An Accurate Kinetic Scheme for 3D Solution of the Boltzmann Equation

Andrew J. Christlieb* and W. Nicholas G. Hitchon†

*Department of Aerospace Engineering
University of Michigan
1320 Beal Avenue
Ann Arbor, MI 48109-2140
†Department of Electrical and Computer Engineering
University of Wisconsin
1415 Engineering Dr.
Madison, WI 53706

Abstract. Low speed neutral particle transport, in long mean free path (LMFP) environments, presents challenges for well-established techniques, such as the direct simulation Monte Carlo (DSMC) method. In particular, at low flow velocities, statistical methods suffer from noise that may render them impractical in LMFP environments [1].

We describe a non-statistical (no random numbers are used) kinetic model for particle transport [2, 3, 4]. The behavior of particles is handled in two stages, Ballistic and Collisional. The ballistic operator tracks the location of the particles across a phase space mesh until the particles undergo a collision. The collision operator redistributes the particles in direction and energy in such a way as to conserve momentum and energy at each spatial location of the phase space mesh.

In the past, the method has been applied to heat transport in LMFP environments [4]. The current application centers on flow past a micro air foil. We focus on the method and its extension to handle directional flows. Some typical results for high Knudsen number, $(K_n = \frac{\lambda}{L})$, flows past a flat plate are presented.

INTRODUCTION

We have developed a kinetic solver that is well suited for low Mach number, $M < 0.3$, flows in high Knudsen number, $K_n > 0.05$, environments. With the emergence of Micro Electromechanical Systems (MEMS) and the ongoing reduction to Nano-Technology Based Systems (NTBS), these types of models are expected to play a key role as design tools for MEMS/NTBS.

The model permits efficient, non-statistical iterative calculation of the scattering rate of particles in each cell of a 6D phase space mesh. The method uses 'propagating' directions for tracking the transport of particles throughout phase space. The spatial meshes may consist of arbitrarily shaped elements. This transport model is similar in spirit to earlier transition probability transport codes [2, 3, 4, 5, 6, 7]. Although earlier codes [5, 6] are capable of handling arbitrary spatial meshes, they typically assume isotropic scattering and make other simplifying assumptions which reduce the computational overhead, making it possible to store large amounts of geometrical information typically needed.

The model presented here does not assume isotropic scattering of particles. Another important difference from the earlier models [5, 6] is that we conserve momentum, giving our model the capability to describe a flowing gas. Two momentum conserving collision operators have been employed [8], a simple monoenergetic operator and the BGK model. The monoenergetic model offers advantages in terms of speed, while the BGK model offers a more realistic description. One consequence is that a 'full' phase space mesh is required. However, the computational overhead of storing the probabilities for a full phase space mesh is large. For example, if N_c is the number of spatial cells on the mesh, then even the geometric information required to find the probabilities of going from each initial cell to each final cell involves $(N_c)^2$ numbers. To overcome this problem we limit the amount of information stored by efficiently computing needed information on the fly.

In the past the method has been applied to heat transfer in a rare gas between parallel plates at different temperatures [4]. Results were generated for several Knudsen numbers in the transition regime. The results of the kinetic simulation

CP663, *Rarefied Gas Dynamics: 23rd International Symposium*, edited by A. D. Ketsdever and E. P. Muntz

which employ the BGK operator compare favorably with those of a finite difference solution of the Boltzmann equation using the BGK collision operator [9]. The results for both collision models exhibit fair agreement with experimental data of Teagan and Springer [10].

The code has been extended to study gas flow past a flat plate at high K_n. Some results of the enhanced code will be presented.

TRANSITION PROBABILITY MATRIX (TPM) METHOD

Our goal is to solve the steady state Boltzmann equation for LMFP environments,

$$\vec{v} \cdot \nabla_r f(\vec{r}, \vec{v}) + \frac{\vec{F}}{m} \cdot \nabla_v f(\vec{r}, \vec{v}) = \frac{\delta f}{\delta t}|_{collision}. \tag{1}$$

The method is based on a one-step transition probability matrix (TPM) which describes steady state particle transport on a phase space mesh. However, this matrix need not be formally constructed. Instead, the transport of the particles is described with two operators; a 'transport' operator and a 'collision' operator.

In the next section we provide an overview of the method, followed by a more detailed description of the two operators. It worth noting that this approach can be extended to handle flows which evolve in time without the introduction of a costly probability history, as was the case in reference [11].

Overview

We present an overview of the TPM. The method is 6*D*, i.e., it uses a uniform 3*D* spatial mesh and a 3*D* velocity space mesh comprising a single energy mesh in combination with a 2*D* (Φ, Θ) directional mesh. In the past, we have not restricted our work to uniform spatial meshes[4]. Energy, momentum and particle conservation are strictly enforced.

Our approach in solving equation 1 is to solve for the collision rates at each location of the phase space mesh. By solving for the collision rates in each phase space mesh cell, information about the direction and energy of the particles that make up the flow is captured at each spatial location of the mesh. T is the matrix which represents the probability that, starting at location $(\mathbf{c}', \mathbf{a}', E')$, a particle will have its next collision and be redistributed to location $(\mathbf{c}, \mathbf{a}, E)$ for all phase space locations $(\mathbf{c}', \mathbf{a}', E')$ and $(\mathbf{c}, \mathbf{a}, E)$. $\mathbf{c}'$ and $\mathbf{c}$ are spatial locations on the phase space mesh. The velocity information is contained in $(\mathbf{a}', E')$ and $(\mathbf{a}, E)$ where $\mathbf{a}$ and $\mathbf{a}'$ represent the (Φ, Θ) directional information and E and E' represent the energy. Let $R(\mathbf{c}', \mathbf{a}', E')$ be the number of particles that collided in cell $\mathbf{c}'$ that were redistributed with direction $\mathbf{a}'$ and energy E' at the previous iteration, i.e., $R(\mathbf{c}', \mathbf{a}', E')$ is the collision rate for particles that collided in phase space location $(\mathbf{c}', \mathbf{a}', E')$ at the previous iteration. Then the number of these particles which collide in cell $\mathbf{c}$ that were redistributed with direction $\mathbf{a}$ and energy $E$is given by

$$R(\mathbf{c}, \mathbf{a}, E) = T(R(\mathbf{c}', \mathbf{a}', E')). \tag{2}$$

We break T into two operators, T_{bal} and T_{col}. T_{bal} is the one-step transition probability matrix for particle ballistic motion and and T_{col} the one-step transition probability matrix which locally redistributes the particles in energy and direction after a collision.

The first transition probability matrix is used to compute the number of particles per second that collide in cell $\mathbf{c}$ with direction $\mathbf{a}'$ and energy E',

$$R(\mathbf{c}, \mathbf{a}', E') = \sum_{\mathbf{c}'} R(\mathbf{c}', \mathbf{a}', E') T_{bal}(\mathbf{c}, \mathbf{a}', E' : \mathbf{c}', \mathbf{a}', E'), \tag{3}$$

where $R(\mathbf{c}, \mathbf{a}', E')$ is the number rate of particles that collide in cell $\mathbf{c}$ with direction $\mathbf{a}'$ and energy E' and $T_{bal}(\mathbf{c}, \mathbf{a}', E' : \mathbf{c}', \mathbf{a}', E')$ is the probability that a particle having started in cell $\mathbf{c}'$ with direction $\mathbf{a}'$ and energy E' will have its next collision in cell $\mathbf{c}$, where the sum is over all mesh cells at location $\mathbf{c}'$ with direction $\mathbf{a}'$ and energy E'.

The second TPM is used to redistribute the particles after a collision,

$$R(\mathbf{c}, \mathbf{a}, E) = \sum_{\mathbf{a}'} \sum_{E'} R(\mathbf{c}, \mathbf{a}', E') T_{col}(\mathbf{c}, \mathbf{a}, E : \mathbf{c}, \mathbf{a}', E'), \tag{4}$$

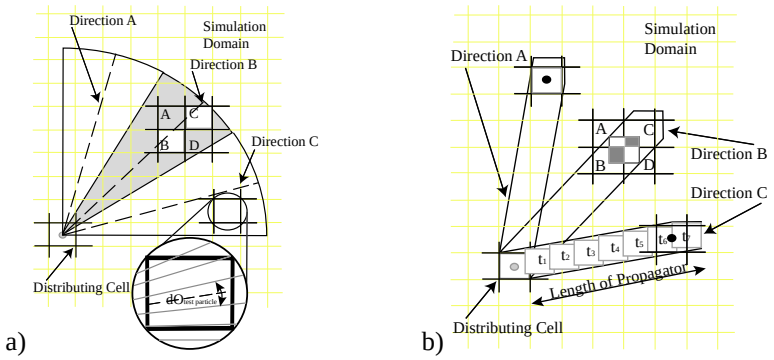

FIGURE 1. Two possible propagating structures used in the transport phase, T_{bal}. a) is a point source propagator and b) is a Convective Scheme based propagator.

where $T_{col}(\mathbf{c},\mathbf{a},E:\mathbf{c},\mathbf{a}',E')$ is the probability that a particle, having collided in cell $\mathbf{c}$ with direction $\mathbf{a}'$ and energy E' will be redistributed with direction $\mathbf{a}$ and energy E (in the same spatial cell $\mathbf{c}$) [3, 8].

The collision rates, $R(\mathbf{c},\mathbf{a},E)$, can be used to reconstruct any desired information about the flow. For example, the density $n(\mathbf{c})$ is

$$n(\mathbf{c}) = \sum_E \frac{R(\mathbf{c},E)\lambda(\mathbf{c},E)}{|v(E)|\gamma(\mathbf{c})}, \tag{5}$$

where $n(\mathbf{c})$ is the density in cell $\mathbf{c}$ of the spatial mesh, $R(\mathbf{c},E) = \sum_{\mathbf{a}} R(\mathbf{c},\mathbf{a},E)$, $\lambda(\mathbf{c},E)$ is the mean free path, $|v(E)|$ is the magnitude of the velocity, and $\gamma(\mathbf{c})$ is the volume of cell $\mathbf{c}$.

In the following sections we describe how the TPM are set up without explicitly constructing either of the large matrices, T_{bal} or T_{col}.

Ballistic Transport Operator: Constructing $R(\mathbf{c},\mathbf{a}',E')$

The ballistic transition probability matrix T_{bal} has been implemented with two different operators, the Point Source (PS) operator and the Convective Scheme (CS) operator. Both operators make full use of symmetry to reduce computational overhead, see reference [4]. The PS operator is nearly identical to the T_{bal} found in reference [4] while the CS operator plays a role similar to the T_{bal} in reference [3].

In computing $R(\mathbf{c},\mathbf{a}',E')$, $T_{bal}(\mathbf{c},\mathbf{a}',E':\mathbf{c}',\mathbf{a}',E')$ is not explicitly constructed. Instead, the TPM employs an average length, $\langle L(\mathbf{c})\rangle$, and a fractional overlap, $f_{\mathbf{a}'}(\mathbf{c},\mathbf{c}')$. $\langle L(\mathbf{c})\rangle$ is the distance a particle would travel on average when passing through a cell $\mathbf{c}$ and $f_{\mathbf{a}'}(\mathbf{c},\mathbf{c}')$ is the fraction of particles starting in cell $\mathbf{c}'$, moving in the direction $\mathbf{a}'$, that pass through cell $\mathbf{c}$ [4].

The propagator subdivides space into a finite number of directions

$$\mathbf{a}'_{i,j} = \left(\Phi_i, \Theta_j\right). \tag{6}$$

and extends a finite distance,

$$L_{prop} = \chi\langle\lambda(\mathbf{c}',E')\rangle \tag{7}$$

where $\langle\lambda(\mathbf{c}',E')\rangle$ is the average mean free path and $\chi > 0$ is a constant. Figure 1a) is the PS propagator and figure 1b) is the CS propagator. Both propagators contain a list (in radial order) for each $\mathbf{a}'_{i,j}$. Each list consists of the geometric information $(f_{\mathbf{a}'_{i,j}}(\mathbf{c},\mathbf{c}'),\langle L(\mathbf{c})\rangle)$ for each mesh cell $\mathbf{c}$ that particles moving in the direction $\mathbf{a}'_{i,j}$ interact with. The PS propagator covers the entire (Φ,Θ) space with propagating rays centered about each direction $\mathbf{a}'_{i,j}$. The PS assumes that particles which collide in mesh cell $\mathbf{c}'$ can be redistributed from the center of the cell [4]. The geometrical information can be computed via test particles, or as in [4]. The Φ-Θ range of each direction $\mathbf{a}'_{i,j}$ is uniformly subdivided in Φ and Θ to give M sub-rays, δ_i. The solid angle, $d\Omega_i$, of each δ_i is computed and a test particle is assigned to δ_i. To obtain $f_{\mathbf{a}'_{i,j}}(\mathbf{c},\mathbf{c}')$ and $\langle L(\mathbf{c})\rangle$, three pieces of information are tracked for each mesh cell

c; the number of test particles that pass through the cell, P_c, the total distance all of the particles travel before leaving **c**, $l(\mathbf{c})$, and the total solid angle that the test particles sweep out when passing though **c**, $d\Omega(\mathbf{c})$. Then $\langle L(\mathbf{c})\rangle = \frac{l(\mathbf{c})}{P_c}$ and $f_{\mathbf{a'}_{i,j}}(\mathbf{c},\mathbf{c'}) = \frac{d\Omega(\mathbf{c})}{d\Omega_{\mathbf{a'}_{i,j}}}$ where $d\Omega_{\mathbf{a'}_{i,j}}$ is the solid angle of the ray centered about $\mathbf{a'}_{i,j}$. The enlarged cell in figure 1a) direction C depicts this process. The light gray lines are the boundaries of the solid angles, the dashed line is the distance a test particle travels when moving through the cell and the solid angle attributed to the test particles is shown.

The CS computes the actual probabilities $T_{bal}(\mathbf{c},\mathbf{a'},E' : \mathbf{c'},\mathbf{a'},E')$. This information is then converted into the format of $(f_{\mathbf{a'}_{i,j}}(\mathbf{c},\mathbf{c'}),\langle L(\mathbf{c})\rangle)$, so that the same code can use either propagator. The propagator is constructed for one direction at a time. Let **A** be an array with dimension $N \times N$ in 2*D* or $N \times N \times N$ in 3*D*, where $N = 2\lfloor \frac{\chi\langle\lambda(\mathbf{c'},E')\rangle}{\Delta M} + 0.5\rfloor$ and ΔM is the mesh spacing. **A** stores the probabilities of particles scattering in any given phase space cell. The CS uses 'long lived' moving cells (LLMC) [12, 13] to compute $T_{bal}(\mathbf{c},\mathbf{a'},E' : \mathbf{c'},\mathbf{a'},E')$. For 'angle' $\mathbf{a'}_{i,j}$, a LLMC is constructed, $MC(\mathbf{c'},\mathbf{a'}_{i,j})$, that initially overlaps the mesh cell $\mathbf{c'}$ and is assigned a 'density', $n = 1$. The $MC(\mathbf{c'},\mathbf{a'}_{i,j})$ is propagated along the direction $\mathbf{a'}_{i,j}$ by time stepping. Figure 1b) direction c depicts seven time steps where t_i is the i^{th} time step and $i \in \{1,\ldots,7\}$. After each time step, the fraction of the particles

$$f_{MC(\mathbf{c'},\mathbf{a'}_{i,j})} = n\left(1 - e^{-\frac{d(t_i)}{\langle\lambda(\mathbf{c'},E')\rangle}}\right), \tag{8}$$

which collided in time step t_i is subtracted from n of $MC(\mathbf{c'},\mathbf{a'}_{i,j})$. $d(t_i)$ is the distance travelled in time t_i. $f_{MC(\mathbf{c'},\mathbf{a'}_{i,j})}$ is subdivided among the cells that $MC(\mathbf{c'},\mathbf{a'}_{i,j})$ overlapped after the time step t_i, and is added to entries in **A** which correspond to these mesh cells, figure 1b) direction b where the cells marked *A*, *B*, *C*, and *D* are the cells overlapped by the moving cell at a particular step. In 2*D*, the fractions of particles added to cells *A*, *B*, *C*, and *D* of **A**, after the t_i step, are;

$$\mathbf{A}_A = \left(\frac{x_{A-max} - x_{MC-min}}{\Delta M}\right)\left(\frac{y_{MC-max} - y_{A-min}}{\Delta M}\right) f_{MC(\mathbf{c'},\mathbf{a'}_{i,j})} \tag{9}$$

$$\mathbf{A}_B = \left(\frac{x_{B-max} - x_{MC-min}}{\Delta M}\right)\left(\frac{y_{B-max} - y_{MC-min}}{\Delta M}\right) f_{MC(\mathbf{c'},\mathbf{a'}_{i,j})} \tag{10}$$

$$\mathbf{A}_C = \left(\frac{x_{MC-max} - x_{C-min}}{\Delta M}\right)\left(\frac{y_{MC-max} - y_{C-min}}{\Delta M}\right) f_{MC(\mathbf{c'},\mathbf{a'}_{i,j})} \tag{11}$$

$$\mathbf{A}_D = \left(\frac{x_{MC-max} - x_{D-min}}{\Delta M}\right)\left(\frac{y_{D-max} - y_{MC-min}}{\Delta M}\right) f_{MC(\mathbf{c'},\mathbf{a'}_{i,j})} \tag{12}$$

where $x/y_{\mathbf{name}-max/min}$ are the coordinates of the mesh cell or moving cell. (See references [12] and [13] for a full description of the CS.) This process of time stepping is continued down the propagating direction until the LLMC reaches the end of the propagator. At this point, all remaining material in $MC(\mathbf{c'},\mathbf{a'}_{i,j})$ is subdivided among the last group of cells the LLMC overlaps. Any non-zero element in **A** represents a cell **c** that $\mathbf{c'}$ interacts with, along direction $\mathbf{a'}_{i,j}$, and will be added to the list. Using the equation,

$$\langle L(\mathbf{c})\rangle = -1\langle\lambda(\mathbf{c'},E')\rangle ln\left(\frac{n_{total}(c)}{n_0} - 1\right) \tag{13}$$

it is possible to back out an average length for the particles that pass though cell **c** moving in the direction $\mathbf{a'}_{i,j}$, where $n_{total}(c)$ is the total fractional amount of particles that collide in cell **c**, n_0 is one and the propagator fractional overlaps are set to one, $f_{\mathbf{a'}_{i,j}}(\mathbf{c},\mathbf{c'}) = 1$. For a single initial cell, the CS does not cover all of (Φ,Θ) space, as can be seen in figure 1b), but the CS does redistribute particles that have collided in cell $\mathbf{c'}$ from the whole volume of $\mathbf{c'}$, as it should [3].

Propagation is performed by allowing particles to move along the $\mathbf{a'}_{i,j}$ from $\mathbf{c'}$ encountering other cells of the fixed mesh in order of increasing radius and is completed when the $\mathbf{a'}_{i,j}$ of all $\mathbf{c'}$ have deposited all of their particles back in the simulation domain. The number of particles originating in cell $\mathbf{c'}$ with direction $\mathbf{a'}_{i,j}$ and energy E' that have their next collision in cell **c** is,

$$N_c(\mathbf{c},\mathbf{a'}_{i,j},E' : c',\mathbf{a'}_{i,j},E') = N_{\mathbf{a'}_{i,j}} f_{\mathbf{a'}_{i,j}}(\mathbf{c},\mathbf{c'})\left(1 - e^{-\frac{\langle L(\mathbf{c})\rangle}{\lambda(c,E')}}\right), \tag{14}$$

where $N_{\mathbf{a}'_{i,j}}$ is the number of particles left in the propagator with direction $\mathbf{a}'_{i,j}$ and energy E' at cell $\mathbf{c}$.

$R(\mathbf{c}, \mathbf{a}'_{i,j}, E')$ can now be computed from

$$R(\mathbf{c}, \mathbf{a}'_{i,j}, E') = \sum_{\mathbf{c}'} N_c(\mathbf{c}, \mathbf{a}'_{i,j}, E' : c', \mathbf{a}'_{i,j}, E'), \tag{15}$$

where the sum is over all mesh cells $\mathbf{c}'$ of the fixed mesh. Particles having scattered in a cell $\mathbf{c}'$, during the ballistic step, are placed in the rays of $\mathbf{c}'$ in such a way as to conserve momentum, as discussed in the next section.

Collision Operator: Constructing $R(\mathbf{c}, \mathbf{a}, E)$

T_{col} redistributes particles on the mesh in energy while conserving momentum in particle-particle collisions. Application of T_{col} to $R(\mathbf{c}, \mathbf{a}', E')$ (constructed during the ballistic move) gives $R(\mathbf{c}, \mathbf{a}, E)$. We now give an overview of T_{col}, followed by its implementation.

In this work the collisions have been described using two different models. In the first, particles that undergo a collision in cell $\mathbf{c}$ are put back on the phase space mesh at the average energy of the particles that collided in the cell during the current simulation step. Momentum is conserved using the angular distribution $f(\mathbf{a})$.

The second collision model used for the interior of the simulation domain is a modified BGK model, which is

$$\frac{\delta f}{\delta t}|_{collision} = \nu n(c) f_0(E) f(\mathbf{a}) - \nu f(\vec{r}, \vec{v}), \tag{16}$$

where $f_0(E)$ is the Maxwellian distribution function. $f(\mathbf{a})$ contains directional information as described below.

The monoenergetic T_{col} and T_{col} defined using Equation 16 conserve energy and momentum, locally on average, i.e., the average of the energy and momentum of the particles in mesh cell $\mathbf{c}$ after redistribution by collisions is forced to equal the average energy and momentum of the particles in cell $\mathbf{c}$ before they are redistributed. Details of the implementation of either T_{col} can be found in reference [4].

For each cell, the total momentum brought into the cell by particles which collide there is computed (their initial direction being taken to be that of the center of the incoming rays.) Then the distribution of outgoing particles is chosen to conserve that momentum [4, 8]. The form of the distribution is

$$f(\mathbf{a}) = (1 + \alpha v_x(\mathbf{a})/v + \beta v_y(\mathbf{a})/v + \gamma v_z(\mathbf{a})/v), \tag{17}$$

where v is the speed of the outgoing particles, $v_x(\mathbf{a})$, $v_y(\mathbf{a})$ and $v_z(\mathbf{a})$ are the components of velocity of the particles moving in the direction $\mathbf{a}$ and α, β, and γ are normalization factors. The distribution $f(\mathbf{a})$ must total to one, i.e., $\int f(\mathbf{a}) d\Omega_a = 1$. The integral of $f(\mathbf{a})$ places the needed constraint on equation 17 in order to find α, β, and γ. The mean value of v_x is

$$\langle v_x \rangle = \alpha v \eta_x, \tag{18}$$

where η_x is an integration factor. Since we chose to approximate the integration using a summation, η_x is given by

$$\eta_x = \frac{\sum_a d\Omega_a}{\sum_a (v_x/v)\, d\Omega_a}. \tag{19}$$

Once η_x is determined, for the given set of directions, α can be found exactly, to ensure momentum conservation. η_y and η_z are found in the same manner. (The η's are the same for all locations and energies.) This approach was outlined in [8], where particle and energy conservation were also discussed. [8] provides details, including how large values of the mean velocity are handled. Other angular distributions could be employed instead, for instance to allow for an accurate differential cross section.

Boundary Conditions and Surface Interactions

The effect of a reflecting surface should, for a uniform density above the surface, be to give the same flux coming back off the surface as we would have for a uniform density behind the surface. Similarly, at the edge of the simulation

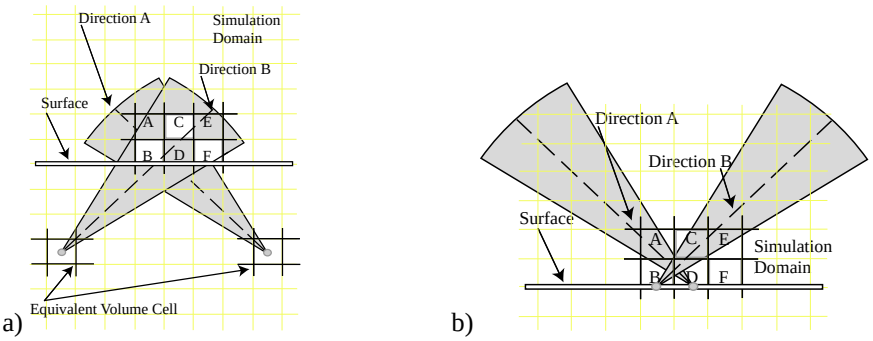

FIGURE 2. Depiction of equivalent volume vs. surface point source coverage of space. Figure a) shows equivalent volume coverage of space for two directions and figure b) shows surface point source coverage of space for roughly the same two directions.

region, where gas is introduced, the flux coming in should be exactly what we would get from a large volume where we (usually) specify a uniform density. Much of our effort in setting up the simulation goes into handling boundaries so as to mimic these 'equivalent' volumes.

One difficulty with making a surface reproduce its equivalent volume is caused by the fact that we sometimes use PS, instead of integrating the source over the entire initial cell. The PS uses rays (about each direction $\mathbf{a}'_{i,j}$) which fill the solid angle centered around the PS. Particles travel down these rays and strike all the cells within each ray. Rays coming from points on a surface cannot exactly reproduce the angular and spatial distribution of particles coming from PS in an equivalent volume behind the surface, see figure 2.

The boundary conditions we describe do achieve the effect of replacing the surface with an 'exact' equivalent. In the case of point sources, we have to replace the surface with a set of fictional volume cells, behind the surface, and use a method of images. Numerous other versions of the boundary conditions were considered, which failed for subtle reasons which we do not have the space to describe here. The methods we describe here are the simplest that we have discovered which satisfy the 'equivalent volume' test.

The CS provides an alternative way to handle generating probabilities, which eliminates the error in using a PS distribution off surfaces. This scheme is more convenient and easier to use than point sources, for several reasons. One advantage is that launching a CS propagator from a surface is indeed capable of generating the same angular distribution as the equivalent volume. This will be described later in this section.

As mentioned, boundary conditions must strictly meet an 'equivalent volume' test, in order to produce a satisfactory density profile. At the outer boundary, we inject particles from a volume region which is several mean free paths deep, to accomplish this. This would be quite unwieldy, but we can store the particle collision rate on the mesh, of the injected particles, and add that rate back at each step. In addition, we can store the profile in a compact form, and allow for variations in mean free path, using a combination of the CS propagator, mapping back, and the null collision operator.

Reflection at a surface is handled by the method of images. Particles which reflect specularly are allowed to travel down a fictitious ray, which is the continuation of the original ray behind the surface, and are placed in fictitious volume cells, see figure 3. After they have been propagated along the ray, they are reflected back into the real volume, i.e., what collides in volume reflecting cell B is placed in cell B with the correct directional information. Again, this would be unwieldy, but in fact we can map them back into (fictitious) volume cells immediately after they cross the reflecting boundary. These are then reflected back into the real volume. This procedure achieves the equivalent effect in a more efficient fashion.

Diffuse reflection is handled by finding a set of 'mirror image' rays for an incoming ray, whose center hits a particular surface cell $\tilde{\mathbf{c}}$(the dot on the wall of figure 3 b). If the incoming ray is at angles labelled (i,j), the mirror image rays are (-i,j), (i,-j) and (-i,-j). Sharing the particles among these rays equally provides total momentum loss. The shared particles encounter both the volume reflecting cell in the original direction (reflecting cell B) and the 'conjugate' volume reflecting cells, reflecting cell B' in figure 3 b). B' is the mirror image obtained when B is reflected about the center of the surface cell $\tilde{\mathbf{c}}$. Particles placed in the conjugate cells are reflected in the same manner as those placed in reflecting cell B. Partial sharing is done to achieve a mixture of specular and diffuse reflection; so perhaps 10% of the ray might scatter specularly and the remaining 90% be shared. In addition to this procedure providing exact momentum loss, for a uniform source above the surface it guarantees satisfaction of the equivalent volume rule, which

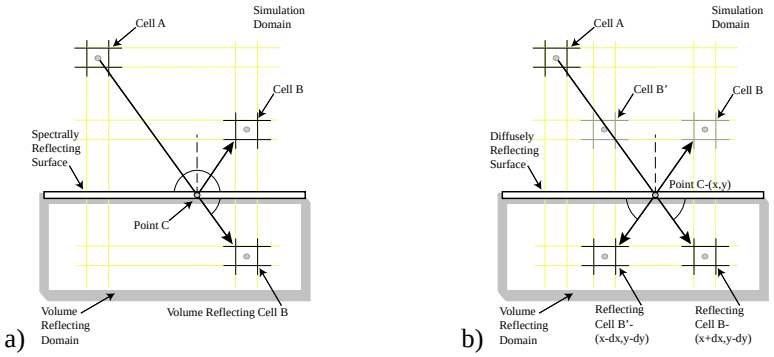

FIGURE 3. Depiction of how wall collisions are handled for spectrally and diffusely reflecting particles. Figure a) depicts spectrally reflecting particles and figure b) depicts diffusely reflecting particles.

is to say, it gives the exact numerical equivalent of the $cos\Theta$ distribution coming off the surface.

Mapping Back and the Null Collision Operator

The advantages of the CS version of the propagator follow in part from the fact that it only employs a discrete set of angles, whereas the PS version employs rays. The rays are centered on discrete angles but they spread over a range of angles. This allows the CS version to use 'mapping back' of particles, without altering the particles' angular distribution. While the mapping back is not essential, it makes several procedures a great deal more efficient. An increase in efficiency comes from fact that the length of time an individual iteration takes is proportional $(\chi\langle\lambda(\mathbf{c},E)\rangle/\Delta M)^2$ in $2D$ and $(\chi\langle\lambda(\mathbf{c},E)\rangle/\Delta M)^3$ in $3D$. This is because the number of cells the propagator must loop over in a single step is proportional to the the number of cells that fit inside the area or volume the propagator sweeps out in space.

In this work we sometimes map back particles in a ray, to achieve greater efficiency, before the particles have travelled as far as they eventually will along a direction **a**. After they have travelled one or two mean free paths, so relatively few particles remain in the ray, there is little benefit to following the particles further. Instead, we replace them in a cell **c** of the mesh, at the exact same angle and speed. Their motion along **a** is then continued when we next propagate all of the particles in that volume cell which have the same momentum. There is some numerical diffusion in space, in this process, but none in momentum, and few particles are involved. Importantly, this process does not introduce non-uniformities into an otherwise uniform flow. Since the density variations we are studying are sometimes small, we must ensure that numerical errors do not introduce density variations.

The null collision operator was introduced for Monte Carlo simulations [14] and we have used it in TPM calculations. It consists of overestimating the collision rate by assuming a constant mean free path λ (or collision frequency ν) which is known to be too small (large). The fixed λ (ν) makes it easier to calculate the propagator. The over counting of the collisions is remedied by taking some of the particles, which were removed from the ray as having collided, and mapping them back, as described above, with the identical momentum they had before their spurious collision.

RESULTS

Typical results for the flow past a flat plate, generated by the TPM, are shown in figure 4. The depicted flow is for argon flowing over a flat plate. K_n is 0.2 and the inlet (far field) velocity is set to $(v_x = 40\frac{m}{sec.}, v_y = 0\frac{m}{sec.}, v_z = 0\frac{m}{sec.})$. Since the thermal velocity of argon is around $430\frac{m}{sec.}$, $M \sim 0.1$. The mesh spacing was about a mean free path, i.e., $\Delta M = \langle\lambda(\mathbf{c},E)\rangle$. The upstream region was $20\Delta M$, the downstream region was $35\Delta M$, and the width of the simulation domain was $20\Delta M$.

These results were generated using the CS propagator. Results generated using the PS propagator, for these flow conditions, exhibit only minor differences, and so they are not shown.

Because there is no experimental data to compare with, the results have been compared to those generated by the Information Preserving (IP) method (reference [1]), results not shown here. The TPM and IP method predict similar

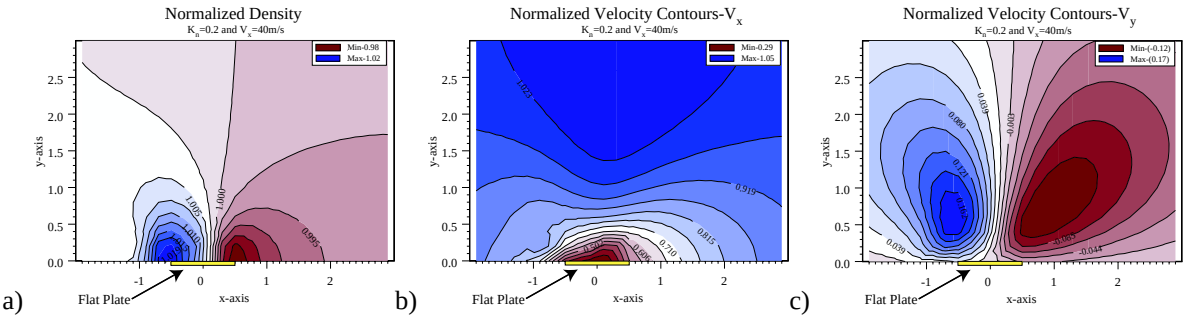

FIGURE 4. Typical results for flow past a flat plate. The flow is argon gas with an inlet velocity of $(40m/sec, 0, 0)$ and a $K_n = 0.2$. Figure a) is the normalized density contours. Figure b) and c) are the normalized v_x and v_y contours respectively.

flow behavior for these flow conditions.

CONCLUDING REMARKS

We have presented a kinetic transport model well suited for handling low Mach number flows in high Knudsen number environments. The model gives a very accurate handling of the flow, in circumstances where particle simulations suffer from statistical noise. In this paper we emphasized the method, including handling of boundary conditions and the construction of a 'propagator' using the 'Convective Scheme'.

ACKNOWLEDGMENTS

We would like to thank the Air Force Office of Scientific Research for supporting the research under grant number F49620-98-1-0433.

REFERENCES

1. Boyd, I., and Sun, Q., *39th AIAA Aerospace Science Meeting, Reno Nevada* (January, 2001).
2. Harvey, R., Hitchon, W., and Parker, G., *Journal of Applied Physics*, **75**, 1940 (1994).
3. Parker, G., Hitchon, W., and Keiter, E., *Physical Review E*, **54**, 938 (1996).
4. Christlieb, A., and Hitchon, W., *Physical Review E*, **65**, Article 056708 (2002).
5. Askew, J., *Journal of British Nuclear Energy Society*, **5**, 564 (1966).
6. Vujic, J., *Proceedings of the International Conference on Mathematical Methods and Supercomputing in Nuclear Applications, Karlsruhe.* (1993).
7. Cale, T., Raupp, G., and Gandy, T., *Journal of Applied Physics*, **68**, 3645 (1990).
8. Hitchon, W., Parker, G., and Lawler, J., *IEEE Transactions on Plasma Science*, **22/3**, 267 (1994).
9. Ohwada, T., *Physics of Fluids*, **8(8)**, 2153 (1996).
10. Teagan, W., and Springer, G., *Physics of Fluids*, **11(3)**, 497 (1968).
11. Tan, W., Hoekstra, R., and Kushner, M., *Journal of Applied Physics*, **79(7)**, 3423 (1996).
12. Christlieb, A., Hitchon, W., and Keiter, E., *IEEE Transactions on Plasma Science*, **28:6**, 2214 (2000).
13. Feng, J., and Hitchon, W., *Physical Review E*, **61**, 3160 (2000).
14. Boardman, A., Fawcett, W., and Swain, S., *Journal of Physical Chemistry*, **31**, 1963 (1970).