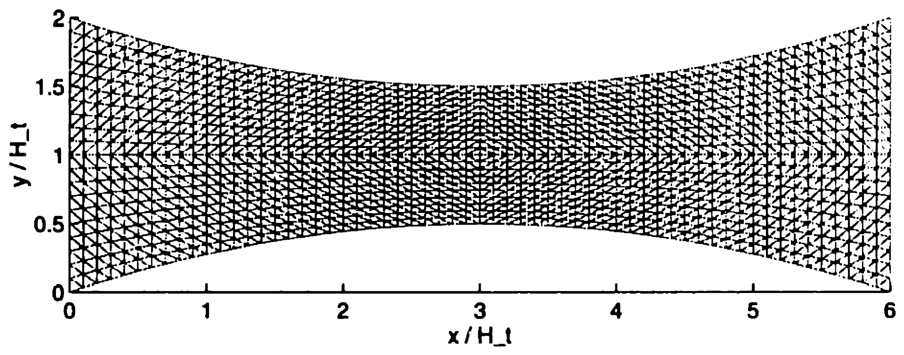

Figure 3-9: Computational grid for a micro-nozzle with area ratio 2.0.

Analytically, a sufficiently long nozzle can be considered quasi-1D and the solution given in Section 3.1 may be used, with appropriate modifications for the slowly varying channel height. The nozzles presented in this section, however, have a total length of only six times their throat height, so the quasi-1D assumption is not valid. In addition, the significant expansion may cause the *Kn*-regime to change at one or more streamwise positions. These factors make analytical and continuum-based numerical treatment of these geometries difficult. Nonetheless, nozzles such as these may play important roles in devices such as micro-rocket thrusters and micro-gas turbine generators, for example. Investigating their behavior is therefore a valuable task for which DSMC is well-suited.

Both cases were run with an inlet at 10 times the reference pressure and exhausted to 'vacuum'. The latter condition was implemented by simply removing from the simulation any particle which crossed the outlet boundary and refraining from introducing new particles at those faces. The walls were isothermal at the reference temperature with full thermal and momentum accommodation. The working fluid was Helium ($\omega = 0.20$), which was supplied at the reference temperature.

3.3.1 Sonic Throat

The first nozzle presented has an area ratio of 3.5 and 4200 cells. With the boundary conditions given above, the resulting pressure ratio was approximately 24 and the outlet Knudsen number, based on the passage height, was 0.03.

The Mach number distribution for this case is shown in Figure 3-10. A number of interesting features are visible. First, as normally expected, the flow is sonic at the throat. Second, a Mach number of 2.4 is reached. This is considerable when it is noted that the nozzle is only approximately 600 inlet mean free paths in length. Finally, the slip flow speed is substantial, exceeding Mach 0.5 near the outlet.

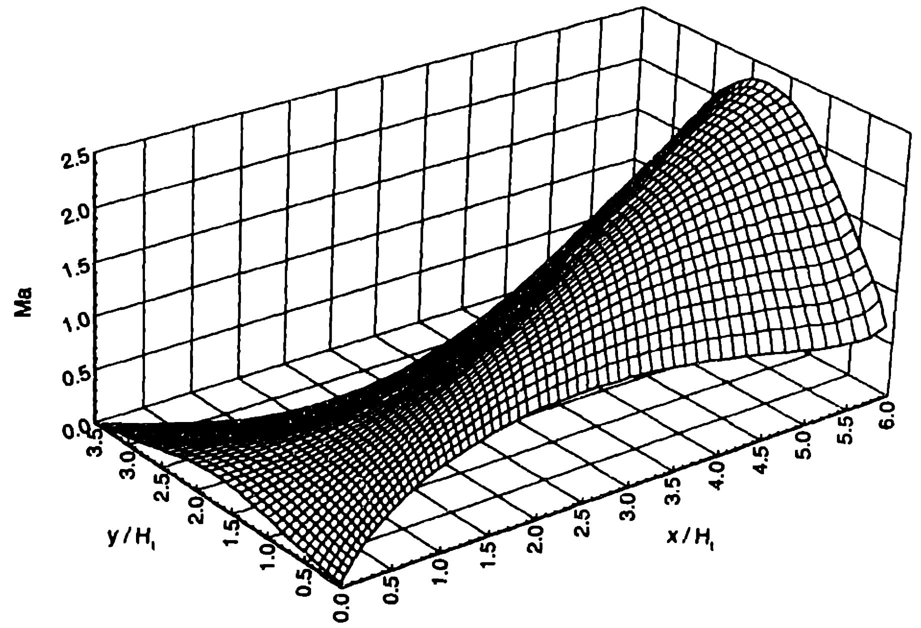

Figure 3-10: Mach number distribution for a micro-nozzle with a sonic throat.

The temperature distribution for this micro-nozzle is shown in Figure 3-11. It is clear that, unlike the channel case, this flow cannot be considered isothermal. A strong temperature gradient exists in both the streamwise and transverse directions, creating another obstacle to analytical treatment. This is also a very notable feature when the diminutive dimensions of the nozzle are considered. Though the walls are only about 30 local mean free paths apart at the exit and are isothermal with full energy accommodation, the fluid is still able to realize a substantial reduction in temperature. The effect of rarefaction has therefore been to significantly reduce the thermal communication between the wall and the fluid. This assertion is supported by noting the large thermal slip at the wall.

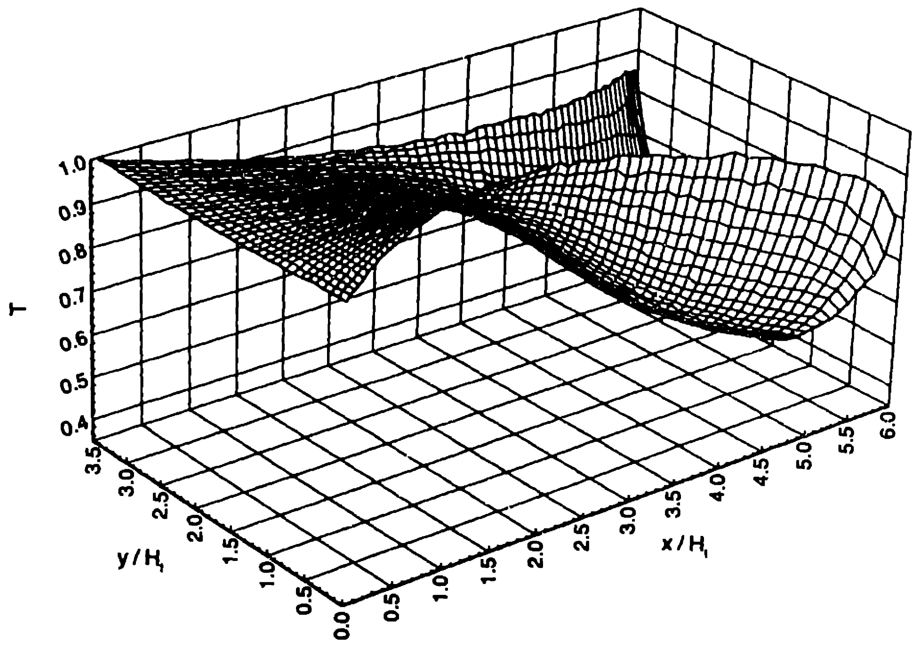

Figure 3-11: Temperature distribution for a micro-nozzle with a sonic throat.

3.3.2 Subsonic Throat

The second nozzle presented is identical to the first, except its area ratio is reduced to 2.0 and its grid to 2400 cells. With the boundary conditions given above, the resulting pressure ratio was 10.2 and the outlet Knudsen number was 0.03.

The Mach number distribution for this case is shown in Figure 3-12. This distribution was plotted in the same manner as the previous case, only the viewpoint was shifted to look along the y-axis. An interesting feature is now visible: the outlet flow is supersonic, but the sonic point is downstream of the throat. This result is at odds with the inviscid, quasi-1D conclusion that sonic flow may only be attained at a point of minimum area.[20] The highly viscous nature of this flow (due to the close proximity of the walls) invalidates this prediction, however, causing the gas to continue to accelerate downstream of the throat despite the fact that it is still subsonic and the duct is diverging.

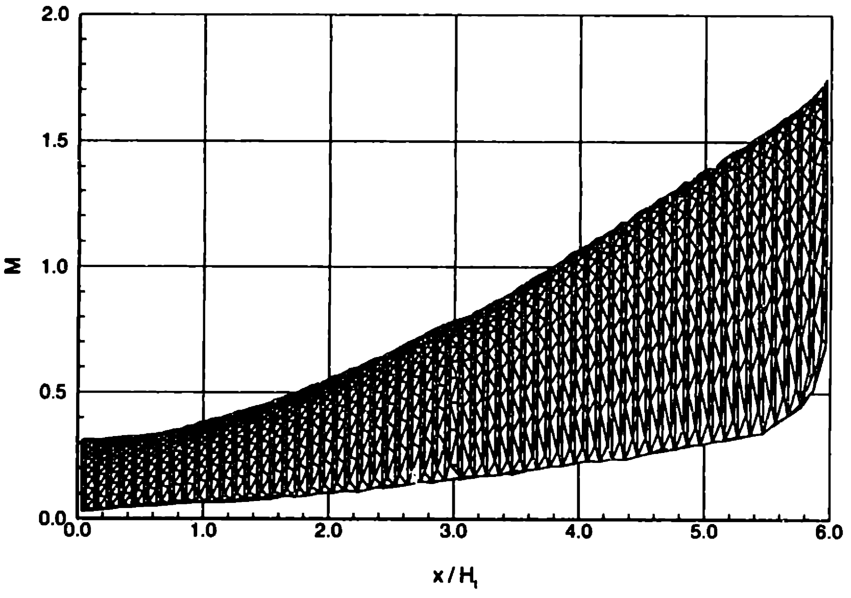

Figure 3-12: Mach number distribution for a micro-nozzle with a subsonic throat.

Because a significant portion of this nozzle is subject to the competing effects of deceleration due to geometry and acceleration due to viscosity, its outlet Mach number is considerably ($\approx$ 30%) smaller than the previous case for similar boundary conditions. This highlights the importance of proper design in such a device. As one of the few analysis tools valid for these flows, DSMC has considerable value to such an effort.

Chapter 4

Scaling Issues[1]

The previous chapter demonstrated that a conventional DSMC code can accurately treat many geometries of interest to MEMS designers. Unfortunately, the demanding cell size and time step constraints outlined in Chapter 1 make DSMC modeling of even certain micro-devices very difficult. For example, the mean free path and most probable molecular velocity of air at standard, sea level conditions are approximately 60 nm and 414 m/s, respectively. Sizing the cell lengths at one λ and the time steps at $\frac{1}{4}c'_m$, as called for in Chapter 1, a 2-D simulation of the *smallest* micro-channel of Arkilic *et al.* ($1.33 \times 5000\,\mu$m) [23] would require over a million cells and, consequently, at least ten million particles. In addition, the time steps would be only 36 ps, causing unsteady cases, even those with microsecond time scales, to be very expensive.

In response to this situation, the consequences of relaxing the cell size and time step requirements commonly placed on DSMC are explored in this chapter. The goal of this effort is to allow the cells in these calculations to be sized by the flow gradients, as in continuum CFD, rather than by the molecular scales. This is a complex issue, however, because this sizing violates some of DSMC's underlying assumptions as the scale length of the gradients increases, reducing the Knudsen number. Certain features of the physical situation are therefore altered or lost. The key to a successful simulation, and the goal of this chapter, is to identify these features and assess their

[1]The author is indebted to David Gonzales, a co-author of the paper (AIAA-95-2088) on which this chapter was based, for allowing the material to be presented here.

importance to the flow.

4.1 Scaling Rules

As discussed in Section 1.4.3, the cell size in a DSMC calculation should be comparable to the mean-free-path of the gas so collision partners may be selected without regard to their position in the cell. In this chapter, it will be convenient to express this constraint in terms of a 'cell Knudsen number', Kn_c:

$$Kn_c = \frac{\lambda}{\Delta x_d} \geq 1 \tag{4.1}$$

where Δx_d is a typical cell dimension (again, the subscript "d" refers to *dimensional* quantities).

A second requirement (discussed in Section 1.4.2) is that the time step, Δt_d, be small compared to a characteristic time so particle movement and collisions may be decoupled:

$$\Delta t_d < \frac{\Delta x_d}{c'_m} \tag{4.2}$$

where c'_m is the most probable molecular speed.

Borrowing from traditional computational mechanics, this constraint may be conveniently expressed as a 'CFL', or Courant-Friedrich-Lewy condition:

$$\frac{c'_m \Delta t_d}{\Delta x_d} < 1. \tag{4.3}$$

Physically, satisfying this condition requires a particle to reside in the same cell for a few time-steps to provide it with ample opportunity to interact with other particles. This ensures that its information can be distributed properly through the computational domain. It is important to realize that, unlike its CFD counterpart, this CFL condition is not a stability requirement, but a validity requirement. Violation of the condition will still yield results, but they may be inaccurate. A DSMC computation must therefore be constructed with very careful attention to its scaling constraints because the algorithm itself will give no indication that its results are completely non-physical.

Under the NTC collision method outlined in Section 1.4.3, the number of collision pairs, N_p, to be considered in a given cell is computed through the equation:

$$N_p \propto \Delta t \frac{N_c^2}{\tilde{n} V_c} \tag{4.4}$$

where V_c is the normalized cell volume, N_c is the number of computational particles in the cell, and $\tilde{n}$ is the reference simulation number density, which is the total number of computational particles divided by the non-dimensional domain volume.

Noting that the normalized cell volume scales like ${Kn_c}^{-3}$ and the simulation number density scales like $N_c {Kn_c}^3$, it may be concluded that:

$$\frac{N_p}{N_c} \propto \Delta t. \tag{4.5}$$

Thus, for a fixed size computation (N_c held constant), N_p depends only on the time step chosen to advance the solution.

The ratio of potential collision pairs to the number of particles in a cell will be referred to as the 'over-collision ratio'. A value of one implies that, on average, each particle will considered for a collision during every time step. In a well-resolved DSMC computation, where $\Delta t \approx 0.2$, roughly 20% of a given cell's particles will be considered for collisions.

The preceding expressions reveal the essential scaling issues involved in applying DSMC methods to low-Kn flows: as the physical size of the problem increases, it becomes computationally impractical to maintain a cell size on the order of the mean-free-path. It must therefore increase or, using the terminology introduced above, Kn_c must decrease, violating the constraint of Equation 4.1. In addition, as the cell size increases, the time step should be reconsidered. Two options exist: scale Δt so the CFL number remains constant, or hold it at some small value and let the CFL number decrease.

The first option maintains proper information propagation across the domain, preserving computational efficiency. Unfortunately, increasing Δt also causes rapid growth in the number of collision pairs to be considered, resulting in an over-collision ratio greater than one, as well as a large number of computationally expensive collisions to process.

The second option maintains a reasonable value of N_p/N_c, but results in a very small CFL number, causing an inefficient advancement of the solution and accumulation of statistics. This inefficiency springs from the fact that a small CFL number implies that most particles will require many time steps to cross a given cell. The collision phase of each time step will then involve essentially the same group of particles as the previous time step because very few particles leave or enter the cell. This situation is therefore equivalent to using a large time step but adding the computationally pointless exercise of moving the particles some small distance at several points in the collision phase. It may therefore be argued that it *only* makes sense to maintain a constant CFL number, regardless of the cell Knudsen number.

A solution to this problem was proposed by Bartel *et al.*[25], who recognized that many of the large number of collisions called for by Equation 4.4 serve no purpose other than to reinforce a Maxwellian distribution amongst the particles in a given cell. It should therefore be sufficient to restrict the number of collisions to some small value (comparable to the number of particles in the cell) and still take large time steps during the computation. In the current terms, Bartel's suggestion was to limit the over-collision ratio to some (arbitrary) value less than its "true" value given by Equation 4.4. This approach was applied by Bartel *et al.* to a Couette flow and an expanding nozzle flow with good results.

4.2 Numerical Investigation

To explore these issues in detail, two canonical cases were run with the current code: a Rayleigh flow and a free shear layer. These cases were chosen for their simplicity, the existence of either analytical or published solutions, and their relatively small CPU time requirements. Ni's bump was also run to confirm assertions made in the course of this work.

4.2.1 Rayleigh Flow

One-dimensional, unsteady Rayleigh flow was chosen as the first model problem for this investigation. This flow, illustrated in Figure 4-1, consists of a stationary gas subjected to the sudden acceleration of its lower boundary to a constant speed, U_o. It is well-suited to this investigation because its length scales are not imposed by geometry, but rather by time and viscosity. In addition, it is a one-dimensional flow so computations are compact and run quickly, allowing several test cases to be considered.

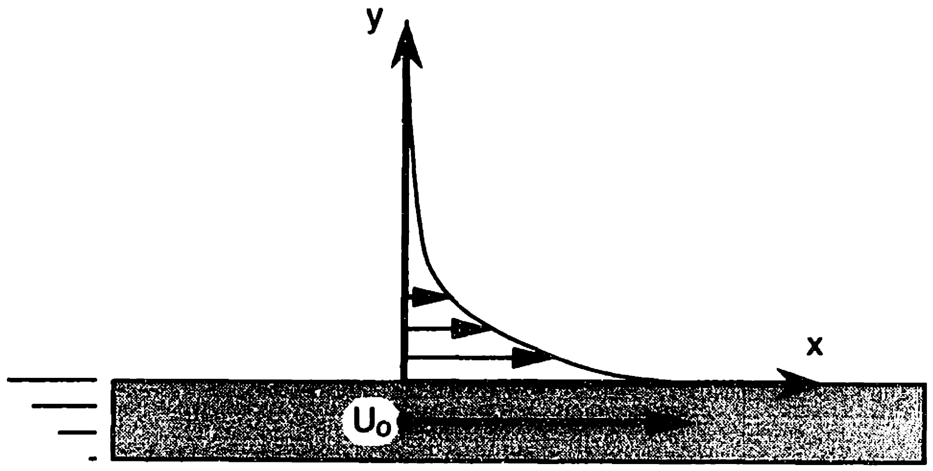

Figure 4-1: Schematic of a One-Dimensional Rayleigh Flow

For this series of calculations, the wall velocity, U_o, was set to 0.2 ($Ma = 0.22$) in order to ensure the applicability of the incompressible Navier-Stokes solution (given below). All cases were run with Helium ($\omega = 0.20$) and the CFL number was held at 0.3.

Analytical Solution

The analytical solution of the Rayleigh problem has two distinct regimes based on the Knudsen number. For large Kn, the collisionless Boltzmann equation is applied.

The resulting solution is given by [9]:

$$u = \frac{U_o}{2}\text{erfc}\left(\frac{y}{\sqrt{2RTt}}\right) \tag{4.6}$$

where R is the gas constant, T is the temperature, and erfc() is the complimentary error function.

For small Kn, the incompressible Navier-Stokes equation is applied. For the Rayleigh problem, this reduces to:

$$\frac{\partial u}{\partial t} = \nu\frac{\partial^2 u}{\partial y^2} \tag{4.7}$$

where ν is the kinematic viscosity coefficient, which is proportional to the mean free path.

This expression can be solved using the slip-flow boundary condition of Equation 3.1, resulting in:

$$u = U_o\frac{\text{erfc}(\xi)}{Kn+1} \tag{4.8}$$

where ξ is a similarity variable:

$$\xi = \frac{y}{2\sqrt{\nu t}}, \tag{4.9}$$

and Kn is the Knudsen number, defined here in a somewhat unusual, but convenient manner as:

$$Kn = \frac{\lambda}{\sqrt{\nu\pi t}}. \tag{4.10}$$

This is a more general version of the classical Rayleigh solution.[26] Note that, in this solution, Kn is a function of time and becomes infinite as $t \to 0$. This makes intuitive sense because the characteristic scale of the solution is the momentum thickness of the viscous layer, which is initially zero, but grows with time. One implication of this time-dependent Knudsen number, however, is that care must be taken in evaluating the solution at $\xi = 0$ and ∞, where appropriate limits of both t and y must be computed.

Well-Resolved Computation

This section presents DSMC results for well-resolved cases (i.e. where $Kn_c \geq 1$). These are intended to demonstrate the resolution of the current grid, which was used for all Rayleigh flow cases, as well as to serve as points of comparison for later sections, where the geometries are not fully resolved and scaling issues are important.

The first of these results, Figure 4-2, shows the computed near-wall velocity profile, $u(y)$, at $t = 0.004$. Because this time is significantly smaller than the mean collision time ($\sqrt{\pi}/2$, under the non-dimensionalizations of Section 2.1), the collisionless Boltzmann solution (Equation 4.6) applies. The solid line represents the analytic solution while the symbols represent the DSMC result. For this computation, the cell Knudsen number was 1.3×10^4 and the time-step 2.5×10^{-5}. The agreement is excellent, providing evidence that the grid is sufficiently dense to accurately resolve the profile.

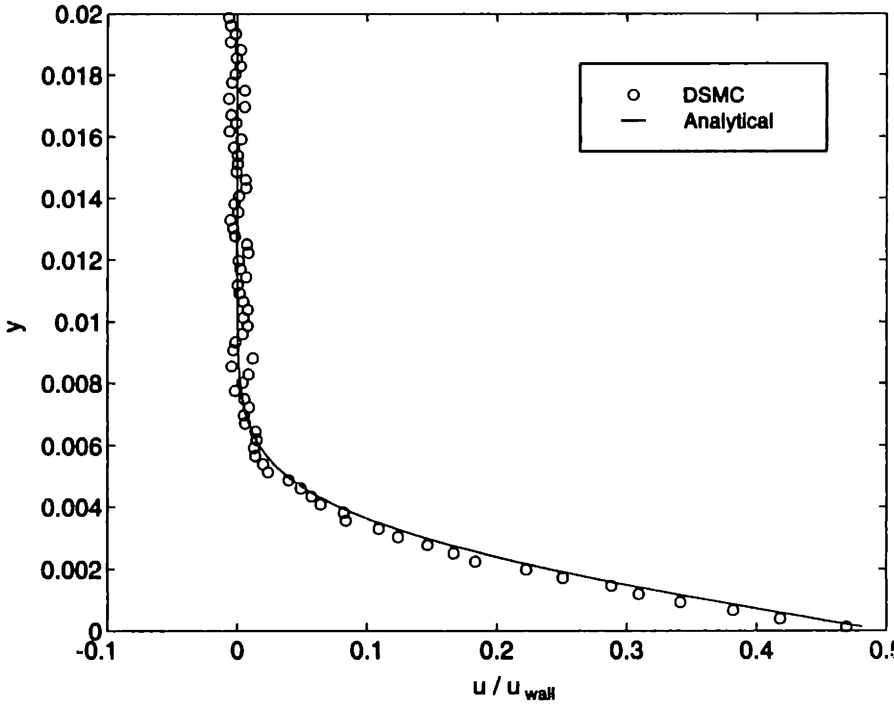

Figure 4-2: Comparison of DSMC and collisionless Boltzmann solutions to the Rayleigh problem at $t = 0.004$.

As the boundary layer grows with time, Kn increases through the point where collisions become important and the fluid begins to behave as a continuum. When

Kn leaves the transition regime (defined in Section 1.2 as $0.1 < Kn < 3$), the slip-flow Navier-Stokes solution of Equation 4.8 becomes applicable. Figure 4-3 contains a comparison of DSMC results with this solution at $t = 100$. It may be noted that $Kn = 0.07$, so there is still a perceptible velocity slip at the wall (ie. $u(0) < U_o$).

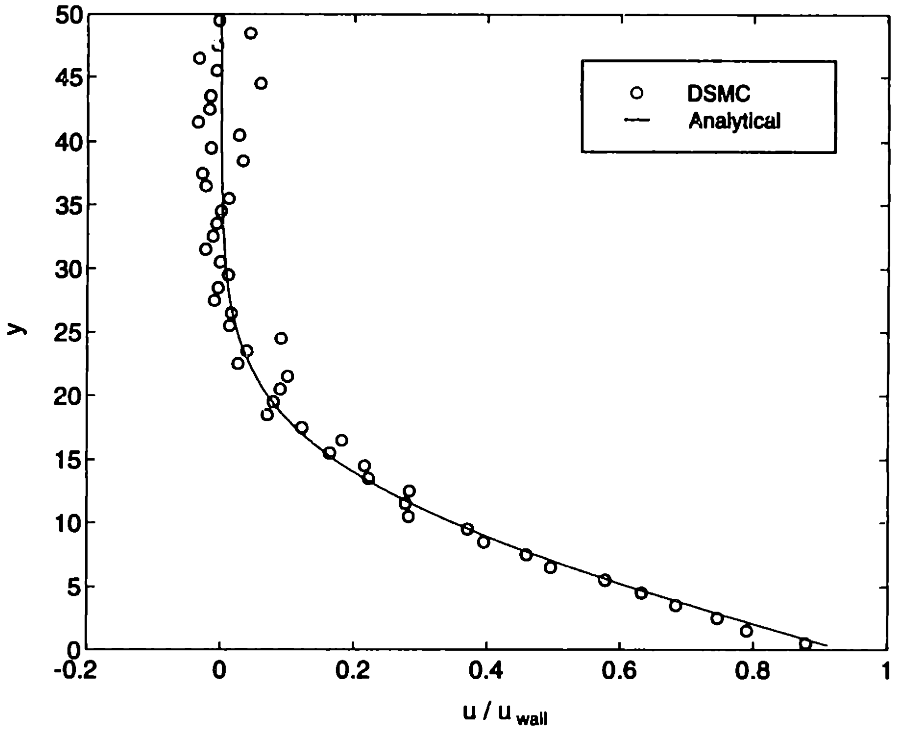

Figure 4-3: Comparison of DSMC and slip-flow Navier-Stokes solutions to the Rayleigh problem at $t = 100$.

If, as Figure 4-3 suggests, DSMC is correctly solving the Navier-Stokes equations, the velocity distribution should be a slight perturbation from Maxwellian, given by the Chapman-Enskog distribution for a one-dimensional isothermal shear flow [1] as:

$$f(u', v') = f_o \left(1 - \frac{\nu u'v'}{(RT)^2}\frac{\partial u}{\partial y}\right) \tag{4.11}$$

where f_o is the Maxwellian distribution. This distribution was computed and is shown in figure 4-4, sampled at $y = 2.33$, $t = 40$. Here, the mean velocity, u, has been subtracted and the four quadrants of (u', v') have been collapsed onto one through algebraic operations which reinforce the perturbation by taking advantage of its anti-symmetry. It should be noted that the Chapman-Enskog distribution is formally only valid for small perturbations. Because this is a low Mach number computation, the thermal fluctuations are significant and therefore only qualitative comparisons are appropriate. Nevertheless, the deviation from Maxwellian computed in the DSMC simulation is in good agreement with its predicted structure, indicating, as expected, that the gas is weakly perturbed from equilibrium by the shear.

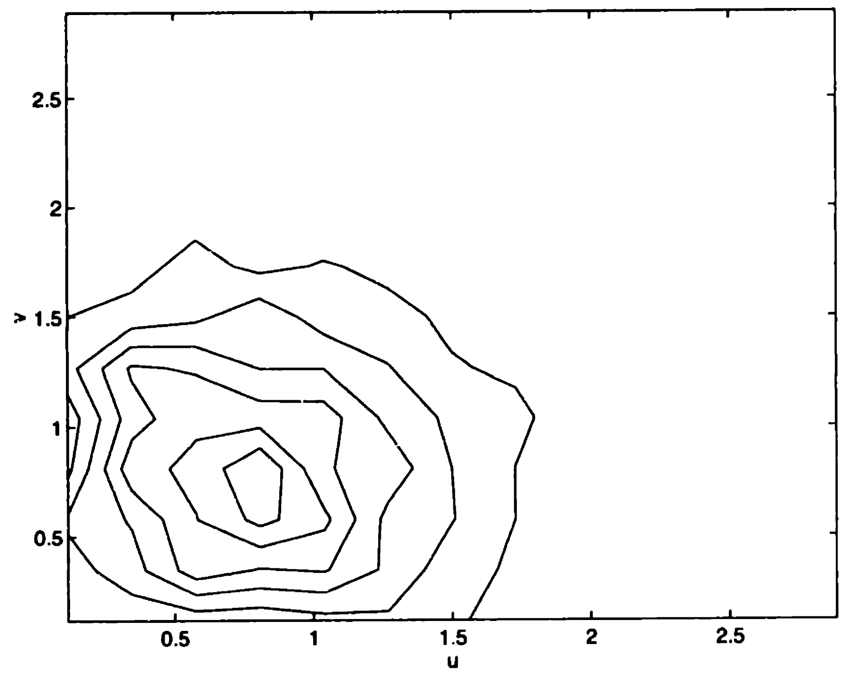

Figure 4-4: Distribution of velocity perturbations for a well-resolved computation.

In the following section, where this case is recomputed with various values of Kn_c, a quantitative comparison of the shear layers will be needed. One such a measure is obtained by performing a non-linear least-squares fit of the streamwise velocity data to the profile of Equation 4.8, with ν as the free parameter. To demonstrate the dependability of this measure, Figure 4-5 shows the time-dependence of the computed viscosity for the well-resolved computation as it evolves from $t = 0$ to 200.

With the non-dimensionalizations used in the code, the normalized kinematic viscosity coefficient of Helium is 0.64. From Figure 4-5, it may be seen that the DSMC solution under-predicts this viscosity for small times, but appears to asymptote to nearly the correct value as $Kn \to 0$. The under-prediction at early times may be explained by noting that Kn is not yet in the region of validity for Equation 4.8. The slight over-prediction at later times is most likely due to a variety of factors including

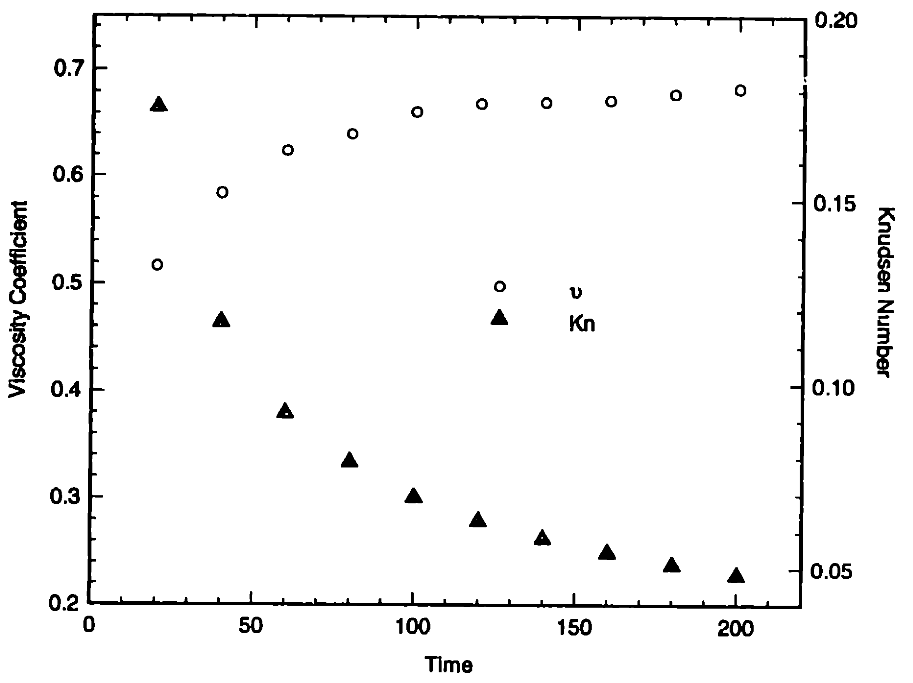

Figure 4-5: DSMC-computed viscosity and Knudsen number as a function of time for a well-resolved computation.

statistical scatter due to the low-Mach number and, a weak influence of the far wall (located 200 λ from the moving surface).

Under-Resolved Computations

The previous section demonstrated that the DSMC code can accurately reproduce the analytical solution of the Rayleigh problem. This is not surprising, however, because the method's ability to model the true gas physics was demonstrated in the previous chapter and by many other researchers. These results were presented mainly as a means of verifying the proper matching of the code and grid to the problem and for calibrating the measures used in the scaling investigation.

This investigation is begun in the current section by manipulating the previous Rayleigh calculation. In all cases, the grid geometry was maintained, but it's linear dimensions were increased by a multiplicative factor. To hold the CFL number constant[2], this factor was also applied to the time step. The number of particles was

[2]Selected computations performed with lower CFL numbers yielded almost identical results, supporting the assertions made in Section 4.1.

not changed.

In a scaled calculation with a high over-collision ratio, it is expected, as argued above, that the particles in each cell will approach a Maxwellian distribution, rather than the proper, perturbed Maxwellian shown in Figure 4-4. This hypothesis was tested by running a case with an over-collision ratio of approximately 5 ($Kn_c = 0.05$). The distribution was then sampled from the same grid location as the previous case at a time selected to obtain approximately the same point in the profile (ie. the same value of $u(y)/U_o$). With the current scaling, this corresponds to $y = 46.67$ and $t = 2700$. The resulting distribution is shown in Figure 4-6, collapsed onto one quadrant in the same manner used in Figure 4-4.

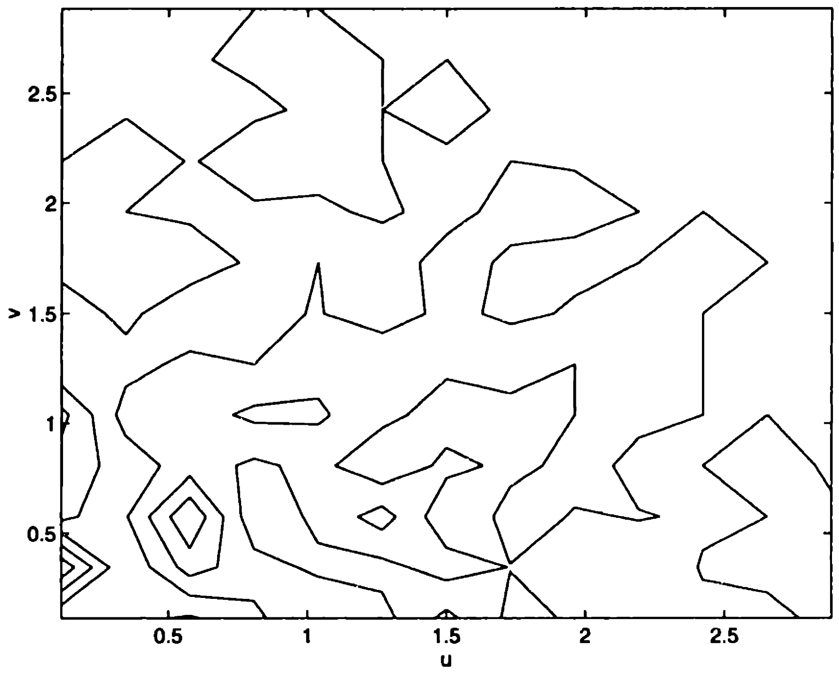

Figure 4-6: Distribution of velocity perturbations for an over-collided (under-resolved) computation

In contrast to the previous case, there is no well-defined structure to the perturbations in Figure 4-6. This lack of structure may be quantified by noting that the integral of the collapsed distribution shown here is an order of magnitude smaller than that for the well-resolved case. The implications of this result are somewhat subtle: if each cell in the over-collided computation contains an equilibrium distribution, then the Navier-Stokes equations, which correspond to a weak perturbation from the Maxwellian state, are not being solved, but rather the Euler equations, which represent an ideal gas in a perpetual state of equilibrium. Indeed, the closer each cell is driven to equilibrium, the more "ideal" the fluid should become. If this is in fact true, one could argue, then the computed viscosity of the Rayleigh layer should go to zero as the cell Knudsen number decreases, confining the effect of the wall motion to an increasingly narrow layer which asymptotes to a vortex-sheet singularity at $y = 0$.

When this assertion is tested through a series of runs with different grid scalings, however, a surprising result is found: instead of a decrease in viscosity as $Kn_c \to 0$, the opposite occurs. As Figure 4-7 indicates, the gas viscosity calculated from the DSMC computations *increases* as a function of the cell size ($\Delta x = 1/Kn_c$).

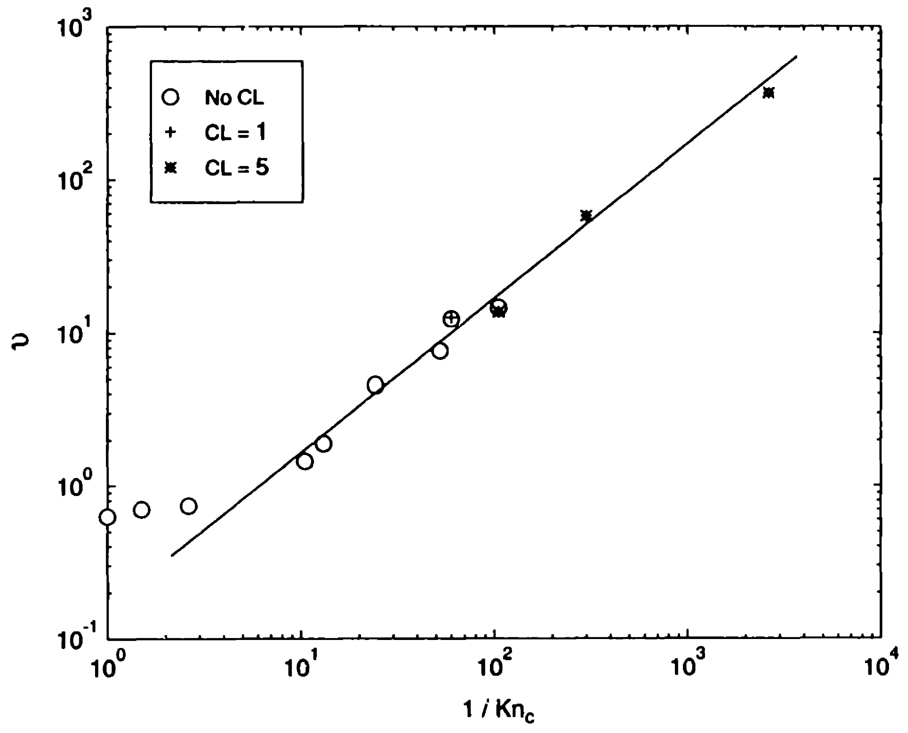

Figure 4-7: DSMC-computed viscosity versus cell size.

Three sets of data are reported in Figure 4-7: in the first set, denoted by circles, the number of collisions computed during each time step was not limited; in the second set, denoted by stars, a collision limiter of 5 was enforced; in the last set, denoted by crosses, a collision limiter of 1 was employed. (A collision limiter of 1 implies that $N_p = N_c$ regardless of the value of called for by the NTC equation (Equation 4.4).)

At low values of $1/Kn_c$, the correct ν is produced. However, as the cell size rises above about 10, the effective viscosity begins to increase. This increase appears to be linear over a very wide range of Kn_c.

This behavior stems from the particulate nature of the simulation. At low Mach numbers, the random thermal velocity of the gas will always result in particles moving in the y-direction even though there is no net motion normal to the surface. In addition, because DSMC chooses collision partners in a given cell without regard to their location, momentum is uniformly diffused through the cell within a few time-steps. In a properly-resolved computation, where the cell dimension is approximately one mean-free-path, this behavior is molecularly "correct" and results in the proper physical viscosity of the fluid (helped, of course, by an appropriate value of the VHS exponent, ω). However, in an under-resolved computation, the disregard for a particle's location in the cell results in a diffusion of momentum due to thermal motion that is far in excess of the physical viscosity and it is this artificial viscosity that is exhibited in these results. In many cases, this thermal motion would not present a problem. However in the presence of a strong mean velocity gradient (as in the case of the wall-driven Rayleigh problem), the thermal motion, coupled with the mean shear, results in an artificial Newtonian viscosity, inversely proportional to the cell Knudsen number, Kn_c, and greater than the physical viscosity.

This argument suggests that the artificial viscosity should be most apparent at low Mach number and will abate as Ma increases because the relative importance of thermal fluctuations is reduced. In other words, at high speeds, particles will be swept from the domain by the mean flow before they have the opportunity to diffuse appreciably in the transverse direction. This was not found to be the case for the Rayleigh problem, however. The apparent discrepancy is explained by noting the flow's one-dimensional nature, which was exploited in this calculation by employing a 'wrap-around' boundary condition. Here, exiting particles are reintroduced to the domain on the opposite edge, where they can continue to diffuse in the y-direction. A 2-D, spatially-developing flow is therefore needed to verify the above argument.

4.2.2 Boundary-Free Shear Flows

A boundary-free shear flow was selected to test the above assertion. Here, two streams at different velocities are introduced at $x = 0$; the upper stream moving at U_1, and the lower at U_2. This arrangement is illustrated in Figure 4-8.

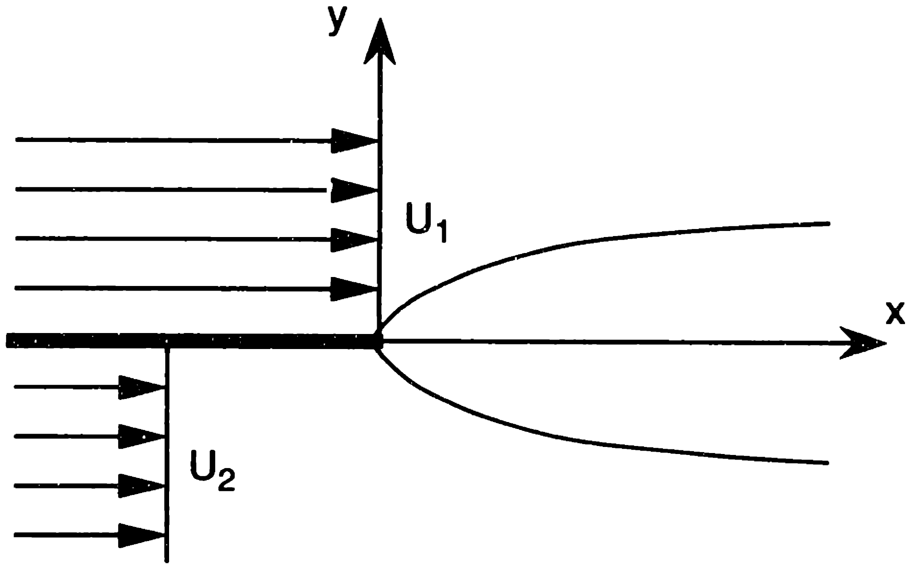

Figure 4-8: Schematic of boundary-free shear flow computational geometry.

Figure 4-9 shows contours of velocity in the $x-y$ plane for a well-resolved calculation where $U_1 = 1.5$ ($Ma = 1.6$), $U_2 = 2.0$ ($Ma = 2.2$), and the working fluid is Argon ($\omega = 0.31$). Note that the singularity at $x = 0$ quickly diffuses and a slowly thickening shear layer develops.

By integrating the streamwise velocity across the layer, a flow can be characterized by its momentum thickness, θ:

$$\theta = \frac{1}{HU_1^2}\int_{-\infty}^{+\infty}(U_1 - u)(u - U_2)dy \qquad (4.12)$$

where θ has been normalized by the upper stream velocity, U_1 and the computational domain height, H. This momentum thickness is shown in Figure 4-10 as a function of x for several computations at different Kn_c.

Theoretically, the spatial evolution of a laminar shear layer shows a square-root

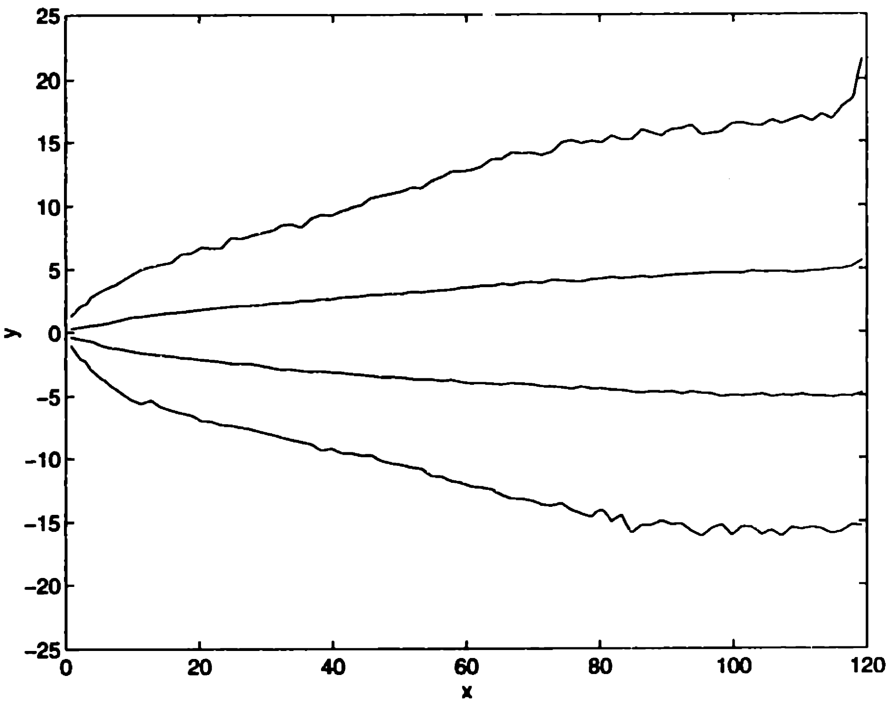

Figure 4-9: Contours of streamwise velocity in a well-resolved DSMC computation of a supersonic shear flow.

growth in momentum thickness [26]:

$$\theta \propto \sqrt{x} \tag{4.13}$$

which is well captured by all of the test cases. Anamolies are observed, however, when comparing these cases to one another.

First, comparing the well-resolved case (120×60) to the first scaled case (2400× 1200), it is expected that the twenty-fold increase in linear dimensions will result in a $\sqrt{20} = 4.47$ decrease in the shear layer thickness at the same value of x/H. In Figure 4-10, however, the shear layer only decreases by a factor of approximately 1.3. This is again explained by the artificial viscosity introduced by the under-resolved grid: as the grid is scaled from the well-resolved case, this artificial viscosity surpasses the physical viscosity, altering the proper relationship between the curves because the cases appear to be run with different fluids.

Second, a further doubling from a scale factor of 20 to 40 (relative to the baseline case) is expected to further reduce the shear layer thickness, but was found

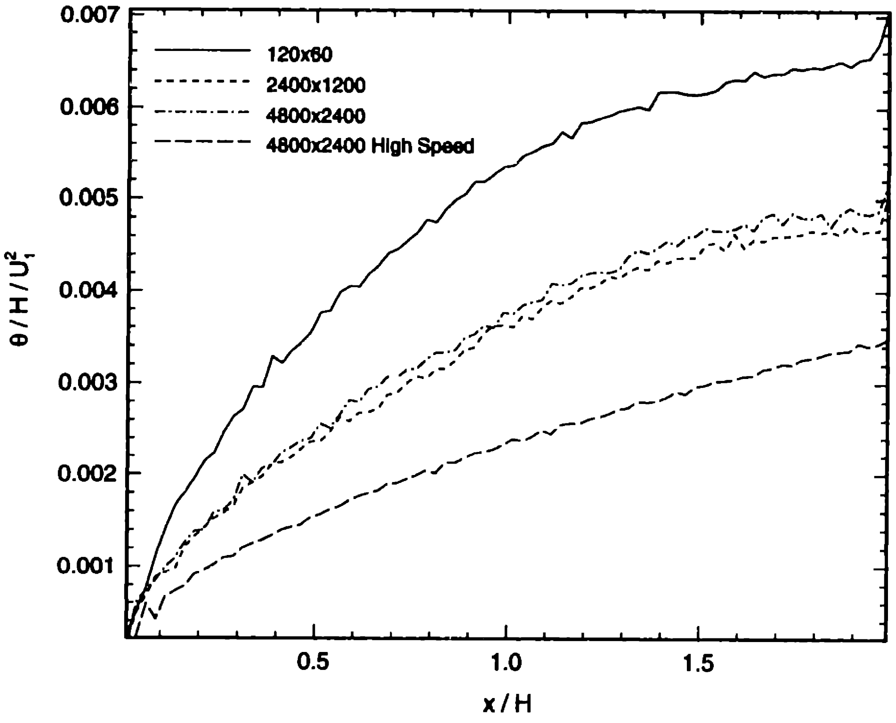

Figure 4-10: Development of shear layer momentum thickness as a function of downstream distance.

to have a negligible influence. This "saturation" effect is related to the particle-transport/collisional-smearing mechanism causing the artificial viscosity. Because the grid dimensions and time step were scaled by the same factor (ie. the CFL number was held constant) identical particles moving through the two cases will pass through the same locations on the grid and undergo the same number of collision phases. In other words, for this Mach number, a characteristic levei of artificial viscosity has been reached which is dependent on the grid itself. For these scaling factors, this grid-based viscosity is so much greater than the physical viscosity that the flow has become insensitive to its physical dimensions.

Finally, when the Mach number of the incoming flow is increased, keeping the ratio U_1/U_2 constant, the theory predicts no change in the shear layer thickness. The trace in Figure 4-10 for this case (labelled '4800×2400 High Speed) shows a thinner shear layer, however (still not reaching its correct value, but closer nonetheless). This improvement is due to the lower importance of thermal velocities with respect to the stream velocity, which results in a lower artificial viscosity. It should be noted that, as in the Rayleigh problem presented in the previous section, the artificial viscosity was observed to be relatively insensitive to the use of a collision limiter and to the choice of the CFL number. In this geometry, however, it was observed that a collision limiter did *thin* the layer very slightly. This is perhaps due to the fact that, with a collision limiter, there is slightly less "uniformity" within each cell because there are fewer collisions, so momentum is not smeared across the shear layer as rapidly.

4.2.3 Euler Flow

The results of the preceding section illustrate two competing trends in DSMC computations as the cell Knudsen number decreases. On one hand, a high value of the over-collision ratio, N_p/N_c, drives the particles in each cell toward a Maxwellian, or equilibrium, distribution. If this were the only effect present, DSMC would therefore, in these cases, formally solve the Euler equations, which are the governing dynamic equations for an equilibrium gas. A competing effect exists, however, because the essential nature of DSMC - the random motion of particles coupled with a disregard for the location of collision partners in a cell - creates an artificial viscosity from the diffusion of momentum through the domain.

From this, one might expect that a scaled DSMC code should fail most dramatically when attempting a low Mach number viscous flow and be optimal for modeling a high Mach number, inviscid flow. The former assertion was demonstrated in the previous sections; this section presents results to confirm the latter.

The test case selected for this task, known as "Ni's Bump", is often run to demonstrate inviscid CFD methods.[27] The geometry consists of a two-dimensional duct with a circular-arc excrescence placed on the floor. A bump with 20% of the duct height was chosen for this work. This geometry and a traditional Euler code solution are shown in Figure 4-11. The inlet Mach number is 3.28 and the working fluid is Argon.

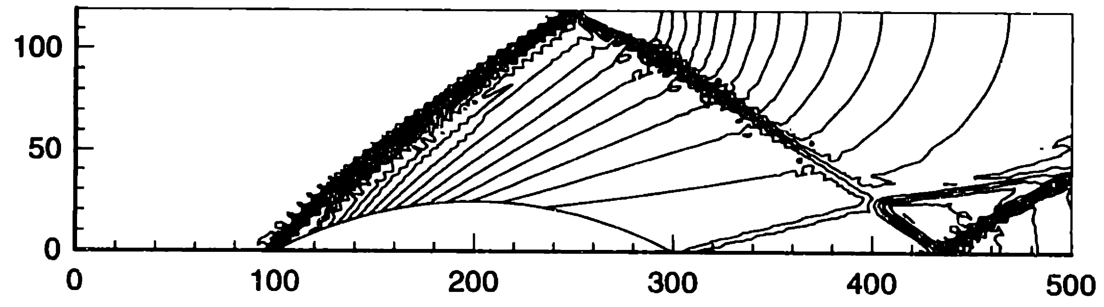

Figure 4-11: Mach number contours of an Euler solution to Ni's Bump.

Results

Figure 4-12 shows Mach Number contours for a well-resolved DSMC computation of Ni's bump with duct dimensions of $125 \times 30\ \lambda$. This run utilized about 42,000 particles on a grid of 1,860 cells with $\omega = 0.31$. The flow reached statistical equilibrium in approximately 10 minutes, but an additional two hours were required to gather adequate statistics for data analysis (on an SGI Indigo). Although specular walls were employed, and thus a viscous solution is not being attempted, it is instructive to note that the effective Reynolds number of the DSMC computation, based on the inlet Mach number and the bump length (which is dimensional due to the mean free path grid scaling), is 243.

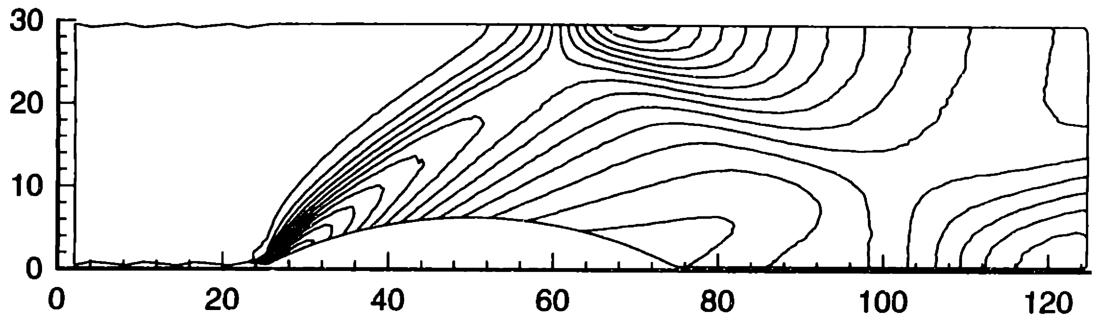

Figure 4-12: Mach number contours of a well-resolved DSMC computation of Ni's Bump.

By comparing this solution to that of the Euler code, it may be concluded that the flow is faithfully reproduced except for an apparently large shock thickness in the DSMC result. This is to be expected since, although there are no viscous effects due to the boundaries, the DSMC method is accurately solving the shock structure, where viscous effects are strong. Because the entire grid is only 125 mean free paths in

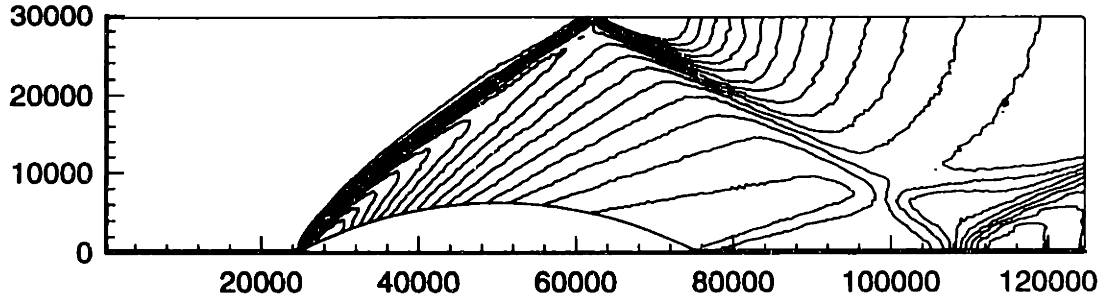

Figure 4-13: Mach number contours of a scaled DSMC computation of Ni's bump.

extent, however, this shock appears excessively thick, though its thickness is actually quite comparable to the physical case ($\approx 10\,\lambda$). In addition, other features of the flow, including the shock angles, the leading and trailing edge shocks, the shock reflections and the shock-shock interaction toward the rear of the duct, are properly reproduced.

Figure 4-13 shows the same computation, scaled by a factor of 1,000 so the dimensions of the duct are now $125,000 \times 30,000\,\lambda$ (approximately 9×2 millimeters at atmospheric conditions). For this case, a collision limiter of one was used. The grid resolution was also doubled in each direction (compared to the last case) to 7,750 cells. This was necessary because the shock thickness was observed to asymptote to a larger-than-expected minimum value as the cell Knudsen number decreased. This was identified as a grid resolution issue and it was found that, in common with standard CFD practice, the computed flow converged to a grid-independent solution through successive refinements in the grid (the average number of particles per cell was held constant).

When compared with the previous case, the shock appears to be much thinner, although it is now actually thicker than a physical shock due to the much larger grid dimensions. This large shock thickness is due to the artificial viscosity inherent in a continuum application of DSMC. This does not render the solution invalid, however; it is, in fact, common practice in inviscid continuum CFD codes to explicitly introduce an artificial viscosity (usually second-order) in order to stabilize the solution near steep gradients such as shocks. The effect of this artificial viscosity is, as in the DSMC solution, a thickening of the shocks. DSMC does not require any explicit introduction of artificial viscosity, however; it provides its own, as demonstrated by these results. Additionally, no oscillations in the solution are observed in the regions surrounding the shock.

Chapter 5

Conclusions

This thesis has presented many aspects of DSMC's application to micromechanical devices. In the preceding chapters, the motivation behind this application, the algorithm developed for it, some of its results, and the issues involved with scaling it beyond its traditional range were discussed. This chapter draws conclusions from these efforts to evaluate DSMC's potential as a tool for MEMS development.

As discussed in Chapter 1, the high Knudsen numbers of many flows of interest to MEMS designers make them inaccessible to continuum-based numerical methods. While analytical models are being developed for moderately-rarefied flows in simple geometries [23], the complexity and mixed-regime nature of many MEMS limits the applicability of these methods. Experimental study of these flows has also proven to be difficult due to the miniscule quantities which must be measured. In the course of their micro-channel investigation, for example, Arkilic *et al.* [23] were required to develop a system capable of measuring mass flows on the order of nanograms per second. In response to this situation, Chapter 3 presented a small subset of the micro-flows of interest, to both the fluid-dynamicist and the MEMS designer, which are easily treated with DSMC.

In Section 3.1, DSMC results for a slip-flow regime micro-channel were compared to an analytical solution presented by Arkilic *et al.*[23] Several aspects of the computed and theoretical streamwise velocity and pressure distributions were considered. In all cases, the numerical results were found to compare well with their analytical predictions. In addition, by manipulating the analytical relations, an expression for a streamwise velocity 'similarity' profile was developed. It was subsequently found that the computed velocity distribution, which is a function of x, collapsed quite well to the predicted x-independent profile when the flow variables were processed according to the theoretical expression.

These findings support the accuracy of both the code and the analytical work and demonstrate another valuable function of DSMC computations: the evaluation of analytical models for rarefied flow behavior. Developing these models is an important task because a great number of MEMS geometries fall into the slip-flow regime. Analytical tools have great value to the designer because they are much less expensive than a full DSMC calculation but are able, for certain geometries and flow regimes, to quantify behavior invisible to continuum techniques. They are difficult to validate, however, because, as mentioned above, the effects they describe are often too small for effective experimental investigation.

In Section 3.2, a DSMC solution for a transition regime micro-channel was compared to predictions from the slip-flow expressions of Arkilic *et al.* By considering the form of disagreement between these results, several aspects of flow behavior in this regime were illuminated. This is an especially important application of DSMC because very few techniques, either analytical or numerical, are available for analyzing these flows. This is, consequently, the least understood of the four Kn-regimes, posing a significant problem for the MEMS community because the geometries and working conditions for many devices place them here. Additionally, through the similarity analysis developed in the previous section, the onset of transition behavior was observed as the Knudsen number increased down the channel. This type of work is intended to determine the applicable range of various models and the mode of their failure. This is valuable because applying convenient models to the largest possible range of problems is preferable to performing large numerical simulations, especially early in the design process.

In the final section of Chapter 3, parabolic micro-nozzles were examined. These cases are intended to represent devices which contain complexities not amenable to other forms of solution. Their more complicated geometry, lack of isothermal flow, and sharp gradients pose serious problems for many types of analysis. No special considerations for these features were required to perform the DSMC calculation, however. This demonstrates the versatility of the method and its value for complex flows, which are common in MEMS devices and will become more so as the field matures.

Chapter 4 extended this work by investigating the issues inherent in scaling a DSMC code to treat continuum flows. It is important to understand these issues because many MEMS devices contain regions in this regime. Due to its molecular-level scaling, however, treating them with a fully-resolved DSMC calculation is very computationally expensive. A greater understanding of DSMC's behavior when its scaling constraints are relaxed is therefore valuable because it may enable the simplified treatment of certain geometries.

In this chapter, two primary effects were observed when the grid was scaled beyond the traditional constraints of DSMC calculations. The first of these was previously identified by Bartel *et al.* [25]: when the scaling reaches a point where particles undergo multiple collisions every time step, the velocity distribution in each cell approaches an equilibrium (Maxwellian) form. The implication of this, which was not previously realized, is that the DSMC technique is formally solving the Euler equations, not the Navier-Stokes equations, in these regions. In cases where this is appropriate, Bartel *et al.* note that the computation may be greatly accelerated by employing a 'collision limiter' large enough to enable each cell to reach equilibrium, but not so large that this state is needlessly reinforced through continued interaction.

A second implication of DSMC's application to continuum flows is that the thermal velocity of the computational particles, coupled with the physically large cell size, results in an artificial viscosity which is proportional to the cell Knudsen number and decreases with flow Mach number (because the thermal fluctuations become less important). This is a consequence of the DSMC collision algorithm, which tends to distribute momentum and energy uniformly through the entire cell due to its disregard for the position of colliding partners within that cell. This tendency is particularly strong when a large number of collisions are performed during each time step, as is the case when Kn_c is small. This artificial viscosity can be a nuisance when solving the Navier-Stokes equations at low Reynolds numbers, where the physical viscosity is important to the overall flow behavior, because it is implicit to the scheme and represents a first-order error term. Large reductions must therefore be made in the cell size to bring the computed viscosity acceptably close to the true value. This artificial viscosity may be an asset for solving Euler flows, however, because it acts as a built-in, self-adjusting means of broadening poorly-resolved flow features such as shock structures, serving a similar function as the artificial viscosity explicitly included in many traditional CFD methods.

In summary, the unique features of rarefied flows have many implications for MEMS designers. The larger mass flow rate for a given geometry and inlet conditions, found in Sections 3.1 and 3.2, must be considered when designing control systems for micro-chemical reactors and the decreased thermal communication between the fluid and its boundaries, demonstrated in Section 3.3, must be considered in applications involving temperature measuring devices or heaters, for example. These flows are not well-studied, however, due to the breakdown of the transport assumptions which characterize the Navier-Stokes equations, the basis of most common fluid-dynamic analysis tools.

DSMC's ability to calculate in any of the four Kn-regimes without modification makes it ideal for investigating flows related to MEMS devices, particularly those with regions in different regimes. It is therefore useful to MEMS researchers for validating analytical models, investigating flow behavior, and modeling potential devices. In addition, it is quite straightforward to include flow features such as chemical reactions, multi-species mixing and particle transport in the code due to its particulate nature and modular construction. In addition, inclusion of unstructured grid capability and the trajectory-tracing movement scheme enable the code to handle arbitrary geometries. This is a valuable asset when analyzing the complex structures included in many of these devices. Finally, its cell-based structure makes it well-suited for parallel computation, which is an increasingly important attribute for a large-scale numerical scheme on modern computers.

These attributes are especially important to consider in light of the scaling investigation performed in Chapter 4. Given the speed and maturity of continuum CFD methods, it appears, at first glance, that the scaling attempted in this chapter is irrelevant because, given the choice, the continuum method would always be selected. This is not necessarily true, however, because continuum methods are often more difficult to parallelize and complexities, such as those mentioned above, each represent large increases in computational work. It is therefore possible that the speed of a DSMC calculation in certain problems and on certain machines, may be quite comparable to that of a continuum technique. In addition, mixed regime flows are not generally treatable with continuum methods because they cannot effectively model the rarefied areas. While hybrid continuum/particulate codes are under development [28] for these flows, it would be desirable to treat the entire domain uniformly to maximize code flexibility and avoid, if possible, the problems associated with interfacing two techniques of entirely different natures.

Many aspects of DSMC's application to MEMS remain to be explored and developed. To model actual devices in their entirety, for example, the current code will require an upgrade to three-dimensional calculations. The corresponding increase in computational requirements will then make it necessary to move from the current, workstation-class, computers to large-scale supercomputers. As mentioned previously, DSMC's structure is very amenable to efficient parallelization, so this move will likely be to a parallel machine. On these architectures, many computational issues remain to be explored which promise to increase the speed and efficiency of the technique. In complex cases for which a particle description is an asset, such as multispecies and chemically reacting flows, this method may then be advantageous even in regimes to which continuum techniques are currently applied. For mixed-regime flows, hybrid DSMC-continuum techniques represent a promising area which still requires significant investigation. When mature, these techniques will allow both methods to be applied where they are most desirable and 'swapped out' where they become inefficient or inaccurate. A numerical solution will then involve applying the best of each discipline to a problem, rather than choosing one or the other when setting up the calculation and patching together corrections and modifications for certain portions of the flow.

Appendix A

Particle Movement Function

This appendix contains the function which performs particle movement. It is intended to serve only as an annotated listing; the full description of the movement operation is given in Section 2.4.

Arguments

The movement function takes four arguments. The first argument, 'cel' is a pointer to the current cell of the particle(s) to be moved. The second argument, 'particles' is a pointer to the first particle in a list of those to be moved. In the case of a routine global move, such as at the beginning of a time step, this is simply the cell's particle list. For single particle moves, such as when one is introduced at an inflow/outflow face or emitted from a diffusely-reflecting surface, this is simply the address of the particle to be displaced. The third argument, 'nparts' is simply the number of particles to be moved. The final argument 'sel_move_flag', is an integer constant that tells the function whether or not to read the movement times remaining for individual particles. When the flag is set to 0, all particles are considered to be moving an entire time step, as is the case in a global move.

Cell Face/Node Numbering Scheme

When calculating intersection times, a simple relation between the face and node indices is required. In order to preserve the trajectory-tracing method's ability to function in cells with an arbitrary number of faces, the following scheme was adopted: face i connects nodes i and $(i+1)\%n_s$ where i is an index which starts at zero, n_s is the total number of sides for the cell, and % is the modulus operator, which returns the remainder resulting from the division of its operands. The macro, 'IRING', that appears in this function performs this modulo-division operation to cycle through cell nodes. For the current, triangulated mesh, it is given by:

```
#define IRING(A) = ((A) % 3))
```

This numbering scheme is shown in Figure A-1.

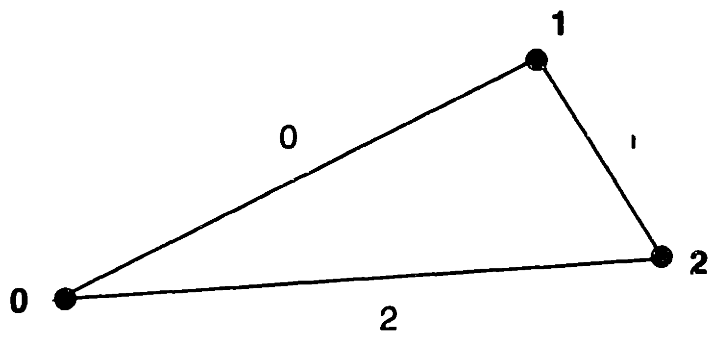

Figure A-1: Schematic of cell node and face numbering.

Code Listing

```
void Move_Particle(struct cell_unit *cel, struct atom *particles,
  unsigned short n_parts, short sel_move_flag)

    short fc, face, entry_face, y_intersect;
    float time_interval, t_int, t_min, traj_slope;
    unsigned curr_cell;
    struct atom *part;

    if(!sel_move_flag){

        /**** Set particle movement time interval to be the time step ****/
        time_interval = T_STEP;

        /**** Set the entry face to be a nonexistant side ****/
        entry_face = 3;
```

```
}
for (part = particles; part < particles + n_parts; part++){
    /**** Skip if this is an empty particle position ****/
    if (part->dest_cell == 0) continue;

    /**** If particle has somehow left grid, bring it back ****/
    if (part->x > x_max+0.1 || part->y > y_max+0.1 || part->x <-0.1
        || part->y<-0.1){
        printf("Relocating Particle at t = %f:\n",t_global);
        printf("  x:  %f y:  %f u:  %.24f v:  %.24f cell:  %d particle:  %d\n",
            part->x, part->y, part->u, part->v, cel - cell, part - particles);
        Random_Plcmt2(cel, part);
    }

    if(sel_move_flag){

        /**** Determine time interval over which to attempt to move particle ****/
        time_interval = part->mvmt_time;

        /**** Skip if this particle is finished moving ****/
        if (time_interval == 0.) continue;
    }

    /**** Determine which face is ineligible for intersection because it was
      just crossed (selective moves only) ****/
    if(sel_move_flag)
        entry_face = (short)part->crosd_face;

    /**** Calculate the slope of the particle trajectory ****/
    if(part->u == 0.)
        traj_slope = INFINITY;
    else
        traj_slope = part->v / part->u;

    if (traj_slope * traj_slope > 1.)
        y_intersect = 1;
    else
        y_intersect = 0;

    /**** Initialize t_min with a large value ****/
    t_min = 1.05 * T_STEP;

    for (fc = 0; fc < 3; fc++){
        if (fc == entry_face) continue;

        /**** If particle path is parallel to face, they will never intersect ****/
        if(cel->slope[fc] == traj_slope) continue;

        /**** If particle's vertical component is greater than its horizontal (in
          absolute value), use y-intersects for time determination ****/
        if(y_intersect)
            t_int = (((traj_slope *cel->slope[fc] *(cel->x[fc] - part->x)
              + cel->slope[fc] *part->y -traj_slope *cel->y[fc])
              / (cel->slope[fc] -traj_slope)) - part->y) /part->v;
        else
            t_int = (((cel->y[fc] - part->y + traj_slope*part->x
              - cel->slope[fc]*cel->x[fc]) / (traj_slope - cel->slope[fc]))
```

```
                - part->x) / part->u;
        /**** If t_int<0 then particle is heading the wrong way to hit this face ****/
            if (t_int < -I_TEST) continue;

        /**** If the intersect time is zero, make sure this particle is heading
          in a direction which makes striking this face possible before accepting
          it to avoid an infinite loop of zero time steps ****/
         if (t_int <= I_TEST && ((part->y + part->v - cel->y[fc])
           * (cel->x[IRING(fc+1)] -cel->x[fc]) -(part->x + part->u
           - cel->x[fc]) *(cel->y[IRING(fc+1)] -cel->y[fc]))
           / ((cel->y[IRING(fc+2)] -cel->y[fc])*(cel->x[IRING(fc+1)]
           - cel->x[fc]) - (cel->x[IRING(fc+2)] -cel->x[fc])
           * (cel->y[IRING(fc+1)] -cel->y[fc])) >= 0.0) continue;

         if (t_int < 0) t_int = 0;

        /**** If this is the minimum time so far, record it and current face ****/
         if (t_int < t_min){
            t_min = t_int;
            face = fc;
         }
      }

    /**** If the time to collide with all faces is greater than time remaining
      for movement, displace particle along trajectory and mark it as maintaining
      its current cell ****/
     if (t_min >= time_interval){
         part->x += part->u * time_interval;
         part->y += part->v * time_interval;
         part->dest_cell = 1;
         part->mvmt_time = 0.;
     }

    /**** If the min time for collision is smaller than remaining time, particle
      has hit something, determine what that was and act appropriately ****/
     else{
         if (cel->nbr[face] == SOLID_BOUNDARY){

            /**** If a solid boundary was contacted, displace particle to boundary
              and call function to process reflection ****/
             part->x += part->u * t_min;
             part->y += part->v * t_min;

             part->mvmt_time = time_interval - t_min;
             part->crosd_face = (char)face;
             Solid_Boundary(part, cel, face);
         }
         else if (cel->nbr[face] == INFLOW_OUTFLOW){
#ifdef IO_BOUNDARY
            /**** If particle left grid, mark its place as empty and decrement
              the number of particles in this cell ****/
             part->dest_cell = 0;
             cel->n_particles--;
             cel->n_vacancies++;
#else
            /**** If we're not doing inflow/outflow, make these bounds solid ****/
```

```
                part->x += part->u * t_min;
                part->y += part->v * t_min;
                part->mvmt_time = time_interval - t_min;
                part->crosd_face = (char)face;
                Solid_Boundary(part, cel, face);
#endif
            }
            else{
              /* If it didn't hit a boundary of some sort, particle has crossed
                 into another cell */

              /**** Displace particle to boundary ****/
                part->x += part->u * t_min;
                part->y += part->v * t_min;

              /**** Increment the current cell traveler counter ****/
                cel->n_tvlrs++;

              /**** Decrement the current cell particle counter ****/
                cel->n_particles--;

              /**** Find current cell index ****/
                curr_cell = cel - cell;

              /**** Find index of intersection face in new cell ****/
                for(fc = 0; fc < 3; fc++)
                    if(cell[cel->nbr[face]].nbr[fc] == curr_cell) break;

              /**** Store index of crossed face in new cell ****/
                part->crosd_face = (char)fc;

              /**** Store destination cell ****/
                part->dest_cell = cel->nbr[face];

              /**** Store time remaining for movement ****/
                part->mvmt_time = time_interval - t_min;

            }
        }
    }
}
```

Appendix B

Particle Communication Function

This appendix contains the function which transfers particles between cells. It is intended to serve only as an annotated listing; the full description of intercell communication is given in Section 2.6.

Arguments

The communication function takes two arguments. The first argument, 'cel' is a pointer to the cell that the particle is departing. The second argument, 'p_indx', is the index of this particle on the list in its current cell. It may be noted that the function accesses other cells in the course of its task. This is possible because the cell data was made globally-accessible to shorten the argument lists of oft-called functions.

Code Listing

```
void Communicate(struct cell_unit *cel,unsigned short p_indx)

    unsigned short d_cell, dcp_indx;
    struct atom *dc_part;

   /**** Decrement the traveler counter for old cell ****/
    cel->n_tvlrs--;

   /**** Note this particle's intended destination ****/
    d_cell = cel->particle[p_indx].dest_cell;

   /**** Mark destination cell as having received a traveler ****/
    cell[d_cell].tvlr_recd = 1;
```

```
/**** Mark this particle as stationary so it is not moved by a subsequent recursive
  call, leading to an infinite loop ****/
 cel->particle[p_indx].dest_cell = 1;

/**** Don't bother searching through particle list if this cell has no
  vacancies and no travelers ****/
 if (cell[d_cell].n_vacancies != 0 || cell[d_cell].n_tvlrs != 0){

    /**** Cycle through destination cell's particles, looking for an open space ***/
     for (dc_part = cell[d_cell].particle; dc_part < cell[d_cell].particle
        + cell[d_cell].plist_length; dc_part++){

        /**** If destination cell for a particle is one, its space is unavailable
          (it's not leaving current cell) so keep looking ****/
         if (dc_part->dest_cell == 1) continue;

        /**** If dest. cell is zero, space is open.  Place particle and return ****/
         else if(dc_part->dest_cell == 0){

            /**** Place particle in new cell ****/
             *dc_part = cel->particle[p_indx];

            /**** Increment new cell's particle counter ****/
             cell[d_cell].n_particles++;

            /**** Decrement new cell's vacancy counter ****/
             cell[d_cell].n_vacancies--;

            /**** Mark its old position as vacant ****/
             cel->particle[p_indx].dest_cell = 0;
             cel->n_vacancies++;
             return;
         }

        /**** If dest. cell is a true cell, recursively call this function to move
          the particle to its new cell then put current particle in its place ****/
         else{
             dcp_indx = dc_part - cell[d_cell].particle;
             Communicate(dcp_indx, &cell[d_cell]);
             cell[d_cell].particle[dcp_indx] = cel->particle[p_indx];
             cell[d_cell].n_particles++;
             cell[d_cell].n_vacancies--;
             cel->particle[p_indx].dest_cell = 0;
             cel->n_vacancies++;
             return;
         }
     }
 }

/* If no spaces are found in list, place particle at the end */

/**** First ensure there is room, making some if necessary ****/
 if (cell[d_cell].plist_length == cell[d_cell].pll_max){
     cell[d_cell].pll_max += P_LIST_INCREMENT;
     cell[d_cell].particle = (struct atom *)realloc(cell[d_cell].particle,
        (size_t)(cell[d_cell].pll_max * sizeof(struct atom)));
 }
 if(cell[d_cell].particle == NULL) printf("FAILED REALLOC!\n");fflush(NULL);
```

```
  /**** Now place particle, increment the particle counter of destination cell
    and decrement traveler counter of old cell ****/
  cell[d_cell].particle[cell[d_cell].plist_length++] = cel->particle[p_indx];
  cell[d_cell].n_particles++;
  cel->particle[p_indx].dest_cell = 0;
  cel->n_vacancies++;

  return;
}
```

Appendix C

Inflow/Outflow Function

This appendix contains the function which enforces boundary conditions at inflow / outflow cells. It is intended to serve only as an annotated listing; the full description of the inflow outflow boundary treatment is given in Section 2.7.2.

Arguments

The IO boundary function takes only one argument. This is a pointer to an array of structures containing IO cells, named 'io_cell'. Each of these structures contains the computed mean velocities as well as a pointer to the cell in the main array to which the IO face belongs.

Code Listing

```
void Process_IO(struct io *io_cell)
{
    float u, u_mean, u_norm, u_ext, u_distmax, dist_max, dist, xtemproot,
      x_temp, y_temp, z_temp, temp, v_mean, w_mean;
    int inp_diff, sgn, flag, pl_pos, ip;
    unsigned long which;
    struct io *io_cl;
    struct cell_unit *cel;
    struct atom *part;

   /**** Enforce boundary conditions at IO cells ****/
    for (io_cl = io_cell; io_cl < io_cell + n_iocells; io_cl++) {

        /**** Figure out to which cell this IO face corresponds ****/
        cel = &cell[io_cl->cell];
        u_mean = io_cl->u_mean;

        /**** Determine sign of inward-facing normal ****/
        if (cel->x[1] < 0.5*x_max){

            /**** If this is an inflow face, enforce temperature and transverse speed ****/
            sgn =  1;
            x_temp = y_temp = z_temp = 1.0;
            v_mean = w_mean = 0;
            temp = 1;
        }
        else{

        /**** If this case is exhausting to vacuum, skip outflow faces ****/
#ifdef VACOUT
            continue;
#endif
            /**** If this is an outflow face, get temp and t.v. speed from flow sample taken
              elsewhere (after the collision routine) ****/
            sgn = -1;
            v_mean = io_cl->v_mean;
            w_mean = io_cl->w_mean;
            x_temp = 2*(io_cl->u2 - u_mean*u_mean);
            y_temp = 2*(io_cl->v2 - v_mean*v_mean);
            z_temp = 2*(io_cl->w2 - w_mean*w_mean);
            temp = (x_temp + y_temp + z_temp) / 3.0;
            if (temp == 0.0) temp = 1;
        }

        /**** Calculate difference between target and actual number of particles,
          adding any round-off from previous steps and storing that from this
          step to lessen its effect in overall number of particles introduced **/
         io_cl->fnp_diff += cel->n_particles -
          (io_cl->pressure * cel->volume  * n_inf / temp);
         inp_diff = io_cl->fnp_diff;
         io_cl->fnp_diff -= inp_diff;

        /* Add particles if there aren't enough in cell */

         if (inp_diff < 0) {

            /**** Normal velocity is positive for inflow ****/
             u_norm = sgn * u_mean;

            /**** Calculate the maximum of the incoming velocity distribution****/
             u_distmax = sgn*0.5*(-u_norm +sqrt(u_norm*u_norm  +2));
             dist_max = sgn*(u_distmax+sgn*u_norm)*exp(-(u_distmax)*(u_distmax));

             xtemproot = sqrt(x_temp);

            /**** Set the extremum of particle speed. If u and u_norm are equal and
              opposite, the particle will not cross the boundary (inflow only)**/
             if (u_norm < 3.0)
                 u_ext = -u_norm;
             else
                 u_ext = -3.0*xtemproot; /*restrict the selection range to lower rejections*/

            /**** First, ensure there is room on cell's list for incoming particles ***/
```

```
if (cel->plist_length + -inp_diff -cel->n_vacancies >= cel->pll_max){
     cel->pll_max += 2 * (-inp_diff -cel->n_vacancies);
     cel->particle = (struct atom *)realloc(cel->particle,
(size_t)(cel->pll_max * sizeof(struct atom)));
}

/**** Initialize particle list position pointer ****/
pl_pos = 0;

for (ip = 0; ip < -inp_diff; ip++){

    /**** Search remaining positions in particle list for empty spots ****/
     for( ; pl_pos < cel->plist_length; pl_pos++)
          if (cel->particle[pl_pos].dest_cell == 0){
               cel->n_vacancies--;
               break;
          }

    /**** Incerement the cell's particle counter ****/
     cel->n_particles++;

    /**** If above search put us off particle list, extend it ****/
     if (pl_pos >= cel->plist_length) cel->plist_length++;

    /**** Assign a pointer to this particle for later convenience ****/
     part = &cel->particle[pl_pos];

    /**** Increment particle list position for next cycle ****/
     pl_pos++;

    /**** Set particle's destination cell and crossed face registers ****/
     part->dest_cell = 1;
     part->crosd_face = (char)io_cl->face;

    /**** Place the particle randomly on the cell's IO edge ****/
     part->x = cel->x[io_cl->face];
     part->y = cel->y[io_cl->face] + ran2(seed)*
        (cel->y[IRING(io_cl->face + 1)] - cel->y[io_cl->face]);

    /**** Assign tangential velocities according to equilibrium
      distribution ****/
     part->v = v_mean + sqrt(y_temp*-log(ran2(seed))) * sin(2.0 * M_PI
        *ran2(seed));
     part->w = w_mean + sqrt(z_temp*-log(ran2(seed))) * sin(2.0 * M_PI
        *ran2(seed));

    /**** Set u using a selection/rejection technique on a fluxal
      distribution ****/
     do
     {
        /**** Randomly select a value of u ****/
         if (sgn>0)
               u = (u_ext + (3.0 *xtemproot - u_ext) *ran2(seed));
      else
               u = (-(u_mean + 3.0 *xtemproot) + 3.0 *xtemproot *ran2(seed));

        /**** Compute the value of the distribution for this u ****/
         dist = sgn*(u/xtemproot + u_mean)*exp(-(u*u)/x_temp)/dist_max;
```

```
                }while (dist < ran2(seed));

              /**** Add mean velocity to assigned thermal speed ****/
                part->u = u_mean + u/xtemproot;

              /**** Move new particle a random fraction of one time step ****/
                part->mvmt_time = ran2(seed) * T_STEP;
                Move_Particle(cel, part, 1, SELECTIVE_MOVE);
            }
        }
    }
  /**** Process communication links until all are empty ****/
    do{
        flag = 0;
        for(cel = &cell[2]; cel < &cell[2]+N_CELLS; cel++){

          /**** Skip if this cell has no travelers ****/
            if (cel->n_tvlrs == 0) continue;

          /**** Set a flag to signify that a traveler was found ****/
            flag = 1;

          /**** Process all travelers in this cell ****/
            /*NOTE: Do not use pointers into the particle array in Communicate
              or its calls because it contains a realloc*/
            for(ip = 0; ip < cel->plist_length; ip++){

              /**** Skip if this is an empty position or stationary particle ****/
                if (cel->particle[ip].dest_cell < 2) continue;

                Communicate(ip, cel);

              /**** Go on to the next cell if we just placed the last traveler ****/
                if (cel->n_tvlrs == 0) break;
            }
        }
      /**** If communication was performed, move newly-placed travelers ****/
        if (flag){
            for(cel = &cell[2]; cel < &cell[2]+N_CELLS; cel++){

              /**** Skip this cell if it received no travelers ****/
                if (!cel->tvlr_recd) continue;

              /**** Reset cell 'traveler received' flag ****/
                cel->tvlr_recd = 0;

              /**** Move any particles which have remaining time ****/
                Move_Particle(cel, cel->particle, cel->plist_length,SELECTIVE_MOVE);
            }
        }
    }while(flag);
    return;
}
```

Bibliography

[1] G.A. Bird. *Molecular Gas Dynamics and the Direct Simulation of Gas Flows.* Oxford Engineering Science. Oxford University Press, New York, NY, 1994.

[2] W.B. Scott. Micro machines hold promise for aerospace. *Aviation Week and Space Technology*, 138(9), March 1993.

[3] S.A. Schaff and P.L. Chambre. *Fundamentals of Gas Dynamics*, chapter H. Princeton University Press, Princeton, New Jersey, 1958.

[4] A. Padmanabhan, H.D. Goldberg, K.S. Breuer, and M.A. Schmidt. A silicon micromachined floating-element shear stress sensor with optical position sensing by photodiodes. In *Proceedings of Transducers '95 and Eurosensors IX - 8th International Conference on Solid State Sensors and Actuators*, 1995.

[5] K. Nanbu. Theoretical basis of the direct simulation Monte Carlo method. In *Proceedings, 15th International Symposium on Rarefied Gas Dynamics, Grado, Italy, 1986*, Stuttgart, 1986. Teubner.

[6] K.C. Kannenberg, I.D. Boyd, and S. Dietrich. Development of an object-oriented parallel DSMC code for plume impingement studies. Paper 95-2052, AIAA, Washington, DC, 1995.

[7] G. Chen and I. Boyd. Statistical error analysis for the direct simulation Monte Carlo technique. Paper 95-2316, AIAA, Washington, DC, 1995.

[8] I. Greber and H. Wachman. Scaling rules and time averaging in molecular dynamics computations of transport properties. In *Rarefied Gas Dynamics: Theoretical*

and Computational Techniques; 16th International Symposium, Pasadena, CA, 1988, Washington, DC, July 1989. AIAA.

[9] G.A. Bird. *Molecular Gas Dynamics.* Oxford Engineering Science. Oxford University Press, London, 1976.

[10] G.A. Bird. Perception of numerical methods in rarefied gasdynamics. In *Rarefied Gas Dynamics: Theoretical and Computational Techniques*, volume 118 of *Progress in Aeronautics and Astronautics.* AIAA, Washington, DC, 1989.

[11] D. Baganoff and J.D. McDonald. A collision-selection rule for a particle simulation method suited to vector computers. *Physics of Fluids A*, 2(7):1248–1259, July 1990.

[12] James B. Elgin. Getting the good bounce: Techniques for efficient Monte Carlo analysis of complex reacting flows. Technical Report SSI-TR-28, Spectral Sciences, Inc., Burlington, MA.

[13] W. G. Vincenti and C. H. Kruger. *Introduction to Physical Gas Dynamics.* Wiley, New York, NY, 1965.

[14] K. Koura and H. Matsumoto. Variable soft sphere molecular model for inverse-power-law or Lennard Jones potential. *Physics of Fluids A*, 3:2459–2465, 1991.

[15] G.A. Bird. Monte Carlo simulation in an engineering context. In *Proceedings, 12th International Symposium on Rarefied Gas Dynamics, Charlottesville, VA, 1980*, volume 74 of *Progress in Aeronautics and Astronautics.* AIAA, Washington, DC, 1981.

[16] H. A. Hassan and D. B. Hash. A generalized hard-sphere model for Monte Carlo simulations. *Physics of Fluids A*, 5:738–744, 1993.

[17] D.F. Watson. Computing the n-dimensional Delaunay tesselation with application to Voroner polytopes. *Computer Journal*, 24(2), 1981.

[18] S. Dietrich. Efficient computation of particle movement in 3-D DSMC calculations on structured body-fitted grids. In *Proceedings, 17th International Symposium on Rarefied Gas Dynamics, Aachen, Germany, 1990*, Weinheim, 1991. VCH.

[19] S. Dietrich and I. Boyd. A scalar optimized parallel implementation of the DSMC method. Paper 94-0355, AIAA, Washington, DC, 1994.

[20] J.D. Anderson. *Modern Compressible Flow with Historical Perspective.* McGraw-Hill, New York, NY, second edition, 1990.

[21] C. Hirsch. *Numerical Computation of Internal and External Flows*, volume 2. Wiley, Chichester, 1990.

[22] J.C. Harley, Y. Huang, H.H. Bau, and J.N. Zemel. Gas flow in micro-channels. *Journal of Fluid Mechanics*, 284, February 1995.

[23] E.B. Arkilic, M.A. Schmidt, and K.S. Breuer. Slip flow in microchannels. In *Proceedings of the 19th Rarefied Gas Dynamics Symposium*, Washington, DC, July 1994. AIAA.

[24] A. Beskok and G.E. Karniadakis. Simulation of heat and momentum transfer in complex microgeometries. *Journal of Thermophysics and Heat Transfer*, 8(4):647–655, 1994.

[25] T.J. Bartel, T.M. Sterk, J.L. Payne, and B. Preppernau. DSMC simulation of nozzle expansion flow fields. Paper 94-2047, AIAA, Washington, DC, 1994.

[26] F.M. White. *Viscous Fluid Flow.* McGraw-Hill, New York, NY, second edition, 1991.

[27] R.H. Ni. A multiple grid scheme for solving the euler equations. *AIAA Journal*, 20(11):1565–1571, 1982.

[28] D.B. Hash and H.A. Hassan. A hybrid DSMC/Navier-Stokes solver. Paper 95-0410, AIAA, Washington, DC, 1995.