JOURNAL OF SPACECRAFT AND ROCKETS
Vol. 48, No. 3, May–June 2011

Particle Simulations of Planetary Probe Flows Employing Automated Mesh Refinement

Da Gao,* Chonglin Zhang,† and Thomas E. Schwartzentruber‡
University of Minnesota, Minneapolis, Minnesota 55455

DOI: 10.2514/1.52129

The data structures and algorithms of a newly developed three-dimensional direct simulation Monte Carlo program are outlined. The code employs an embedded three-level Cartesian mesh accompanied by a cut-cell algorithm to incorporate triangulated surface geometry. A simple and efficient adaptive mesh refinement algorithm that maintains a local cell size and time step consistent with the local mean-free path and local mean-collision time is detailed, along with a cut-cell algorithm that sorts triangulated surface elements into Cartesian cells and uses a Monte Carlo technique to compute the cut volume. Three-dimensional direct simulation Monte Carlo simulations of hypersonic flow over a 70 deg blunted cone geometry at various angles of attack are presented. For a 0 deg angle of attack, direct simulation Monte Carlo solutions are in excellent agreement with other simulations reported in the literature and with experimental results for density field and surface heat flux. New simulation results are presented for three-dimensional flows over the probe geometry at 10 and 30 deg angles of attack where direct simulation Monte Carlo predictions for density field and heat flux are in close agreement with the experiment. The largest discrepancy is found in the heat flux along the windward forebody (for 30 deg angles of attack), where the simulations overpredict the experimental measurements by, at most, 25%.

Nomenclature

c_r	=	relative collision speed, m/s
e_{ROT}	=	rotational energy associated with a simulation particle, J
e_{VIB}	=	vibrational energy associated with a simulation particle, J
i, j, k	=	Cartesian indices
N	=	number of particles per cell
n	=	number density
$\mathbf{n}$	=	normal vector
N_{real}	=	number of real molecules per cubic mean-free path
N_{samples}	=	number of independent statistical samples
q	=	heat flux, W/cm^2
S	=	uniform scaling factor
$\mathbf{u}_p$	=	molecular velocity vector (v_x, v_y, v_z), m/s
v_x, v_y, v_z	=	molecular velocity components, m/s
x, y, z	=	spatial Cartesian coordinates, m
Δt	=	local simulation time-step size, s
Δx	=	local simulation cell size, m
λ	=	mean-free path, m
ρ	=	mass density, kg/m^3
σ	=	collision cross section, m^2
τ_c	=	mean-collision time, s
ω	=	variable hard-sphere model power law exponent

Subscripts

E	=	maximum coordinates of Cartesian bounding box
O	=	minimum coordinates of Cartesian bounding box
1	=	corresponding to level one cells
2	=	corresponding to level two cells
3	=	corresponding to level three cells
∞	=	freestream value

I. Introduction

DIRECT simulation Monte Carlo (DSMC) is a particle-based numerical method [1] that simulates the Boltzmann equation. As a result, DSMC is an accurate simulation tool for modeling dilute gas flows ranging from continuum to free-molecular conditions. The DSMC method tracks a representative number of simulation particles through a computational mesh, with each simulation particle representing a large number of real gas molecules. A key aspect of the method is that molecular movement and collision processes are decoupled. Specifically, during each simulation time step, all particles are first translated along their molecular velocity vector without interaction. After the movement phase, nearby particles that lie within the same computational cell are collided in a statistical manner. This decoupling is accurate for dilute gas flows as long as the simulation time step remains less than the local mean-collision time ($\Delta t < \tau_c$) and the cell size dimension remains less than the local mean-free path ($\Delta x < \lambda$). Unlike the molecular dynamics technique, precise atomic positions are not integrated using physically realistic interatomic potentials. Rather, simulation particles within each DSMC cell are randomly selected to undergo collisions. This enables the DSMC method to require only a small number of simulation particles per cell ($\approx$20) in order to obtain acceptable collision statistics and an accurate solution to the Boltzmann equation. If the correct collision rate in each cell is specified in addition to the correct probabilities of internal energy exchange and chemical reaction during each collision, the DSMC method can accurately simulate complex dilute gas flows.

The numerical restrictions on cell size and time step become significant for full-scale three-dimensional (3-D) flows at moderate densities where small λ and τ_c require large numbers of computational cells (and thus particles) and many simulation time steps. It has been shown that reducing the cell size and time step significantly below λ and τ_c results in a negligible improvement in accuracy [1–3]. As a result, an ideal DSMC simulation (which maximizes both

Presented as Paper 2010-0450 at the 48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum, Orlando, FL, 4–7 January 2010; received 23 August 2010; revision received 8 January 2011; accepted for publication 21 January 2011.

*Research Associate, Department of Aerospace Engineering and Mechanics, 110 Union Street SE; dagao2008@gmail.com. Senior Member AIAA.

†Research Assistant, Department of Aerospace Engineering and Mechanics, 110 Union Street SE; zhang993@umn.edu. Student Member AIAA.

‡Assistant Professor, Department of Aerospace Engineering and Mechanics, 110 Union Street SE; schwartz@aem.umn.edu. Senior Member AIAA.

accuracy and efficiency) would have every computational cell adapted to λ (each containing $\approx$20 simulation particles) and move all particles for a local time step of τ_c. Typically, $\Delta x \approx \frac{1}{2}\lambda$ and $\Delta t \approx \frac{1}{5}\tau_c$ [1]. For flows involving little variation in density ($\lambda \approx$ constant and $\tau_c \approx$ constant), a uniform mesh and simulation time step is accurate, efficient, and easy to implement. However, many low density flows are associated with high-altitude flight that, in turn, is inherently linked to high flow velocities and significant compressibility effects. Such flows can exhibit orders-of-magnitude variation in density and therefore present a challenge for the DSMC method both in terms of accuracy and in terms of computational efficiency.

Existing state-of-the-art DSMC codes generally adopt one of two approaches for the geometry model. Here, geometry model refers to both the computational mesh and surface representation. For example, the MONACO code [4] uses an unstructured body-fitted quadrilateral or tetrahedral geometry model. Such tetrahedral meshes are easily fitted to complex surface geometries. MONACO is also equipped with a preprocessor that enables the generation of a new mesh that is refined based on the local λ values from a previous solution. In addition, different particle weights and time steps can be specified in each cell. The Icarus code [5] uses a multiblock system of algebraic meshes to define the computational domain. This grid system allows flexibility for capturing local high gradient regions without excessively gridding the entire domain, because there is no requirement for side correspondence between regions [5]. In contrast to body-fitted grids, the second approach is to use a Cartesian-based geometry model. Here, a Cartesian grid is used for the flowfield, and the surface mesh must be cut from the Cartesian grid in order to simulate complex geometries. For example, the DSMC analysis code (DAC) [6] uses a two-level Cartesian grid where each level one (L1) cell can be refined into any number (in each coordinate direction) of level two (L2) Cartesian cells. A preprocessor enables a new mesh to be adapted using a previous flow solution, and different particle weights and time steps can be specified for each L1 cell. The DAC code then employs a cut-cell method that cuts an arbitrary triangulated surface mesh from the two-level Cartesian flowfield grid. Similarly, the SMILE code [7] also employs a two-level Cartesian grid and a cut-cell method for simulating complex surface geometries.

In this paper, a new DSMC implementation called the molecular gas dynamic simulator (MGDS) code, which extends upon the aforementioned DSMC approaches, is described in detail. Specifically, the geometry model chosen for the MGDS code is a three-level Cartesian flowfield grid using a cut-cell method to imbed arbitrary triangulated surface geometries. The MGDS code performs fully automated adaptive mesh refinement (AMR) during a DSMC simulation and automatically sets different time steps in each cell. The main purpose of this paper is to, first, outline the algorithms used for automated AMR, variable local time steps, and the general cut-cell procedure and, second, to test the code on a benchmark planetary probe flow and present new 3-D simulations at angle of attack. The major reasons for development of this particular geometry model (in order of importance) are as follows.

First, the memory required to store cell-node vertices and cell-face normal vectors for unstructured tetrahedral meshes is substantially larger than the memory required to store a multilevel Cartesian grid. For large flowfield meshes (greater than 100 million cells), an unstructured tetrahedral mesh must not only be partitioned across parallel processors during a DSMC simulation, but pre/postprocessing, grid generation, and AMR routines must also be parallelized due to the large memory requirements. A Cartesian geometry model significantly alleviates these problems and may therefore be more suitable for future large-scale DSMC simulations.

Second, a Cartesian geometry model necessitates a cut-cell method in order to handle complex surface geometries. Although this may seem like a drawback, a cut-cell approach is an extremely general and accurate technique for complex geometries [6,8]. As discussed by LeBeau [6], a cut-cell approach allows for complete decoupling between the flowfield and surface mesh discretization. Such decoupling is especially important for the DSMC method and may be necessary for large-scale simulations of near-continuum flows where the mean-free path near a vehicle surface may be orders of magnitude smaller than accurate surface resolution requires. Likewise, decoupling is equally important for low density flows (such as low orbiting satellites) where the local value of λ may be much larger than fine surface geometry features.

Third, the MGDS code employs a three-level embedded Cartesian grid for the flowfield. Adding the third independent level of Cartesian refinement adds considerable flexibility over previous two-level implementations for flows with large density gradients. Precise AMR is essential for both simulation accuracy and computational efficiency.

Fourth, use of a Cartesian geometry model not only results in lower memory storage requirements, but it also enables many DSMC calculations to be performed with fewer floating point operations. Algorithms for initial grid generation, successive AMR, particle movement, and particle re-sorting during AMR require fewer operations when using a Cartesian grid compared with a non-Cartesian grid. The efficiency of such operations, and the degree to which they can be automated, will become increasingly important as larger DSMC simulations are performed, especially for time-accurate unsteady problems where frequent AMR is required.

The paper is organized as follows. In Sec. II, the data structures used to organize geometry, cell, and particle data are described, after which, particle movement algorithms are outlined. In Sec. III, the AMR and variable time-step algorithm, as well as the cut-cell algorithm, are described. Results for 3-D simulations of hypersonic flow over a 70 deg blunted cone geometry at various angles of attack are presented in Sec. IV and compared with experimental measurements. Section V contains a summary of the major aspects of the MGDS code and conclusions drawn from the planetary probe simulations.

II. Data Structures

A. Geometry Data Structure

The geometry data structure used within the MGDS code is depicted in Fig. 1. It consists of a 3-D array of L1 cell structures. Each L1 cell structure contains the coordinates of the bounding Cartesian vertices ($x_0, y_0, z_0, x_E, y_E, z_E$) and a pointer to a 3-D array of L2 cell

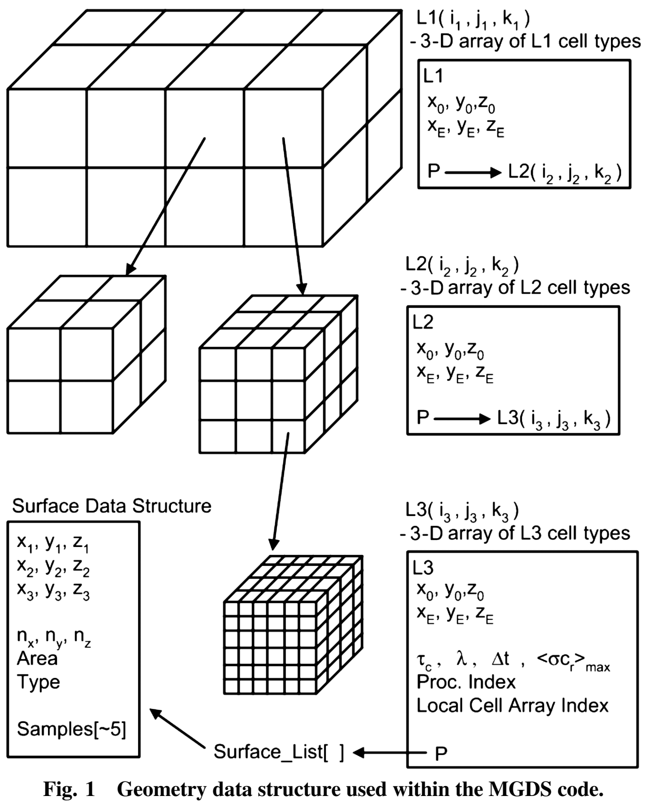


Fig. 1 Geometry data structure used within the MGDS code.

structures. This 3-D array can have arbitrary dimensions, meaning each L1 cell can contain an arbitrary number of L2 cells. Each L2 cell structure is identical to a L1 cell structure, and therefore contains a pointer to a 3-D array of L3 cell structures. While the current MGDS implementation stores vertex coordinates for each L3 cell, it is noted that this information could simply be computed from the L2 vertex information and the Cartesian index of the L3 cell, thereby greatly reducing the memory storage requirement. In addition, each L3 cell structure contains the local values of τ_c, λ, Δt, and a parameter $<\sigma c_r>_{\max}$ that corresponds to the maximum expected collision rate. Also, each L3 cell structure points to the indices of the boundary surfaces that cut it (described in Sec. III.C), which are stored in a global array of surface structures. Each surface structure in the array contains vertex, surface normal, area, surface type, and sampling variables. Finally, each L3 cell structure contains both the processor number and local array index (on that processor) where its particle data and additional cell data are stored. Given the coordinates of a particle, the geometry data structure enables efficient cell indexing of the particle.

The most important aspect of this geometry data structure (Fig. 1) is that it contains all of the information required for the AMR, cut-cell, and variable time-step (movement) procedures while requiring relatively little memory storage. Thus, large portions of the geometry data structure could be stored locally on each processor. This enables each processor to move all of its particles for a complete time step without accessing any data stored on other partitions. Specifically, the destination processor and cell index (refer to Fig. 2) can be updated, even for particles that cross multiple partitions, move with variable time steps, and collide with triangulated surface elements located on a different partition. The result is a compact DSMC loop where all particles are collided and moved for an entire time step in parallel, followed by an interprocessor communication step, where particle data are relinked to their known destination processor and local cell. Such data independence has been demonstrated to be useful for threading the DSMC algorithm over multicore processors using shared-memory parallelization [9].

B. Cell/Particle Data Structure

A DSMC code requires the storage and bookkeeping of a large amount of data. Complex data structures are often employed for data organization and can provide the flexibility required for a general DSMC implementation. Particle data are the dominant contributor to memory; however, the global number of particles, as well as the local number within each cell, can vary greatly within a DSMC simulation. Thus, statically allocated arrays of particles must either be conservatively large (unused memory) or resized often (computationally and memory intensive). Likewise, since the mesh resolution in DSMC depends on the solution itself; ideally, the number of cells would change during a simulation, and data structures should account for this possibility as well.

The MGDS code adopts the cell and particle data structures approach used in the MONACO code [4] with certain modifications. The specific cell/particle data structure used by the MGDS code is depicted in Fig. 2. On each parallel partition, an array of L3 cells is maintained. This array is actually an array of pointers to L3 cell structures, so that if the array requires resizing (during AMR), resizing an array of pointers is much less memory intensive and more efficient than resizing an array of cell structures. An important aspect of the MGDS data structures is that any L3 cell data can be partitioned to any processor. That is, domain decomposition is not limited to L1 or L2 cells, which is essential for load balancing of large parallel simulations. Each cell structure contains substantial data, including an array that stores sampled (cell averaged) data of the particles within the cell. Currently, each cell structure also contains the nine integer indices of the corresponding L3 cell in the geometry data structure. Finally, each cell structure contains pointers to the head and tail of a doubly linked list of particle structures. Each particle structure contains all required particle information (as shown in Fig. 2); in addition, each structure also contains two integers for the processor number and index within the local cell array where the particle is located. Finally, it should be noted that cell-face data and cell-connectivity data are not required, since the grid is Cartesian.

The MGDS code is written in Fortran 90. All data structures are defined as Fortran derived data types, which enable general and efficient packaging of data for broadcast among partitions in a parallel implementation.

C. Particle Movement and Tracking

Ray tracing is a general and efficient method of tracking particle movement through a computational mesh that is widely used in the computer graphics industry and is used by many DSMC codes. The procedure is depicted schematically in Fig. 3 for movement within an unstructured two-dimensional (2-D) triangular mesh. Essentially, the time to hit each face of the current cell ($\Delta t_{\text{hit}-f}$ in Fig. 3) is computed using the particle position/velocity and the face vertices/normal vector, where Δx_f is the normal distance between the particle and cell face. The particle is then advanced for the minimum time $\Delta t_{\text{hit-min}} = \min(\Delta t_{\text{hit}-f})$ and the particle is relocated to the neighboring cell. The process is then repeated for the remaining time ($\Delta t_{\text{sim}} - \Delta t_{\text{hit-min}}$). Of course, if ($\Delta t_{\text{hit-min}} > \Delta t_{\text{sim}}$), then the particle can be moved for the full time step and remains located in the current cell. This procedure is very general; it naturally sorts particles while moving them and naturally allows for variable time steps to be used in each cell.

Within a Cartesian grid, a particle may be moved for the full time step, regardless of how many cells are crossed. The particle's new

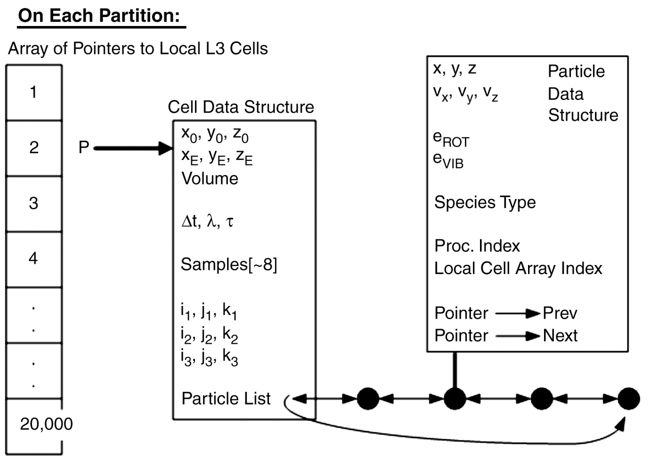
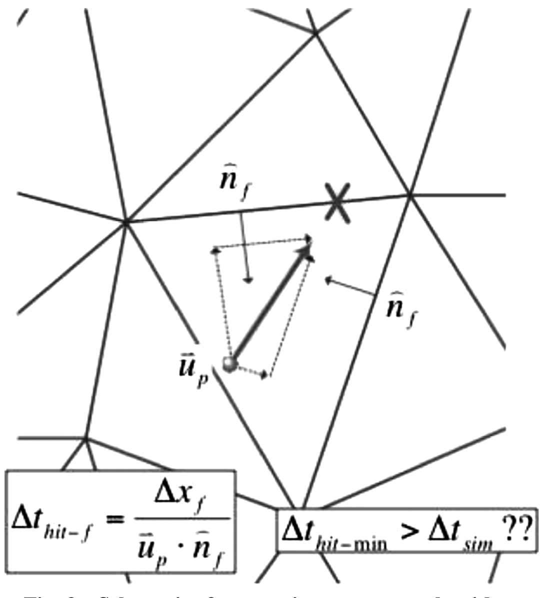


Fig. 2 Cell/particle data structure used within the MGDS code.



$\hat{n}_f$
$\hat{n}_f$
$\vec{u}_p$
$\Delta t_{hit-f} = \frac{\Delta x_f}{\vec{u}_p \cdot \hat{n}_f}$
$\Delta t_{hit-\min} > \Delta t_{sim}$??

Fig. 3 Schematic of ray-tracing movement algorithm.

position can then be re-indexed on the Cartesian grid very efficiently compared with re-indexing on a non-Cartesian grid. However, even on a Cartesian grid, there are three main drawbacks to this move and re-indexing procedure. The first is that variable time steps cannot be used in each cell. This is not a drawback for computing unsteady flows; however, it is a significant drawback when computing steady-state flows, as variable time steps increase the simulation efficiency greatly. The second drawback is that re-indexing on a multilevel Cartesian grid is rather computationally expensive, and it may approach the expense of ray tracing. Third, when simulating flow over complex geometries, ray tracing must be used in cells close to boundary surfaces, thereby requiring the additional complexity of mixed Cartesian-move and ray-trace algorithms.

Although the MGDS code maintains the option of the Cartesian-move procedure, the default is to use the ray-tracing procedure throughout the entire simulation domain. It is important to realize, however, that the ray-tracing algorithm is more efficient on a Cartesian grid than on a non-Cartesian grid, since the dot products (shown in Fig. 3) involve only a single component. Furthermore, a trend in DSMC algorithm research is to enlarge cells above the $\Delta x \approx \lambda$ constraint without loss of accuracy through the use of virtual subcells [10]. The consequence of this trend for 3-D simulations will be that a large majority of simulation particles remain within the same cell during a given time step. On a Cartesian grid, the ray-trace procedure to determine if $\Delta t_{\text{hit-min}} > \Delta t_{\text{sim}}$ is highly efficient. Further quantitative results comparing these two movement procedures can be found in [9].

III. Algorithms

A. Adaptive Mesh Refinement

The AMR algorithm used within the MGDS code inputs the current geometry data structure, generates a new geometry data structure adapted precisely to the local mean-free path, sets local time steps in accordance with the local mean-collision time, and interpolates the maximum expected collision frequency $\langle \sigma c_r \rangle_{\max}$ between old and new meshes. The algorithm then updates the cell/particle data structure as cells are added or removed and, finally, the global sort algorithm re-sorts all particles into the appropriate linked lists in the revised data structure. In this manner, the MGDS simulation continues in a smooth and accurate manner. The AMR procedure is called fewer than 10 times during a typical steady-state MGDS simulation. With the current implementation described next, one call to the AMR function requires approximately the same computational time as 10 simulation time steps, regardless of the size of the simulation. While this has a negligible effect on the overall simulation time for a steady-state solution, future efficiency improvements will be important for unsteady simulations where the AMR procedure might be called every few time steps. An overview of the AMR procedure is detailed next along with a three-level Cartesian grid schematic in Fig. 4.

It is important to note that some L1 and L2 cells may not be refined; that is, as seen in Fig. 4, it is possible for L3 = L2 = L1 or for L3 = L2 ≠ L1. Also, in the current implementation, L1 cell sizes remain fixed during the simulation. Thus, the AMR procedure is applied within each L1 cell, independently of all other L1 cells, according to the following steps. First, the maximum value of λ ($\lambda_{\max}$) is determined in the given L1 cell by looping through all L3 cells contained within it. The new L2 cell size is then set equal $\lambda_{\max}$, and a new Cartesian L2 cell grid is generated within the L1 cell. At this point, a loop is performed over all new L2 cells to determine $\lambda_{\min}$ in each. This involves determination of which old L3 cells intersect with each of the new L2 cells. On a 3-D Cartesian grid, computing the volume of intersection between two generic cells is straightforward and efficient. Within each new L2 cell, the new L3 cell size is then set to $\lambda_{\min}$. Finally, properties such as λ, τ_c, and $\langle \sigma c_r \rangle_{\max}$ are interpolated (using a volume average) from the old L3 grid to the new L3 grid contained within a given L1 cell. This information can then be used to accurately set an appropriate new local time step and maintain the local collision rate. At this stage, after processing all L1 cells via the preceding steps, the complete geometry data structure is now updated. Then, the new geometry data structure is now used to update the pointer array of cell/particle structures (Fig. 2), as well as modify cell data. Note that in adding/removing cells and changing the dimensions/vertices of cells, the particles linked within these cells may now be completely unsorted. Finally, the coordinates of all particles are re-indexed using the new geometry data structure. The global sort algorithm is then called, which adds/removes particle pointers to/from the correct linked lists. The MGDS simulation is now ready to carry on with the general move/collide DSMC algorithm.

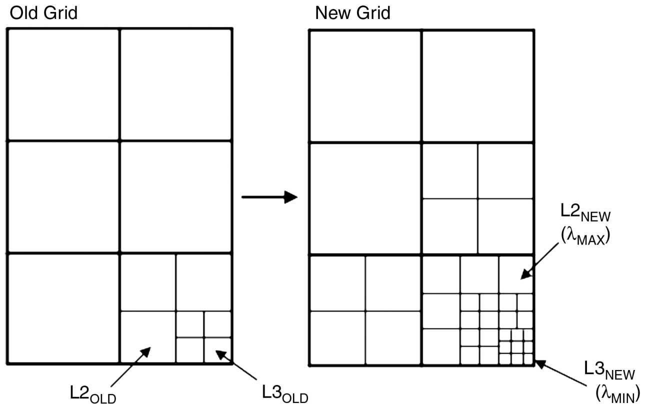


Fig. 4 Schematic of three-level Cartesian mesh and AMR procedure.

B. Variable Local Time-Step Algorithm

Precise AMR should naturally provide some control in maintaining a constant (and optimal) number of particles per cell; that is, as the density increases, the cell size is reduced proportionally via AMR. However, if one considers the number of real gas molecules located within a volume of one cubic mean-free path N_{real}, the following dependence is realized:

$$N_{\text{real}} = n\lambda^3, \qquad n \sim \rho, \qquad \lambda \sim 1/\rho \tag{1}$$

where n is the number density with units of particles per cubic meter. If constant simulation particle weights are used, the number of simulation particles per cell should therefore scale with $1/\rho^2$ in 3-D (and $1/\rho$ in 2-D). For example, if the density drops by one order of magnitude, although the number of molecules per unit volume drops equally, the volume under consideration rises by three orders of magnitude. As stated earlier, only 20 particles per cell are required for accuracy, and using additional particles per cell is inefficient. Thus, even with precise AMR for 3-D simulations, large variations in the number of particles per cell will occur naturally in a DSMC simulation. This results in either an inaccurate simulation with too few particles in portions of the domain or a very inefficient simulation using far more particles than required for statistical accuracy.

As described by Kannenberg and Boyd [11], the use of a variable time step in each cell significantly alleviates this problem. Compared with a global reference time step Δt_{ref}, the time step used to advance particles within a given cell is increased or decreased by a factor that varies from cell to cell ($\Delta t = \Delta t_{\text{ratio}} \times \Delta t_{\text{ref}}$). This factor is equal to the ratio of the local mean-collision time to the global reference time step, $\Delta t_{\text{ratio}} = S \times \tau_c / \Delta t_{\text{ref}}$, and can be scaled with a globally constant factor S. At the same time, the weight for all particles within the cell is also multiplied by Δt_{ratio}. Essentially, where τ_c (and therefore λ) is large, particles move more rapidly through these larger cells. During a given instant, this reduces the number of particles found in that cell and, by adjusting the particle weight by the same factor, the number density in the cell remains correct. Since $\tau_c \sim 1/\rho$, the combination of variable time steps and AMR improves the scaling for the number of particles per cell to $N \approx$ constant in 2-D and $N \sim 1/\rho$ in 3-D. As detailed in [11], the new time step, number of particles, and particle weight are used in each cell to compute collision rates and outcomes without any modification to the DSMC algorithm.

The MGDS code is able to set a variable time step in each L3 cell. In addition, this local time step can be slightly adjusted to account for imprecise AMR. Even when using a three-level adaptive grid, locally $\Delta x \neq \lambda$ precisely. In the MGDS code, the local time-step factor is set as $\Delta t_{\text{ratio}} = S \times (\tau_c/\Delta t_{\text{ref}}) \times (\Delta x/\lambda)$. Thus, if a given cell is slightly too small, the time step is lowered such that additional particles will accumulate in this smaller cell. It should be noted that adjusting the local time step in this manner is accurate as long as $\Delta x/\lambda \approx 1$, which is the case when using a three-level Cartesian grid with AMR. The combination of precise AMR and precise local time-step adjustment leads to a much more uniform distribution of particles per cell within MGDS simulations [12].

Varying the particle weight in each cell independent of the time step is an alternate method of controlling the number of particles per cell that are used in existing DSMC codes, and are discussed in [11]. Here, particles are either cloned or deleted in order to obtain the desired number of particles per cell, and their weights are adjusted accordingly. This technique enables precise control of the number of particles per cell and, in contrast with the variable time-step approach outlined previously, can also be employed for unsteady flows. While the deletion of simulation particles is not thought to influence solution accuracy, the cloning of particles may correlate statistics and result in random walk errors [1]. Although techniques exist to minimize such errors, the combination of AMR and local time steps should greatly reduce the degree to which particles are cloned/destroyed.

C. Cut-Cell and Cut-Volume Algorithms

Since the MGDS code already uses the ray-tracing technique to determine particle intersections with arbitrary cell faces, triangulated surface meshes can be naturally imbedded within the flowfield grid. The cut-cell algorithm performs two main functions. The first is to read in a list of triangular surface elements (generated by various commercial surface triangulation packages) and sort all surface elements into the appropriate L3 cells within the geometry data structure. The second main function is to compute the volume of each cut cell required to determine the collision rate and various macroscopic properties in the cut cell. Both functions involve moderately complex geometrical calculations, but they leave the basic DSMC data structures and algorithms completely unchanged.

To sort a list of triangular surface elements into the list of L3 Cartesian cells, the MGDS code employs a cut-cell intersection technique initially detailed in computer graphics literature and more recently adapted for CFD simulations by Aftosmis et al. [8]. The technique is able to use computationally efficient bitwise operations and comparisons to determine intersections between triangular elements and Cartesian cells. Further details on computing the intersection between generally positioned triangles in 3-D are contained in [8].

Once all surface elements are sorted into appropriate L3 cells, the volume of each of these cut cells is computed. The MGDS code uses a simple Monte Carlo technique to compute cut volumes. Coordinates are chosen at random within the uncut Cartesian cell, and each coordinate is determined to lie within the body or within the flowfield. The average error in such a volume calculation scales directly as $1/\sqrt{N_{\text{samples}}}$. However, it should be noted that for very small cut volumes, the percentage error in computed volume could be substantially larger. At the same time, the flow through such small cut volumes is likely free molecular; thus, error in the cell volume (and therefore collision rate) may not influence the simulation. The procedure to determine which side of the surface a Monte Carlo coordinate P lies is as follows.

First, choose one cut-cell Cartesian vertex Q that is known to lie within the flowfield. Second, determine the number of intersections that the line segment $P - Q$ has with all triangulated surface elements stored within the Cartesian cut cell. The procedure to determine an intersection between a line segment and a triangular surface element is already used for the cut-cell intersection techniques [8] discussed previously. Third, if the number of intersections for line segment $P - Q$ is even, then Monte Carlo coordinate P lies within the flowfield. If odd, P lies outside of the flowfield, within the body surface. Note that it is possible that none of the Cartesian vertices lie within the flowfield, yet the cell is still a cut cell. In this case, the point Q is selected as one of the vertices that is known to lie within the body surface, and the logic regarding even and odd numbers of intersection is reversed. Fourth, the cut volume is then simply computed as the full volume of the uncut Cartesian cell multiplied by the ratio of Monte Carlo coordinates determined to lie within the flowfield to the total number of Monte Carlo coordinates considered.

Although more computationally expensive and, possibly, less accurate than more direct cut-volume methods [8], this Monte Carlo method is general, simple to implement, and robust for the most complex configurations, including the case when a single 3-D Cartesian cell is cut by a large number of small triangulated surface elements. One important limitation of the current cut-volume implementation in the MGDS code is that split cells (where the surface mesh actually splits a Cartesian cell into two distinct flow volumes with substantially different properties) are not yet accounted for.

The cut-cell algorithm is called at the beginning of each simulation and immediately following each call to the AMR function. The current cut-cell algorithm requires the same computational expense as approximately 100 simulation time steps for the planetary probe simulations presented in Sec. IV. Therefore, although more expensive than the AMR process, it still does not contribute significantly to the overall simulation time. Finally, inflow, outflow, symmetry, or wall surfaces may be specified by the user on all sides of the overall Cartesian bounding box of the simulation. These outer surfaces are added to the appropriate L3 cells in addition to the triangulated surface elements sorted by the cut-cell algorithm. An example of the triangulated surface mesh and adapted three-level Cartesian grid for the 30 deg angle-of-attack planetary probe simulation discussed in the next section is shown in Fig. 5. Here, the grid on the symmetry plane is shown, as well as on a normal plane that cuts through the probe geometry near the leeward shoulder.

IV. Hypersonic Flow Simulations Over a Planetary Probe

The planetary probe is a 70 deg blunted cone geometry, detailed in Fig. 6, where s is the distance around the surface of the probe beginning at the stagnation point. Experimental data were obtained in the SR3 wind tunnel in Meudon, France, for a 5-cm-diam probe. The freestream conditions investigated in this paper correspond to case 1

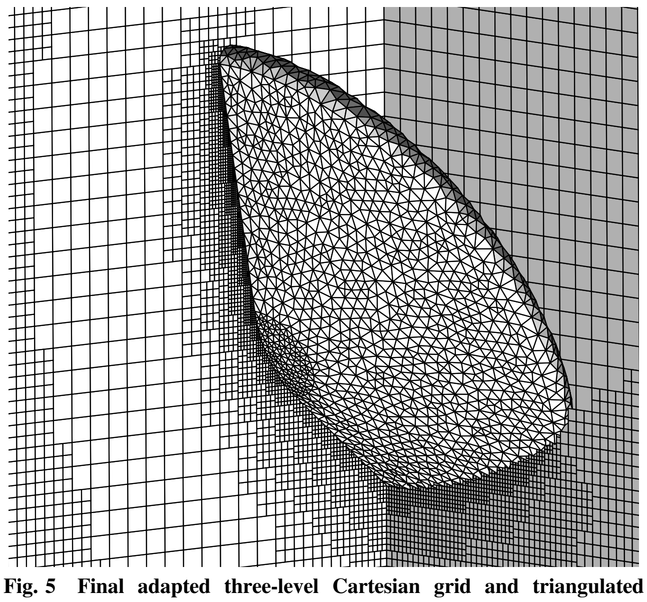


Fig. 5 Final adapted three-level Cartesian grid and triangulated surface mesh for a planetary probe simulation.

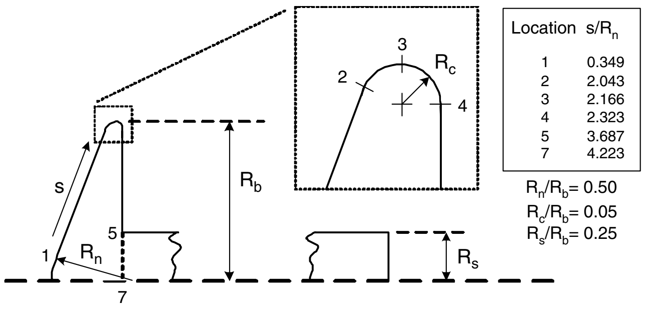


Fig. 6 Planetary probe geometry.

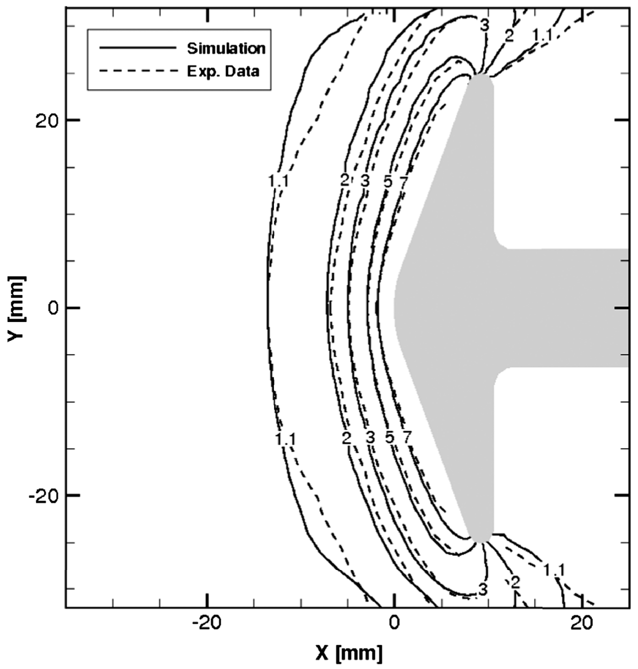
of the experiments (nitrogen gas at a Mach number of 20.2, with freestream density and temperature of 1.73×10^{-5} kg/m^3 and 13.3 K, respectively). Experimental measurements for density fields, heating rates, and integrated aerodynamic properties are detailed in [13–15], and a survey of numerical simulation results for this geometry are summarized in [16]. To the authors' knowledge, only simulation results for case 1 at a 0 deg angle of attack (axisymmetric) appear in the literature, and angle-of-attack (3-D) simulations corresponding to case 1 conditions have not yet been reported.

A. Density Field

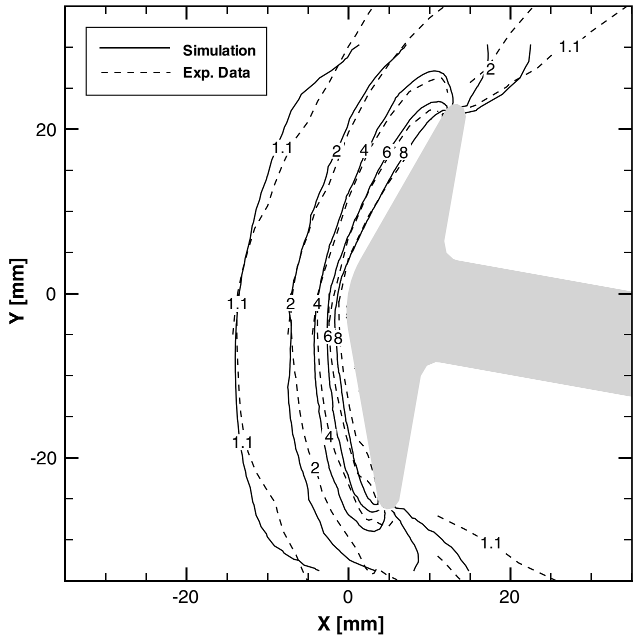
Experimental measurements were made of the density field at both 0 and 10 deg angles of attack using the electron beam fluorescence technique [13]. Since the experiment used a conical nozzle, this resulted in nonuniform freestream density with a variation of roughly 20% from the stagnation point to the shoulder of the probe. To adjust for this, the density field was also measured in the absence of the planetary probe model [13]. The normalized density contours, defined as the ratio of the density field with the model to that without the model, are plotted in Fig. 7 along with MGDS simulation results. The MGDS simulations assume uniform freestream conditions as well as diffuse reflection and full thermal accommodation to a wall temperature of 300 K, consistent with other numerical simulations reported in the literature [16]. The variable hard-sphere collision model is used with a power law value of $\omega = 0.75$. The Larsen–Borgnakke model [17] is used for translational–rotational energy exchange, together with the variable rotational energy exchange probability model of Boyd [18], using a maximum rotational collision number 18.1 and a reference temperature for rotational energy exchange of 91.5 K. Vibrational energy is not considered.

Unless specified otherwise, each MGDS simulation begins with a uniform Cartesian mesh that is sized to $\Delta x = 0.6\lambda_\infty$, a uniform time step set to $\Delta t = 0.2\tau_{c\infty}$, and the particle weight is set such that each freestream cell contains approximately 10 particles. The simulation reaches steady state after approximately $1000\Delta t$, after which the solution is sampled for approximately $2000\Delta t$ in order to obtain a sufficiently smooth density (and λ) field. At this point, the AMR function is called, which refines the mesh, sets all local time-step values, re-sorts all particles in the new mesh, resets all sample values, and then continues with the simulation. Since the freestream cells are already properly sized, the global particle weight remains unchanged. The addition of refined cells and modification of local time steps causes an unsteady period where solution adjusts to the new mesh and the number of simulation particles increases. When the number of particles approaches a new constant value (a new steady state is reached), the solution is again sampled for approximately $2000\Delta t$ and the AMR function is called again. For the simulations presented in this paper, after three calls to the AMR function, the mesh no longer changes significantly. The final solutions are then sampled for $3000\Delta t$, and the resulting local cell sizes and time steps are verified to be quite close to 0.6λ and $0.2\tau_c$. The most important aspect of the MGDS simulation methodology is that, automatically, a complex geometry is cut from a three-level Cartesian mesh that is precisely refined during the DSMC simulation. In this manner, an accurate and efficient solution is obtained with no user intervention.

DSMC simulation results for case 1 conditions at a 0 deg angle of attack have been reported by Moss et al., where although the agreement with the experiment is good, there is a noticeable shift between the measured and computed density fields (seen in Fig. 6 of [19]). We report that our MGDS simulation agrees very well with the simulation of Moss et al.; therefore, our results also show a shift between measured and computed density fields for case 1 at a 0 deg angle of attack. It is important to note that the uncertainty in the position of density measurements is actually 1–1.5 mm (on the order of the electron beam diameter) relative to the stagnation point of the model [13]. In Fig. 7a, the experimental density field is first translated by 0.65 mm in the negative x-coordinate direction (away from the stagnation point) and then overlaid with the MGDS solution. This amount of translation is selected in order that the experimental and



a) Case1 conditions at 0 deg angle of attack



b) Case1 conditions at 10 deg angle of attack

Fig. 7 Contours of density normalized by density field in the absence of the probe model. Comparison of MGDS prediction with experimental measurement.

computed contour line, $\rho_{\text{norm}} = 1.1$, coincide on the symmetry axis. With this translation (below the experimental uncertainty), the MGDS prediction is now in excellent agreement with the experiment.

MGDS simulation results are compared with experimental measurement (with no translation) for the 10 deg angle-of-attack case in Fig. 7b. Overall, good agreement is found within experimental uncertainty, with better agreement on the leeward forebody than on the windward forebody. Despite the experimental data being normalized by the density field in absence of the model, the discrepancy (more noticeable near the shoulder of the probe) could certainly still be caused by the nonuniform density field in the experiment compared with the uniform freestream boundary conditions used in the MGDS simulation.

B. Heat Flux

Heat flux measurements, obtained using the thin wall technique, are detailed in [14]. When compared on a logarithmic scale, as seen in Fig. 8, good agreement with previous DSMC simulations [16] and with experimental results are found along the forebody and sting for a 0 deg angle of attack. The heat transfer on thermocouples five and six was too low to be measured and was set arbitrarily to 0.002 $\mathrm{W/cm^2}$ in the experimental data set [14].

Figure 9 compares the windward forebody heat flux predicted by MGDS simulations with experimental measurements for 0 and 10 deg angles of attack. Figure 10 shows a number of simulation results for a 30 deg angle of attack when certain numerical parameters are varied. It is noted that no heat flux measurements were taken on the leeward portion of the model. In Figs. 9 and 10, the results are no longer plotted on a logarithmic scale, enabling a clear determination of the agreement between simulation and experiment. For 0 and 10 deg angles of attack (Figs. 9a and 9b), close agreement between simulation and experiment is found. For the 10 deg case, the stagnation point and location of maximum heat flux has moved slightly off the axis according to the simulation results. The experiment is unable to resolve this due to the spacing of the thermocouples.

For the 30 deg case (Figs. 10a–10c), the maximum heat flux is now found near the probe shoulder. This trend is also seen in the experimental measurements; however, the simulation overpredicts the heat flux by up to 25% along the forebody cone. The reason for this discrepancy is unclear, although it may be a result of the nonuniform freestream conditions in the experiment. To ensure that the simulation results are independent of numerical parameters, a convergence study is performed. Specifically, the variation in heat flux around the planetary probe surface resulting from a change in the average number of particles per cell, the ratio of local cell size to mean-free path, and the ratio of local time step to mean-collision time is shown in Figs. 10a–10c, respectively. The baseline simulation result can be considered as the solid line in Fig. 10a. This solution is obtained in the same manner as the 0 and 10 deg cases, where all cells are adapted to $\Delta x = 0.6\lambda$, local time steps are set to $\Delta t = 0.2\tau_c$, and the particle weight is set to obtain approximately 10 particles ($N = 10$) in each freestream cell. For the 30 deg case, this particle weight results in approximately 40 particles per cell in the stagnation region.

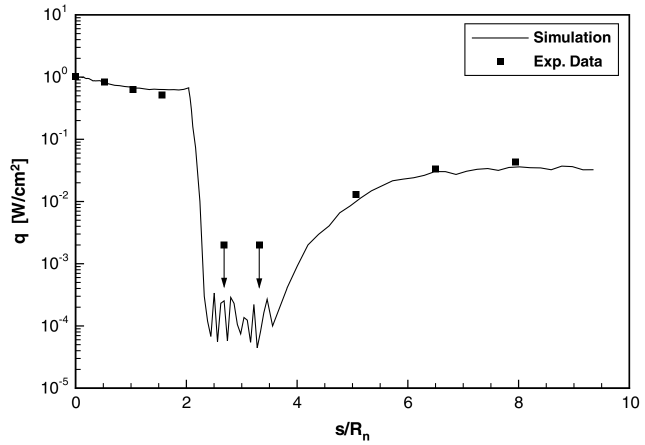
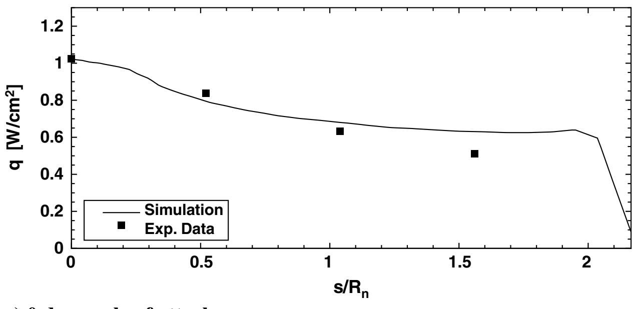
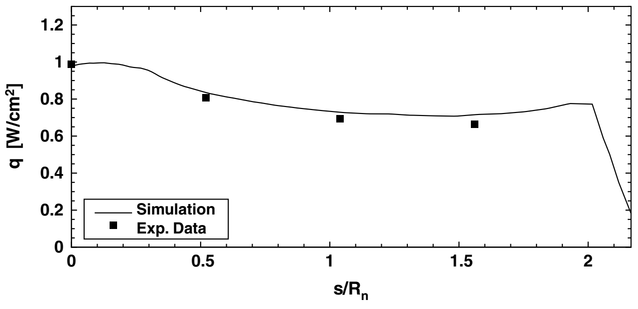


Fig. 8 MGDS prediction for heat flux compared with experiment at 0 deg angle of attack.



a) 0 deg angle of attack



b) 10 deg angle of attack

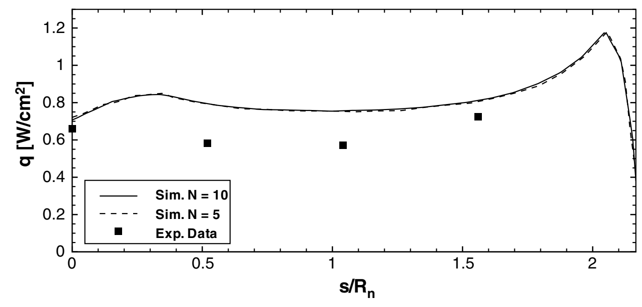
Fig. 9 Comparison of MGDS predictions for heat flux with experiment for 0 and 10 deg angles of attack.

To determine how a variation in the number of particles per cell affects the heat flux solution, the particle weight is exactly doubled. All other simulation parameters (cell sizes and time steps) remain unchanged. This results in approximately five simulation particles ($N = 5$) in each freestream cell and, therefore, half the number of particles in each cell compared with the baseline simulation. As seen in Fig. 10a, the heat flux changes by a negligible amount, indicating that further increasing the number of simulation particles is unlikely to affect the solution.

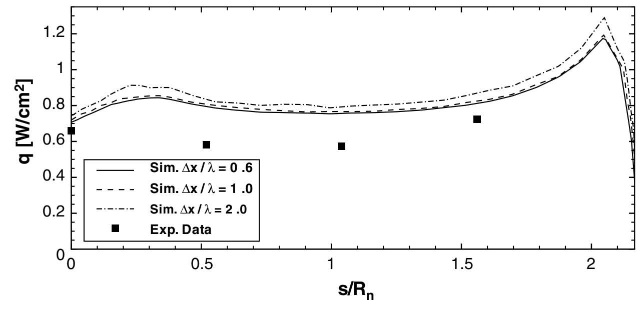
Next, the particle weight is fixed (at the value used for the $N = 10$ solution in Fig. 10a), local time steps are maintained at $\Delta t = 0.2\tau_c$, and only the cell size is varied. Specifically, the scaling factor used in the AMR routine is set such that cells are refined to $\Delta x = 0.6\lambda$, 1.0λ, and 2.0λ. As seen in Fig. 10b, while the heat flux result is noticeably higher for the 2.0λ grid, there is essentially no variation in the heat flux between 1.0λ and 0.6λ grids. This indicates that refining the grid beyond 0.6λ is unlikely to affect the solution.

Finally, the particle weight is fixed (at the value used for the $N = 10$ solution in Fig. 10a), cells are adapted locally to $\Delta x = 0.6\lambda$, and only the local time step is varied. Specifically, the scaling factor S used by the AMR routine is set such that the local time step in each L3 cell is $\Delta t = 0.2\tau_c$, $0.5\tau_c$, and $1.0\tau_c$. As seen in Fig. 10c, while the heat flux result is noticeably higher for the $1.0\tau_c$ simulation, there is little variation in the heat flux between $0.5\tau_c$ and $0.2\tau_c$ simulations. This indicates that lowering the local time step below $0.2\tau_c$ is also unlikely to affect the solution.

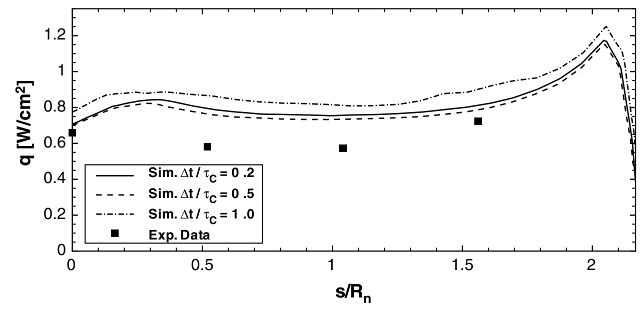
The parameter convergence study demonstrates that the baseline values for particle weight, cell size, and time step are appropriate. In fact, it is evident that the baseline heat flux result can be obtained using far fewer cells and particles, and larger time steps (fewer iterations). Ensuring both the accuracy and efficiency of DSMC simulations has been the focus of recent research [20–22]. Specifically, the mean-collision separation (MCS), defined locally as the average distance separating collision partners, has been proposed as a parameter that should be minimized within a DSMC simulation for accuracy [22]. A variety of techniques may be used to this end, and



a) Heat flux variation due to number of particles per cell



b) Heat flux variation due to cell size



c) Heat flux variation due to time-step size

Fig. 10 Comparison of MGDS predictions for heat flux with experiment for 30 deg angle of attack.

the minimum MCS can be achieved by performing a nearest-neighbor search for all collision pairs. However, when such techniques are employed, there may be new restrictions on the simulation time step, and multiple collisions between the same collision pair must be restricted [22]. The L3 cells in the MGDS code are precisely adapted flowfield cells required by any general DSMC simulation for storing macroscopic sampled values and for accurately computing local collision rates. In the current MGDS implementation, particles are randomly chosen to collide within each L3 cell, and thus the MCS is on the order of the local L3 cell size. However, techniques to further reduce the MCS [22], such as nearest-neighbor collisions or creating smaller collision subcells, can naturally be incorporated into the MGDS framework.

V. Conclusions

In this paper, the data structures and algorithms of a newly developed 3-D DSMC program, called the MGDS code, are outlined. The code employs an embedded three-level Cartesian grid accompanied by a cut-cell algorithm to incorporate triangulated surface geometry into the adaptively refined Cartesian grid. This geometry model is selected for its low memory storage requirements, its ability to decouple surface and flowfield discretizations, and its increased computational efficiency of the particle movement procedures over non-Cartesian grids. A simple and efficient 3-D AMR algorithm that maintains local cell size and time step consistent with the local mean-free path and local mean-collision time is detailed. Additionally, a cut-cell algorithm is outlined that sorts an arbitrary list of triangulated surface elements into local Cartesian cells and computes the cut volume using a Monte Carlo technique.

MGDS solutions are obtained for rarefied hypersonic flow over a 70 deg blunted cone geometry at various angles of attack. For a 0 deg angle of attack (axisymmetric), the MGDS solutions are in excellent agreement with other DSMC simulations reported in the literature and with experimental results for density field and surface heat flux. New simulation results are presented for 3-D flow over the probe geometry at 10 and 30 deg angles of attack. The MGDS predictions for density field and heat flux for these 3-D flows are in close agreement with the experiment. The largest discrepancy is found in the heat flux along the windward forebody (for 30 deg angle of attack), where the simulations overpredict the experimental measurement by up to 25%. This discrepancy may be due to the fact that the experiment used a conical nozzle, which resulted in a nonuniform freestream density that varied by approximately 20% over the radius of the probe geometry.

All numerical results presented in this paper are performed with fully automated AMR and cut-cell algorithms, with no user intervention required, ensuring efficient and accurate DSMC simulations.

Acknowledgments

This work is partially supported by a seed grant from the University of Minnesota Supercomputing Institute. This work is also supported by the U.S. Air Force Office of Scientific Research (AFOSR) under grant number FA9550-04-1-0341. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official polices or endorsements, either expressed or implied, of the AFOSR or the U.S. Government.

References

[1] Bird, G. A., *Molecular Gas Dynamics and the Direct Simulation of Gas Flows*, Oxford Univ. Press, New York, 1994, pp 208, 213.

[2] Rader, D. J., Gallis, M. A., Torczynski, J. R., and Wagner, W., "DSMC Convergence Behavior of the Hard-Sphere-Gas Thermal Conductivity for Fourier Heat Flow," *Physics of Fluids*, Vol. 18, No. 7, 2006, Paper 077102.
doi:10.1063/1.2213640

[3] Gallis, M. A., Torczynski, J. R., and Rader, D. J., "DSMC Convergence Behavior for Transient Flows," 39th AIAA Thermophysics Conference, Miami, FL, AIAA Paper 2007-4258, June 2007.

[4] Dietrich, S., and Boyd, I. D., "Scalar and Parallel Optimized Implementation of the Direct Simulation Monte Carlo Method," *Journal of Computational Physics*, Vol. 126, No. 2, 1996, pp. 328–342.
doi:10.1006/jcph.1996.0141

[5] Otahal, T. J., Gallis, M. A., and Bartel, T. J., "An Investigation of Two-Dimensional CAD Generated Models with Body Decoupled Cartesian Grids for DSMC," 34th AIAA Thermophysics Conference, Denver, CO, AIAA Paper 2000-2361, June 2000.

[6] LeBeau, G. J., "A Parallel Implementation of the Direct Simulation Monte Carlo Method," *Computer Methods in Applied Mechanics and Engineering*, Vol. 174, Nos. 3–4, 1999, pp. 319–337.
doi:10.1016/S0045-7825(98)00302-8

[7] Ivanov, M. S., Markelov, G. N., and Gimelshein, S. F., "Statistical Simulation of Reactive Rarefied Flows: Numerical Approach and Applications," 7th AIAA/ASME Joint Thermophysics and Heat Transfer Conference, Albuquerque, NM, AIAA Paper 1998-2669, June 1998.

[8] Aftosmis, M. J., Berger, M. J., and Melton, J. E., "Robust and Efficient Cartesian Mesh Generation for Component-Based Geometry," *AIAA Journal*, Vol. 36, No. 6, 1998, pp. 952–960.
doi:10.2514/2.464

[9] Gao, D., and Schwartzentruber, T. E., "Optimizations and OpenMP implementation for the direct simulation Monte Carlo method," *Computers and Fluids*, Vol. 42, No. 1, 2011, pp. 73–81.
doi:10.1016/j.compfluid.2010.11.004

[10] LeBeau, G., Jacikas, K., and Lumpkin, F., "Virtual Sub-Cells for the Direct Simulation Monte Carlo Method," 41st Aerospace Sciences Meeting and Exhibit, Reno, NV, AIAA Paper 2003-1031, Jan. 2003.

[11] Kannenberg, K. C., and Boyd, I. D., "Strategies for Efficient Particle Resolution in the Direct Simulation Monte Carlo Method," *Journal of Computational Physics*, Vol. 157, No. 2, 2000, pp. 727–745. doi:10.1006/jcph.1999.6397

[12] Gao, D., Zhang, C., and Schwartzentruber, T. E., "A Three-Level Cartesian Geometry Based Implementation of the DSMC Method," 48th AIAA Aerospace Sciences Meeting, Orlando, FL, AIAA Paper 2010-0450, 2010.

[13] Allegre, J., Bisch, D., and Lengrand, J. C., "Experimental Rarefied Density Flowfields at Hypersonic Conditions over 70-Degree Blunted Cone," *Journal of Spacecraft and Rockets*, Vol. 34, No. 6, 1997, pp. 714–718. doi:10.2514/2.3300

[14] Allegre, J., Bisch, D., and Lengrand, J. C., "Experimental Rarefied Heat Transfer at Hypersonic Conditions over 70-Degree Blunted Cone," *Journal of Spacecraft and Rockets*, Vol. 34, No. 6, 1997, pp. 724–728. doi:10.2514/2.3302

[15] Allegre, J., Bisch, D., and Lengrand, J. C., "Experimental Rarefied Aerodynamic Forces at Hypersonic Conditions over 70-Degree Blunted Cone," *Journal of Spacecraft and Rockets*, Vol. 34, No. 6, 1997, pp. 719–723. doi:10.2514/2.3301

[16] Moss, J. N., and Price, J. M., "Survey of Blunt Body Flows Including Wakes at Hypersonic Low-Density Conditions," *Journal of Thermophysics and Heat Transfer*, Vol. 11, No. 3, 1997, pp. 321–329. doi:10.2514/2.6252

[17] Larsen, P. S., and Borgnakke, C., "Statistical Collision Model for Monte Carlo Simulation of Polyatomic Gas Mixture," *Journal of Computational Physics*, Vol. 18, No. 4, 1975, pp. 405–420. doi:10.1016/0021-9991(75)90094-7

[18] Boyd, I. D., "Rotational-Translational Energy Transfer in Rarefied Nonequilibrium Flows," *Physics of Fluids A*, Vol. 2, No. 3, 1990, pp. 447–452. doi:10.1063/1.857740

[19] Moss, J. N., Price, J. M., Dogra, V. K., and Hash, D. B., "Comparison of DSMC and Experimental Results for Hypersonic External Flows," 30th Thermophysics Conference, San Diego, CA, AIAA Paper 1995-2028, June 1995.

[20] Bird, G. A., Gallis, M. A., Torczynski, J. R., and Rader, D. J., "Accuracy and Efficiency of the Sophisticated Direct Simulation Monte Carlo Algorithm for Simulating Noncontinuum Gas Flows," *Physics of Fluids*, Vol. 21, No. 1, 2009, Paper 017103. doi:10.1063/1.3067865

[21] Gallis, M. A., Torczynski, J. R., Rader, D. J., and Bird, G. A., "Convergence Behavior of a New DSMC Algorithm," *Journal of Computational Physics*, Vol. 228, No. 12, 2009, pp. 4532–4548. doi:10.1016/j.jcp.2009.03.021

[22] Gallis, M. A., and Torczynski, "Effect of Collision-Partner Selection Schemes on the Accuracy and Efficiency of the Direct Simulation Monte Carlo Method," *International Journal for Numerical Methods in Fluids* [online], 2010. doi:10.1002/fld.2409

I. Boyd
Associate Editor