Adaptive Mesh Refinement and Cut-cell Algorithms for DSMC Simulation of Hypersonic Flows

A THESIS

SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL

OF THE UNIVERSITY OF MINNESOTA

BY

Chonglin Zhang

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

Master of Science

May, 2010

Acknowledgements

A lot of people should be thanked for their kindness help during my study at AEM department over the past two years:

My advisor:

Dr. Thomas Schwartzentruber

My master's thesis committee members:

Dr. Graham Candler and Dr. Sean Garrick

Classmates and friends:

Hao Dang, Yintao Song, Xiaochuan Chai, Paul Norman, Xian Chen, Zhijiang Ye, and many other AEM classmates.

The most thanks belong to my parents and my sibings. Thank you all for your continuous support and encouragement.

Abstract

Adaptive mesh refinement (AMR) and cut-cell algorithms were developed for a 3-level Cartesian mesh based Direct Simulation Monte Carlo (DSMC) implementation. The simple and efficient AMR algorithm adapts the cell size to the local mean free path of the flow field. Variable time step technique was implemented together with the AMR algorithm to set a time step consistent with the local mean collision time. The control of simulation particles through the use of variable time step was also illustrated. The cut-cell method decouples the flow field Cartesian mesh and the triangulated surface mesh representing any object inside the flow field. Two key aspects of the cut-cell method: cut-cell sorting and volume calculation were discussed in detail. The 3-level embedded Cartesian mesh combined with AMR and variable time step allows increased flexibility for precise control of local mesh size and time step, both vital for accurate and efficient DSMC simulation. Hypersonic flow simulations were conducted to highlight the performance of AMR, variable time step and cut-cell algorithms. Three dimensional simulation of Planetary probe reproduced the experimental heat flux measurement.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Direct Simulation Monte Carlo (DSMC) is a particle-based numerical method [1, 2, 3] that simulates the Boltzmann equation [2, 4]. 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 process are decoupled.

The general procedures of the DSMC method are listed below:

(1) Initially, generate particles in the whole simulation domain.

(2) In each time step, generate new particles on flow boundaries. The number of newly generated particles will be determined by the mass flux of the flow. The molecular velocities and internal energies are sampled from equilibrium distributions in function of velocity $\vec{v}$, temperature T and density ρ.

(3) Move particles in a straight line manner to their new positions. During the particle movement, check if the particle will encounter with a boundary, if this happens, go to (4)

(4) (i) For a wall, specify the velocity and internal energy of the particle according to the type of wall and move the particle further to its new position.

(ii) For a flow boundary, delete the particle.

(5) Select particles within the same computational cell to collide with each other. The collision rate will be calculated based on the flow field information.

(6) Modify the particle information to post-collision value, based on the specific collision model and energy involved in the collision.

(7) Sample to get macroscopic flow field quantities.

(8) Go to (2), and repeat the above process.

To implement the DSMC method, a three-level Cartesian flow field grid, utilizing a cut-cell method to embed arbitrary triangulated surface geometries is used in this thesis [5]. The data structure shown in Fig 2.1 contains a uniform background grid of level-1 (L1) cells. Each L1 cell can be refined into an arbitrary number of level-2 (L2) cells. Finally, each L2 cell can be further refined into an arbitrary number of level-3 (L3) cells. Under such a grid structure, the program can perfom fully automated adaptive mesh refinement (AMR) during a DSMC simuation and automatically set different time steps in each L3 cell. The major reasons for development of this particular geometry model (in order of importance) are as follows [5].

(1) 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 multi-level Cartesian grid. For large flow field meshes ($>$ 100 million cells), an unstructured tetrahedral mesh must not only be partitioned across parallel processors during a DSMC simulation, but pre/post-processing, 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.

(2) 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, 7]. As discussed by LeBeau [6], a cut-cell approach allows for complete decoupling between the flow field 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 λ 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.

(3) Current code employs 3-level embedded Cartesian grid for the flow field. Adding the third independent level of Cartesian refinement adds considerable flexibility over previous 2-level implementations for flows with large density gradients. Precise AMR is essential for both simulation accuracy and computational efficiency.

(4) The use of a Cartesian geometry model not only results in lower memory storage requirements, but 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 to 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.

In the next two chapters, Chpater 2 and Chapter 3, the implementiaon details of AMR, variable time step, and cut-cell method based on the current three-level Cartesian grid structure are discussed.

Chapter 2

Adaptive Mesh Refinement and Variable Time Step

This chapter first outlines the two main restrictions of the DSMC method, which must be followed for accuracy. Next, the adaptive mesh refinement (AMR) and variable time step techniques implemented in this thesis are given in detail. A sufficient number of simulated particles in each cell (usually ≥ 20 per cell) is required for accurate results. On the other hand, cells with too many simulated particles will affect the efficiency of the simulation. The strategy on controlling the number of particles in each simulation cell (≈ 20 per cell, ideally) to achieve both accuracy and efficiency of the simulation will be discussed.

2.1 Restrictions for Accurate DSMC Simulation

In DSMC simulation, two main restrictions should be followed to obtain accurate results:

(i) The size of the simulation cells h used to group particles should be in the same order as the local mean free path λ;

(ii) The time step Δt used in the simulation should be in the same order as the local

mean collision time τ.

While (i) allows only particles with distance in the order of the local mean free path λ to have a chance to collide, (ii) validates the decoupling of particle movement and collisions. In practice, the cell size is chosen to be $h = \frac{\lambda}{2}$, and the time step used in the simulation is chosen to be $\Delta t = \frac{\tau}{5}$.

In this thesis, the Variable-Hard-Sphere (VHS) gas model [8] is used to model the particle collision cross section and calculate the mean free path λ and the mean collision time τ. The VHS gas model corresponds to the following power law gas viscosity relation for continuum flow, and is widely used in DSMC simulation.

$$\mu = \mu_{ref}(\frac{T}{T_{ref}})^{\omega}, \quad \mu_{ref} = \frac{15\sqrt{\pi m k T_{ref}}}{2\pi d_{ref}^2(5-2\omega)(7-2\omega)} \tag{2.1}$$

The expressions for λ and τ in terms of VHS gas model are:

$$\lambda = \frac{m}{\sqrt{2}\pi d_{ref}^2 \rho}(\frac{T}{T_{ref}})^{\omega-\frac{1}{2}} \tag{2.2}$$

$$\tau = \frac{m}{4 d_{ref}^2 \rho \sqrt{\frac{\pi T}{m}}}(\frac{T}{T_{ref}})^{\omega-1} \tag{2.3}$$

The numerical simulations presented in this thesis use Nitrogen gas N_2 and Argon Ar, with parameters used in VHS model listed in Table 2.1.

Gas	$T_{ref}(K)$	$d_{ref}(m)$	ω
N_2	273.0	4.17×10^{-10}	0.75
Ar	273.0	4.17×10^{-10}	0.81

Table 2.1: VHS model parameters for N_2 and Ar

From Eq. 2.2 and Eq. 2.3, it can be seen that the mean free path λ and the mean collision time τ are inversely proportional to density ρ. $\lambda \propto \frac{1}{\rho}$, $\tau \propto \frac{1}{\rho}$. For hypersonic flow, the strong shock wave and boundary layer will result in large density and temperature variation, which means large variation of the mean free path λ and the mean

collision time τ in the flow field. In high density regions, the mean free path λ and the mean collision time τ would be small, hence a small cell size and time step should be used in these regions for accuracy of the result; while in low density regions, a large cell size and time step would be enough to guarantee both accuracy and efficiency of the simulation.

It can be seen from Eq. 2.2 and Eq. 2.3 that temperature T also affects the mean free path λ and the mean collision time τ for the VHS gas model. For molecular nitrogen N_2, $\lambda \propto T^{0.25}$, while $\tau \propto \frac{1}{T^{0.25}}$. The mean free path λ is proportional to temperature T, while the mean collision time τ is inversely proportional to temperature T. However, it should be noted that the dependence of λ and τ on T is weaker than on ρ for N_2, as can be seen from Eq. 2.2 and Eq. 2.3.

To follow the restrictions on DSMC simulation, one way is generating a mesh that is consistent with the density and temperature distribution of the flow field prior to the simulation, and using a fixed time step that is less than the smallest value of the mean collision time τ. However, the detailed information of the flow field cannot be completely obtained prior to the simulation, since the mesh and the time step should be coupled to the solution.

Thus, the use of adaptive mesh refinement (AMR) to regenerate the mesh during simulation, and variable time step technique to assign different time step for each cell, is desired for DSMC simulation. This process should ideally be fully automated, requiring no user time or intervention. The implementation of AMR and variable time step technique will be introduced in detail in the next two sections.

2.2 Adaptive Mesh Refinement

Different approaches of performing AMR exists in both Computational Fluid Dynamics (CFD) and DSMC, which differs on the grid structures used (hexahedral, tetrahedral, cubic, etc.) and the ways of refining an old mesh to form a new mesh [9, 10, 7, 11, 6].

A DSMC mesh does not require alignment with shocks or strong gradients as required for accurate CFD simulation. The mesh used in DSMC simulation is mainly for grouping particles, collision partners selection and sampling information of particles to get macroscopic flow field property. Thus, we can take advantage of existing AMR approaches developed for CFD method, while keeping the resulting data structure of the mesh (Cartesian-based) simple and easy to implement.

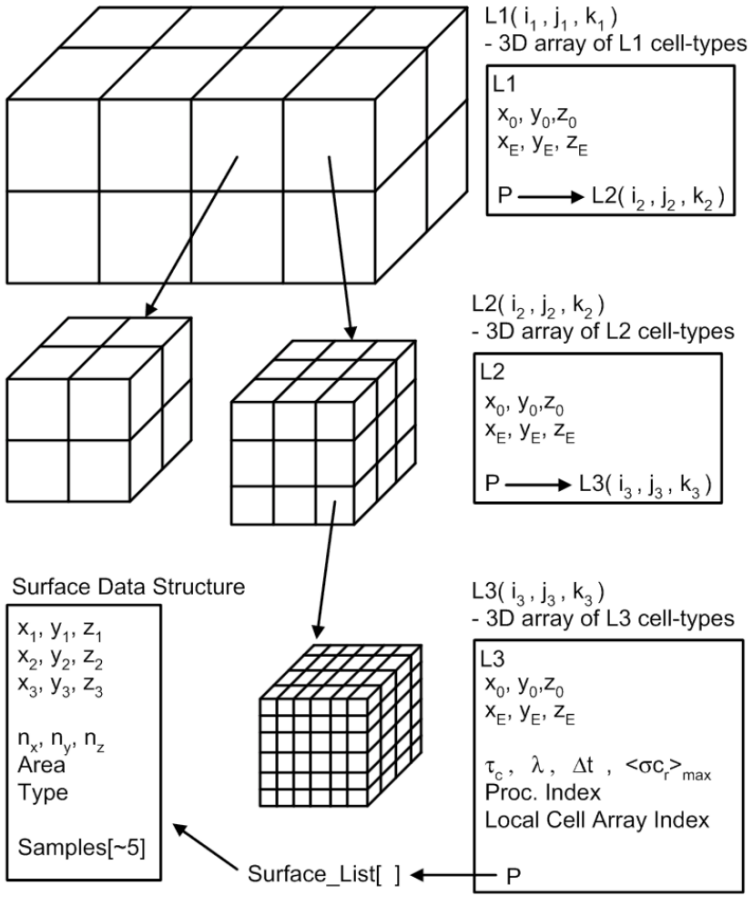


Figure 2.1: Geometry data structure used for the three level Cartesian mesh

2.2.1 Procedures of Adaptive Mesh Refinement

The program developed here uses a three-level Cartesian grid hierarchy. The geometry data structure used for the three-level Cartesian mesh is shown in Fig 2.1. (It should be noted that different L1 (L2) cell can have a L2 (L3) cell with different size, as can be seen from Fig 2.1.)

Cell and particle data are stored in a seperate Cell/Particle data structure shown in Fig 2.2. This data structure must be partitioned for large simulations due to large memory requirements. The data structure is adopted from the cell and particle data structures approach used in MONACO code [12] with certain modifications.

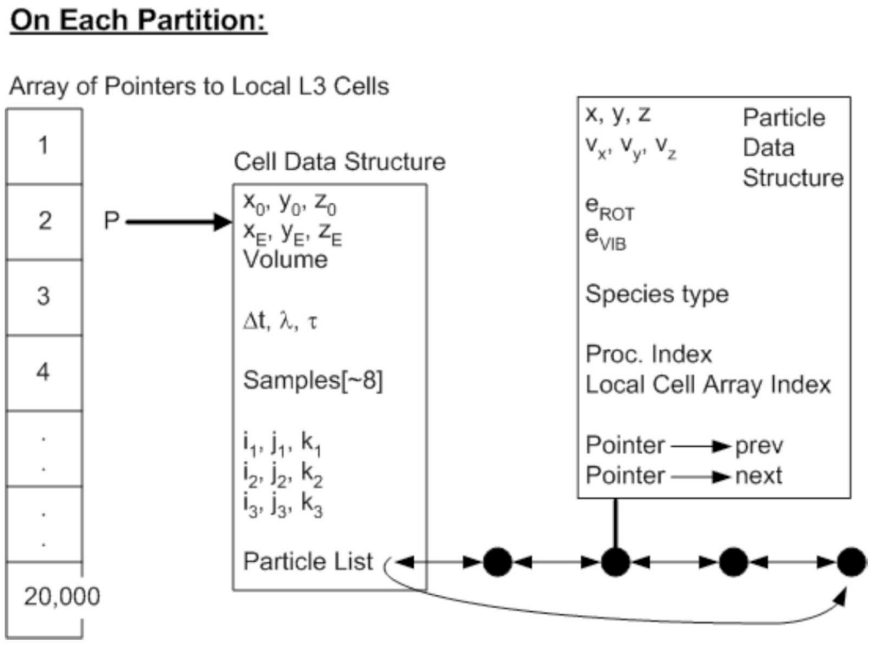


Figure 2.2: Cell/Particle data structure

The DSMC program starts with a uniform mesh. After the program runs on the uniform mesh for a certain time and reaches a steady state, the macroscopic flow field information required for AMR, including density ρ and temperature T, will be sampled

in each cell. The mean free path λ and the mean collision time τ will then be calculated based on T and ρ for each cell using Eq. 2.2 and Eq. 2.3. After the flow field information is acquired, the AMR algorithm will be called to generate a new mesh based on the mean free path λ. The cell size of the newly generated mesh is adapted to the local mean free path λ. During the generation of new mesh, information such as the mean collision time τ and the maximum collision cross section ($\sigma_{cr,max}$) for each cell will also be interpolated from the old mesh, which will be used later to set local time step and select collision partner in each cell. The particles stored in old cells will also be sorted to proper new cells to continue the simulation after AMR.

After AMR is finished, the simulation will continue with the new mesh. The AMR algorithm will be called again based on temperature T and density ρ obtained on that particular mesh structure, until a converged mesh is obtained and the cell size is adapted to a fraction of the local mean free path (usually, the cell size is adapted to $\frac{1}{2}$ of the local mean free path λ).

During AMR, the L1 cell size remains unchanged and only L2 cell size and L3 cell size are changed. Because of the use of a three level mesh, the mesh flexibility can still be obtained for flows with large variation in density even when L1 cell size remains fixed. As a result of a fixed L1 cell size during AMR operation, the AMR algorithm can be performed on each L1 cell independently, which leaves the flexibility of parallelization of the current AMR algorithm.

The detailed procedures of the current AMR algorithm, are listed as follows:

(1) Based on the density ρ, temperature T in each L3 cell, calculate the mean free path λ for each cell using Eq. 2.2.

(2) Loop over all L3 cells within each L1 cell, find the maximum mean free path λ_{max}.

(3) Set the new L2 cell size to $C_h\lambda_{max}$ and generate a uniform new L2 Cartesian grid within each L1 cell (here, C_h is a constant to satisfy the restriction on cell size, usually $C_h = \frac{1}{2}$ as discussed before).

(4) Loop over each new L2 cell and find the minimum mean free path λ_{min} within the new L2 cell. λ_{min} within the new L2 cell will be obtained from λ of all the old L3 cells which intersect with the new L2 cell.

(5) Set the new L3 cell size within each new L2 cell to $C_h\lambda_{min}$.

(6) Obtain the λ, τ and $\sigma_{cr,max}$ for each new L3 cell by weighted average of volume of intersection from all old L3 cells which intersect with the new L3 cell. The interpolated information obtained here will be used to set a new local time step Δt for each L3 cell in Section 2.3. It should be noted that on a Cartesian mesh, the calculation of volume of intersection is straight forward.

(7) Repeat (1) through (6) for each L1 cell.

(8) The entire new Geometry data structure as shown in Fig. 2.1 is now complete.

(9) Use the new Geometry data structure to update the Cell/Particle data structures and add, remove, or modify cell data as required.

(10) Sort particles from the old Cell/Particle data structure to the new Cell/Particle data structure based on their position relative to the new cell. Again, the sorting is straight-forward for Cartesian mesh.

(11) Return from AMR, and resume the simulation using the new mesh.

Fig 2.3 shows a sketch of a 2 dimensional three-level Cartesian mesh before and after AMR, based on the current algorithm.

Due to the decoupling of the background flow field Cartesian mesh and the triangulated surface mesh representing objects inside the flow field, the relative position of the surface mesh and the flow field Cartesian mesh should be determined explicitly to specify the flow field boundary. This is achieved by the cut-cell method and will be discussed in Chapter 3 in detail.

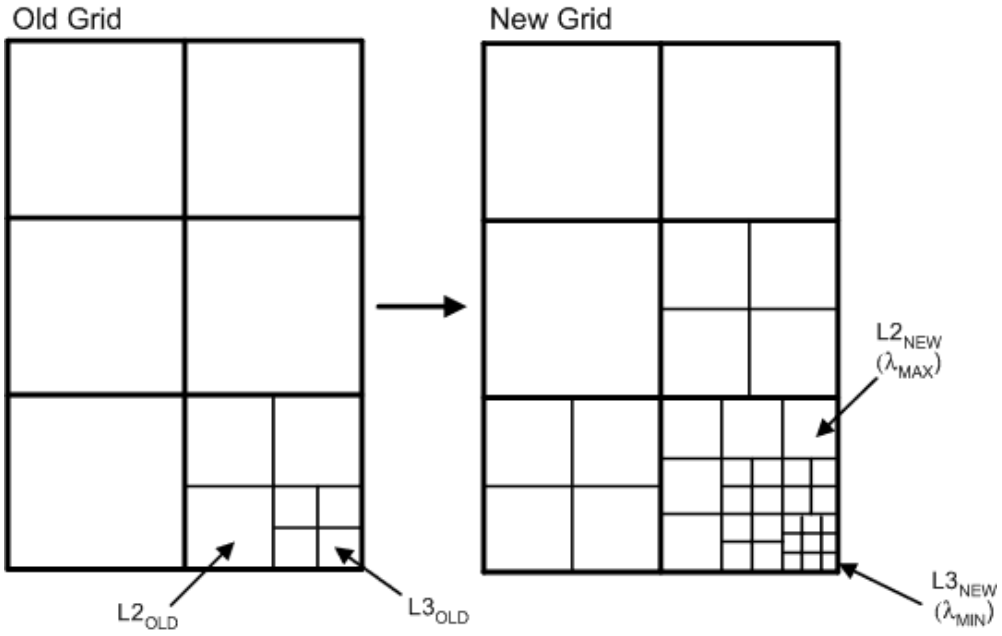


Figure 2.3: The 3-level Cartesian mesh and AMR procedure in 2 dimension

2.2.2 Impact of AMR on Simulation Results

A simulation with condition stated in Table 2.2 is conducted to compare the effect of AMR on simulation results. This is the same simulation as the one that will appear in Chapter 4, Section 4.1. Flow with free stream Mach number $M_\infty = 15$ past a vertical wall, which act as a blunt body, will result in a bow shock in front of the wall. While near the wall region, a boundary layer will be formed. The resulting flow field has large variation in both density and temperature. The AMR is desired to achieve a well resolved flow field mesh. The simulated flow fields without and with the use of AMR are plotted in Fig 2.4 and Fig 2.5, respectively.

Gas	M_∞	T_∞ (K)	ρ_∞ (kg/m^3)	V_∞ (m/s)
N_2	15	80	$6.5 * 10^{-6}$	2735.5

Table 2.2: Simulation condition for flow past a blunt body

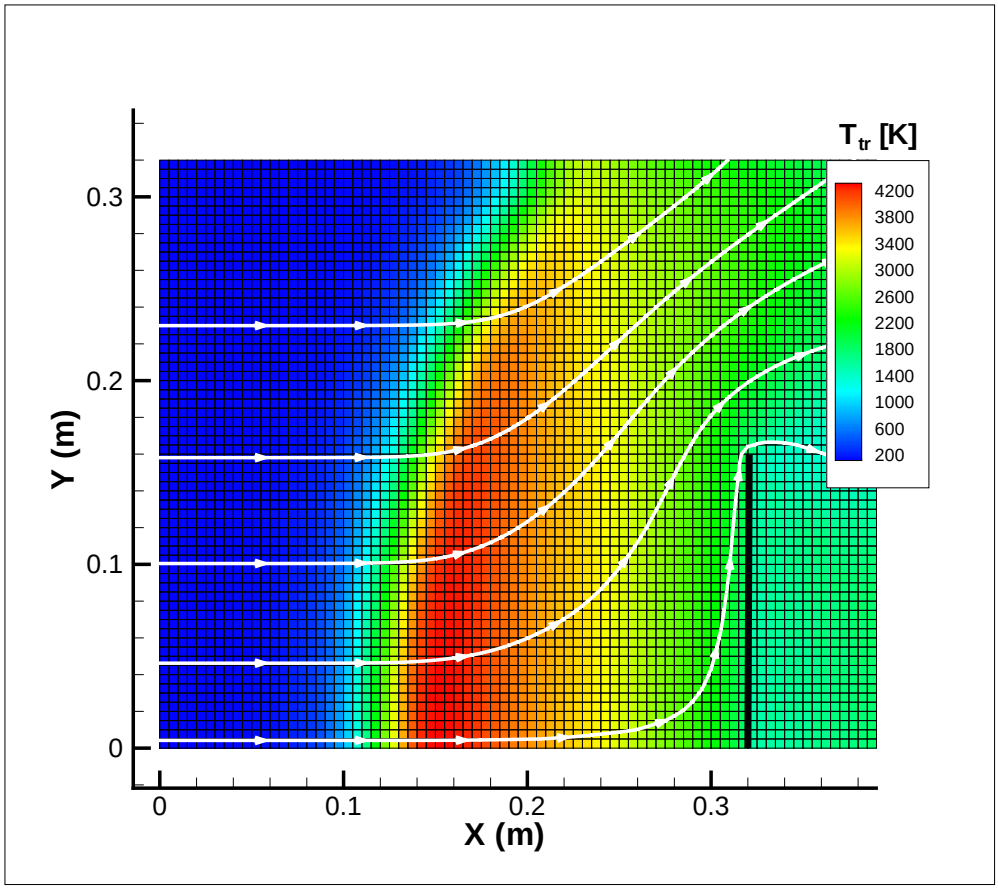


Figure 2.4: Translational temperature without the use of AMR

For both cases, the initial mesh size (cell size) are set according to the free stream mean free path λ_∞. As a result, the mesh size near the wall region is larger than the local value of the mean free path λ, due to high density in the near wall region. It should be noted that the initial mesh size of the flow can be set according to the mean free path near the wall region. However, this value is not known a priori and will result in a small mesh size for regions far from the wall and thus too many cells used in the simulation.

Without the use of AMR, due to the fact that the mesh size in the near wall region is larger than ideal value, the result will not be accurate, in the sense that the restrictions

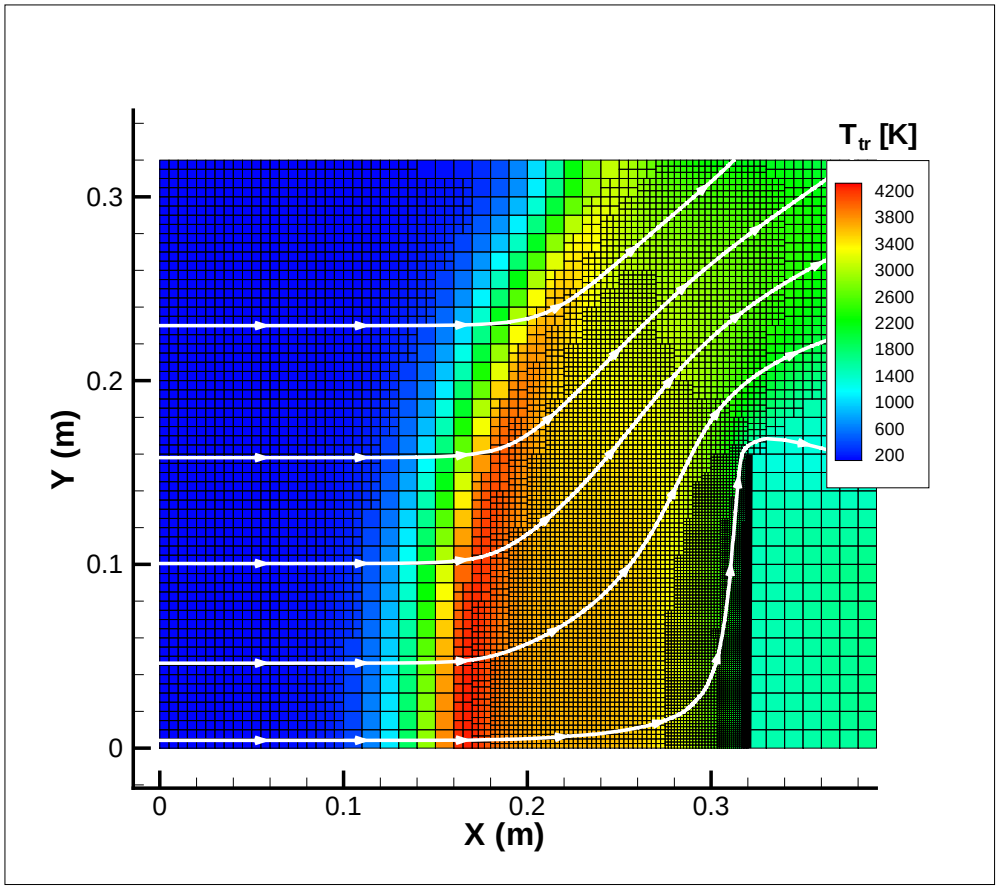


Figure 2.5: Translational temperature with the use of AMR

on both cell size and local time step are not satisfied. From the comparison between Fig 2.4 and Fig 2.5, we can clearly see that the position of the bow shock moves toward the wall with the use of AMR, which is more accurate.

Quantitatively, the change of shock stand-off distance will affect the surface property, such as heat flux q. To verify the statement, a simulation of Mach 20.2 nitrogen N_2 flow past a planetary probe is conducted. This is the same simulation as that will be detailed in Chapter 4, Section 4.2. Experimental results [13] of heat flux on the Planetary Probe surface exist for this case, which were obtained in the SR3 wind tunnel in Meudon, France for a 5-cm diameter probe. The Planetary Probe is a 70° sphere-blunted cone

geometry shown in Fig 2.6 [14], where s is the distance around the surface of the probe beginning at the stagnation point.

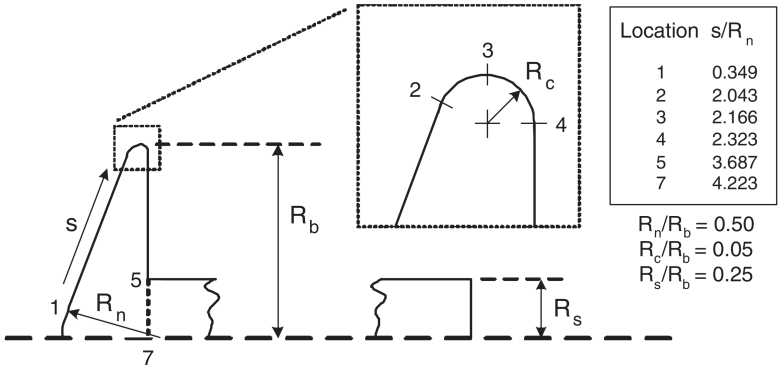


Figure 2.6: Geometry of the planetary probe

Position (s/R_n)	0	0.416	0.832	1.248
SR3 Experiment	1.023	0.839	0.633	0.512
DSMC without AMR	1.424	1.240	1.146	1.084
Error	39.2%	47.8%	81.0%	112.1%
DSMC with AMR	1.0173	0.780	0.690	0.643
Error	-0.56%	-7.0%	9.0%	25.6%

Table 2.3: Heat flux q along the Planetary probe surface (unit is in W/cm^2)

For the case with angle of attack $\alpha = 0°$, the calculated heat flux q at 4 points along the probe are compared with experimental results in Table 2.3. The position of the 4 points are based on their distance from the stagnation point, and are normalized by the radius of the nose R_n. In Table 2.3, the percentage error are based on the difference between simulation values and experimental values, and are normalized by experimental value. Without the use of AMR, the calculated heat flux will be higher than the experimental value, while with the use of AMR, the calculated heat flux is consistent with experiment value. Clearly, the requirement of $h < \lambda$ must be met for accuracy (here h is the mesh size). Also, a comparision of Fig. 2.4 and Fig. 2.5 reveals

a factor of 30 variation in mesh size h between stagnation and wake regions. This highlights the necessity of a multi-level Cartesian mesh for simulation efficiency.

2.3 Variable Time Step

The variable time step is required to maintain the restriction on time step for each cell. Besides this, the introduction of variable time step also has a positive effect on the control of simulated particles in each cell [15], which will be discussed in the next section, section 2.4. The variable time step is achieved by the introduction of a time step ratio for each cell. Initially, the time step Δt_{ref} used for each cell is the same, which is usually determined based on the free stream flow mean collision time τ_∞:

$$\Delta t_{ref} = C_t \tau_\infty \tag{2.4}$$

where, C_t is a specified constant, and usually $C_t = \frac{1}{5}$ as discussed in Section 2.1

The ideal time step of a cell with mean collision time τ would be

$$\Delta t = C_t \tau \tag{2.5}$$

Hence, the time step ratio t_r for a cell would be determined simply by

$$t_r = \frac{\Delta t}{\Delta t_{ref}} \tag{2.6}$$

By substitution of Eq. 2.3 into Eq. 2.6, the time step ratio t_r for a cell can be expressed in terms of temperature T and density ρ. For VHS gas model, this is:

$$t_r = \frac{\rho_\infty}{\rho} (\frac{T}{T_\infty})^{\omega - 1} \tag{2.7}$$

Instead of moving particles in the time scale Δt_{ref}, the particle movement will be in the time scale $t_r \Delta t_{ref} = \Delta t$. For cells with $t_r > 1$, the movement of particle will be "sped-up ", while for cells with $t_r < 1$, particle movement will be "slowed down ".

2.4 Control of Particle Distribution/Number

For an accurate DSMC simulation, besides the time step and cell size restrictions, the particles in each cell will also affect the accuracy of the simulation result. A minimum particle number for each cell should be satisfied, which is usually around $10-20$, while in the same time, the particle number in each cell should be kept as small as possible to achieve simulation efficiency. As the time required for the simulation will increase with increasing number of particles.

2.4.1 The Scaling of Number of Particles in Each Cell

Assume that we want the number of simulated particles in each cell to be $N_{p,ref}$ (usually, $N_{p,ref} = 10-20$), then we can get the reference particle weight (one simulated particle represents $W_{p,ref}$ real molecules) as,

$$W_{p,ref} = \frac{n_{ref} h_{ref}^3}{N_{p,ref}} \tag{2.8}$$

n_{ref} is the reference number density, usually we choose n_{ref} to be the number density that is representative of the flow field, such as the free stream flow number density n_∞. h_{ref} is a reference cell size, and typically is chosen based on the free stream mean free path λ_∞.

To avoid cloning or deleting of particles, the reference particle weight $W_{p,ref}$ must be held constant through the entire domain and over the entire simulation. In this thesis, $W_{p,ref}$ will be kept as a constant for all the cells for the simulation.

For flow field with small density change, thus small variation of number density across the flow field, the number of particles N_p in each cell will be around the reference value $N_{p,ref}$. However, for problems with very large density variation, the number of particles N_p in a cell may be far from the ideal value $N_{p,ref}$. Assume that after AMR, the cell sizes have been adapted to the ideal value ($\approx C_h\lambda$). Also suppose that the density, number density, temperature, number of particles and volume of a cell are ρ,

n, T, N_p, V, respectively. Then we have

$$N_p = \frac{nV}{W_{p,ref}} \tag{2.9}$$

where ideally $V = h^3 = (C_h\lambda)^3$ (this is achieved by AMR).

For a cell with informations that are the same as the reference states we choose, the number of particles is $N_{p,ref}$:

$$N_{p,ref} = \frac{n_{ref}V_{ref}}{W_{p,ref}} = \frac{n_{ref}h_{ref}^3}{W_{p,ref}} = \frac{n_{ref}(C_h\lambda_{ref})^3}{W_{p,ref}} \tag{2.10}$$

From the above expressions, Eq. 2.9 and Eq. 2.10, we can get:

$$\frac{N_p}{N_{p,ref}} = \frac{\rho_{ref}^2}{\rho^2}(\frac{T}{T_{ref}})^{3\omega-\frac{3}{2}} \tag{2.11}$$

It can be seen from Eq. 2.11 that the number of particles will decrease for cells with large density. This is due to the fact that although the number density increases linearly with density, the cell volume (equals to $(C_h\lambda)^3$) decreases with the cube of the density. Also note that temperature will also affect the number of particles in each cell. Suppose that a cell with density ρ and temperature T equal to reference values ρ_{ref} and T_{ref} has the desired amount of particles $N_{p,ref}$, then cells with density $\rho > \rho_{ref}$ will have $N_p < N_{p,ref}$ if the temperature keeps the same as reference value T_{ref}. This means that the number of particles in a cell with density larger than the reference value will be smaller than the ideal value $N_{p,ref}$, and the accuracy of the simulation cannot be guaranteed.

A direct way to solve this would be keeping the number of particles around the ideal value $N_{p,ref}$ for cells with the largest value of density ρ. With this modification, the number of particles in cells with small density will now be much larger than the ideal value, which will affect the efficiency of the simulation.

2.4.2 Effect of Variable Time Steps on Number of Particles

With the introduction of variable time step, besides the effect of maintaining the restriction on time step, it also helps to partially solve the problem faced here of getting

a nearly uniform distribution of simulated particles in each cell.

By introducing a variable time step [15] for each cell, the particle weight for each cell is changed to:

$$W_p = t_r W_{p,ref} = \frac{\rho_\infty}{\rho}(\frac{T}{T_\infty})^{\omega-1} W_{p,ref} \tag{2.12}$$

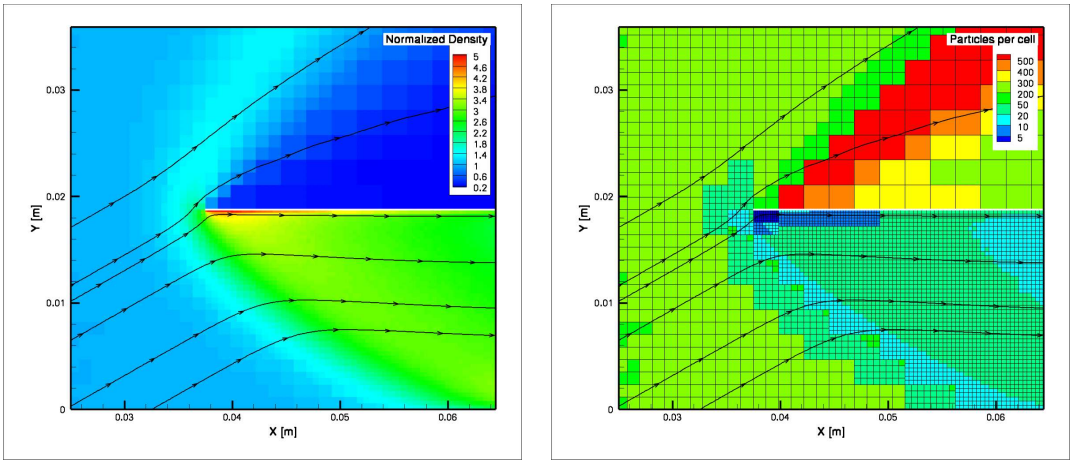
Essentially, if the time step is increased by a factor t_r, then less particles will be found in that cell at any instant. As a result, the weight of each particle must be increased by the same factor t_r in order to maintain the correct density. Using Eq. 2.9 with $W_{p,ref}$ replaced by W_p and Eq. 2.10, we can get the number of particles in each cell relative to reference value $N_{p,ref}$ as below.

$$\frac{N_p}{N_{p,ref}} = \frac{\rho_{ref}}{\rho}(\frac{T}{T_{ref}})^{2\omega-\frac{1}{2}} \tag{2.13}$$

Compared to Eq. 2.11, it can be seen that, with the use of variable time step, the effect of density on simulated number of particles in each cell is reduced.

A Mach 5 flow of Argon over a 10 cm flat plate at 30 degrees of attack is simulated to steady-state with and without the use of variable time steps [5]. The free-stream density and temperature are $7.5 \times 10^{-5} kg/m^3$ and $200K$ respectively. The temperature of the flat plate is held fixed at $T_{wall} = 2000K$ on the bottom and $T_{wall} = 200K$ on the top. The Variable Hard-Sphere (VHS) collision model is used with a power law value of $\omega = 0.81$, and diffuse reflection and full thermal accommodation are imposed for surface collisions. The simulation is three-dimensional stretching approximately $4\lambda_\infty$ in the z direction. However, since symmetry boundary conditions are imposed on the z boundary planes, the resulting flow is uniform in the z coordinate direction and only the two-dimensional flow field results are presented.

Fig 2.7(a) shows the resulting density field (normalized by the free-stream density) when a constant simulation time step is used. The 3-level Cartesian grid has been adapted to the local mean free path and the constant time step is less than $\frac{1}{2}\tau$ everywhere, thus producing an accurate solution. It is evident from Fig 2.7(a) that the density increases by a maximum of 5 times between the free-stream and the leading edge



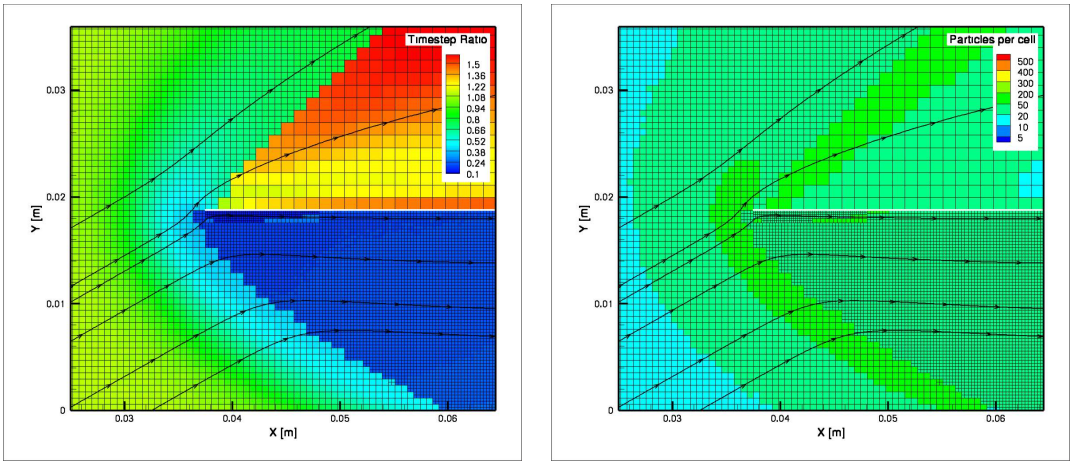
(a) Normalized density field.

(b) Number of simulated particles per cell.

Figure 2.7: Flat plate simulation results without the use of variable time step

of the plate. Although the density (and therefore the number density n) increases towards the plate leading edge, the cell size decreases proportionately. Thus AMR should naturally provide some control in maintaining a constant number of particles per cell. However, as evident from Fig 2.7(b), even with AMR, the average number of simulated particles per cell varies widely from less than 10 to more than 500. This observation is consistent with the scaling of particles per cell from Section 2.4.1.

With the use of variable time step, the number of simulated particles can be reduced substantially for this problem. Fig 2.8(a) shows the time step ratio used in each L3 cell for the flat plate problem. A time step ratio $t_r > 1$ in the region above the flat plate will speed up the particles, and thus, reduce the simulated particles per cell in this region. The combination use of AMR and variable time step, lead to a much more uniform distribution of particles per cell as seen in Fig 2.8(b). The flow field solution, surface heating rate, and surface shear stress profiles obtained when using the variable time step technique are verified to reproduce the results obtained using a constant simulation time step which required substantially more particles. The simulation results are in very close agreement with identical simulations of Bird [2].



(a) Local time step ratio t_r in each L3 cell. (b) Number of simulated particles per cell.

Figure 2.8: Flat plate simulation results with the use of variable time step

It should be noted that, although with the introduction of variable time step, total number of particles required for the simulation would be reduced, the total number of particles are still far from ideal. For flow with very high Mach number, this is even worse. Due to the existence of strong shock wave and boundary layer, density will have large variation in the flow field. As a result, the simulated particles per cell will have large variation in different regions. This cannot be solved solely based on the modification of time step ratio t_r used in each cell, as shown by the remaining ρ and T dependence in Eq. 2.13.

It is worth to mention that, techniques such as deleting particles in cells with too many particles, and adding additional particles in cells with less amount of particles would be another way to get a better particle distribution. However, such cloning and deleting operations should be minimized to avoid statistical random-walk errors. The combination of the above methods may be promising for particle distribution control.

Chapter 3

Cut-cell Method

With the use of a decoupled flow field Cartesian mesh and the boundary surface mesh, which act as the constrain on the Cartesian mesh, the flow field boundary needs to be explicitly determined, based on the information of the boundary surface. Although different from the usual body-fitted grid structure, the decoupled approach used in this thesis has its advantage. From the mesh generation perspective, the time needed to generate good quality mesh is substantially reduced and mesh generation process is made automatic. Other advantages of using a Cartesian mesh have been discussed in Chapter 1.

However, to determine the boundary of the flow field mesh, the relative position between the two sets of mesh: (a) flow field Cartesian mesh, and (b) triangulated boundary surface mesh, should be determined prior to the simulation. Such information is needed to determine the interaction between particles and the boundary, during particle movement.

The current DSMC implementation employs the "Ray-Tracing "algorithm to move particles from cell to cell. In order to detect the collision between a particle and a flow field Cartesian cell face or a boundary element, the "Ray-Trace "algorithm requires only the geometry of the planar element. Thus each triangulated boundary element needs

only to be sorted into the appropriate flow-field Cartesian cell.

When the triangulated boundary surface mesh cuts through a Cartesian cell, the resulting volume of the part of the Cartesian cell that lie within the flow field will be only a portion of the original volume. This volume is required for collision probability calculation, thus should be determined as well.

All these are done by the Cut-cell method. The cut-cell method basically involves two aspects:

- Sorting the triangulated boundary surface mesh elements into particular flow field Cartesian cells.
- Cut-cell volume determination. This is done by the Monte Carlo method.

3.1 Cut-cell Procedures

Let the Cartesian cells be represented by $C_1, C_2, C_3, ..., C_n$, each representing one Cartesian cell. All the Cartesian cells form the whole simulation domain Ω.

$$\Omega = \cup_{i=1}^{n} C_i \tag{3.1}$$

Also let the triangulated surface of the object (objects) inside the simulation domain be represented by triangles $T_1, T_2, T_3, ..., T_m$. All the triangles form the boundary of the object (objects) inside the simulation domain. Let the boundary be S, then,

$$S = \cup_{i=1}^{m} T_i \tag{3.2}$$

The objective is to sort each triangle T_i into the corresponding Cartesian cell C_j. For a triangle T_i, if it intersects with a particular C_j, then T_i should be sorted into that C_j. In other words, T_i will be sorted into one Cartesian cell C_j, if at least one point of T_i is contained in C_j, or mathematically speaking,

$$T_i \cap C_j \neq \emptyset \tag{3.3}$$

It should be noted that, one triangle T_i can be sorted into multiple Cartesian cells. Because one triangle can intersect with multiple Cartesian cells. In this case, each L3 cell contains its own copy of the surface information.

For the situation faced here of determing the intersection between three dimensional Cartesian cell C and two dimensional triangle T, the posible situations between Cartesian cell C and triangle T are listed below:

(1) $T \cap C = \emptyset$. This means T and C don't intersect with each other.

(2) $T \subset C$. This means T is complelely contained in C.

(3) $T \cap C \neq \emptyset$, and $T \nsubseteq C$. This means T intersect with C, but T is not completely contained in C.

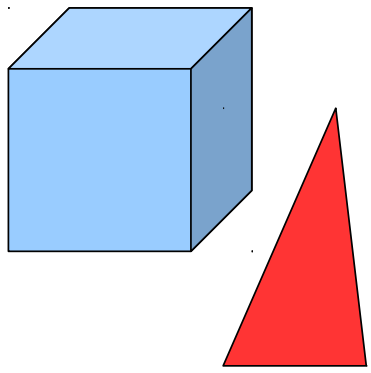


Figure 3.1: Relative position between Cartesian cell and triangle, situation (1)

Situation (1) and (2) are shown skematically in Fig 3.1 and Fig 3.2, respectively. There are two possible cases in situation (3), as we want to determine the intersection between the Cartesian cell and the triangle surface solely based their faces and edges. Case (I) is that at least one edge of the Cartesian cell C intersects with the triangle face, as shown in Fig 3.3. Case (II) is that at least one edge of the triangle T intersects with one face of the Cartesian cell, as shown in Fig 3.4.

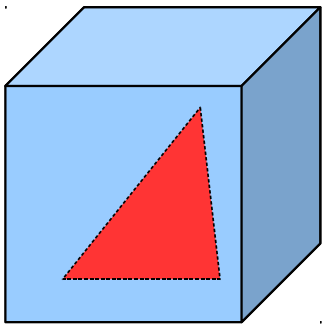


Figure 3.2: Relative position between Cartesian cell and triangle, situation (2)

Situation (2) and (3) are considered as intersection of C and T. For situation (2), it's easy to determine. If all the three vertices of the triangle T are contained in the domain of C, then $T \subset C$, or T intersects with C. This is achieved by the comparison between coordinates of the three vertices of the triangle T and the maximum and minimum coordinates of the Cartesian cell C in each coordinate direction. Situation (2) is checked first. If this situation doesn't happen, situtation (1) and (3) will be checked further.

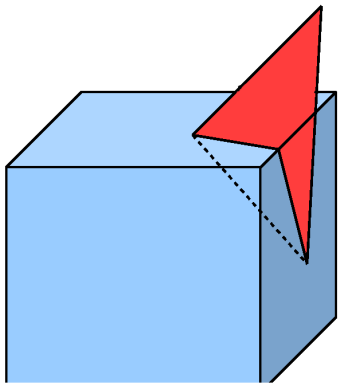


Figure 3.3: Relative position, situation (3) case (I)

For situation (1), no edge of the triangle T intersects with the face of the Cartesian

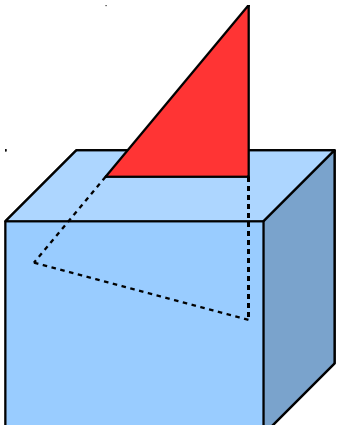


Figure 3.4: Relative position, situation (3) case (II)

cell C, and no edge of the Cartesian cell C intersects with the triangle T.

For situation (3), at least one edge of the Cartesian cell C intersects with the triangle T (as shown in Fig 3.3), or at least one edge of the triangle T intersects with one face of the Cartesian cell C (as shown in Fig 3.4).

If one edge of T intersects with one face of C, or if one edge of C intersects with T, then Cartesian cell C intersects with triangle T in situation (3). Otherwise, C and T don't intersect with each other, this is situation (1). Therefore, situation (2) and situation (3) can be determined based on the same approach. The problem then is equivalent to determine whehether a line segment intersects with a triangle. This is done by the signed volume approach [7] discussed below .

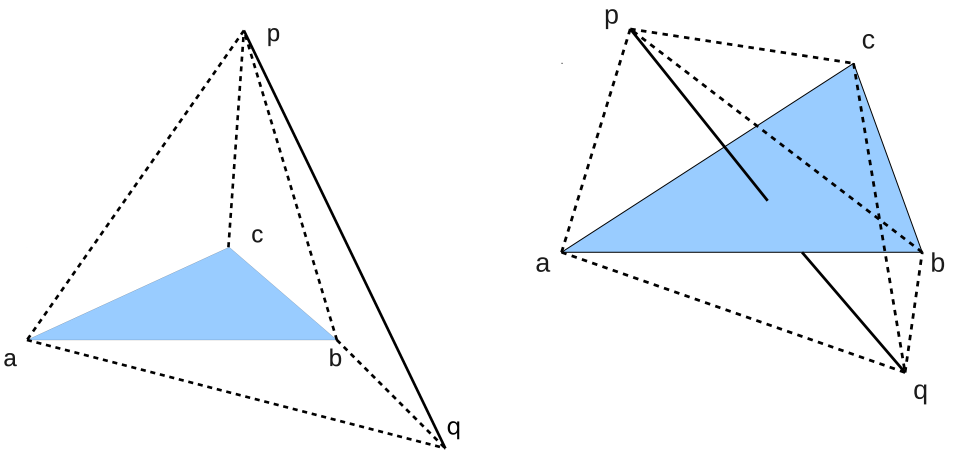
A general tetrahedron formed by four vertices m, n, r, s, has Cartesian coordinates (m_x, m_y, m_z), (n_x, n_y, n_z), (r_x, r_y, r_z) and (s_x, s_y, s_z). The signed volume of the tetrahedron $V(m, n, r, s)$ [7] is given by:

$$V(m,n,r,s) = \frac{1}{6}\begin{vmatrix} m_x & m_y & m_z & 1 \\ n_x & n_y & n_z & 1 \\ r_x & r_y & r_z & 1 \\ s_x & s_y & s_z & 1 \end{vmatrix} \tag{3.4}$$

For a general line segment l_{pq} and a triangle T_{abc}, let the end points of line segment l_{pq} be (p_x, p_y, p_z) and (q_x, q_y, q_z) in Cartesian coordinate system, also let the coordinates of the three vertices of triangle T_{abc} be (a_x, a_y, a_z), (b_x, b_y, b_z) and (c_x, c_y, c_z). The determination of whether the line segment l_{pq} will intersect with the triangle T_{abc}, is achieved in two steps through the following procedures:

(1) If signed volume $V(p, a, b, c)$ and $V(q, a, b, c)$ have a different sign, then l_{pq} intersects with plane formed by T_{abc}, continue to step (2); otherwise, l_{pq} doesn't intersect with triangle T_{abc}, exit.

(2) If signed volume of $V(p, a, b, q)$, $V(p, b, c, q)$ and $V(p, c, a, q)$ have the same sign, then l_{pq} intersects with triangle T_{abc}; otherwise, l_{pq} does not intersect with triangle T_{abc}.



(a) l_{pq} does not intersect with T_{abc} (b) l_{pq} intersects with T_{abc}

Figure 3.5: Using signed vulume to detect intersection between l_{pq} and T_{abc}

As shown in Fig 3.5(b), the conditions above are all satisfied, and line segment l_{pq} intersects with the triangle T_{abc}. While in Fig 3.5(a), although l_{pq} intersects with the plane formed by T_{abc}, the second condition is not satisfied and l_{pq} does not intersect

with the triangle T_{abc}.

Alternatively, the intersection determination between an object and a coordinate aligned region (the situation faced in the current thesis), can be achieved based purely on bitwise operation. This is a more promising method due to its low computational cost. The details of this method can be found in [7].

The current thesis uses a combination of the two methods mentioned above to sort the triangulated boundary mesh into corresponding Cartesian cells. The detailed procedure are as follows:

(1) Read in the surface triangles T_i, store the coordinates of the three vertices of each triangle T_i and the normal vector of each triangle T_i, and number each triangle.

(2) Loop over each triangle T_i and determine all the Cartesian cells which it will intersect with. This is done by:

(a) Loop over all the Cartesian cells C_j $(j = 1, 2, ..., n)$.

(b) For a Cartesian cell C_k, determine whether the triangle T_i is contained in C_j (situation (2)). If yes, then T_i intersects with C_k. Thus, record the number of triangle T_i in C_k, and mark this Cartesian cell C_k as a cut-cell. If not, continue to (d).

(c) Continue to the next Cartesian cell C_{k+1}. Repeat the operations stated in (b).

(d) For a Cartesian cell C_k, determine whether at least one of the three edges of the triangle T_i intersects with a face of C_k.

If yes, then T_i intersects with C_k, record the number of triangle T_i in C_k, mark this Cartesian cell C_k as a cut-cell, and go to (f).

If not, continue to (e).

(e) Determine whether at least one of the twelve edges of the Cartesian cell C_k intersects with the triangle T_i.

If yes, record the number of triangle T_i in C_k, mark this Cartesian cell C_k as a cut-cell, and go to (f).

If not, go to (f).

(f) Continue to the next Cartesian cell C_{k+1}. Repeat the procedures in (d) and (e).

(3) The intersection between T_i and all the Cartesian cells C_j $(j = 1, 2, ..., n)$ are now complete.

(4) Proceed to the next triangle T_{i+1}, go to (2), repeat the whole process in (2).

(5) The intersection between all the triangles T_i and Cartesian cells C_j are now completely determined.

In the implementation of determining the intersection between a triangle and a Cartesian cells, loop over all the Cartesian cells is not a good idea. The number of Cartesian cells usually are very large, and the same is true for the boundary surface triangles. As a result, the time needed to complete the intersection will be in the order of $O(mn)$, where m and n are the number of triangles and number of Cartesian cells respectively. It is therefore very important that for a particular triangle, check only each Cartesian cell which has a potential to intersect with the triangle. In the current implementation, the time needed for the cut-cell algorithm is in the order of 100 simulation time steps, which is not optimal. A typical DSMC simulation usually requires at least several thousand simulation time steps. Further improvements need to be conducted in order to reduce the time required to perform a full cut-cell process.

Based on the data structure used to organize all the flow field Cartesian cells, the number of Cartesian cells which has a potential to intersect with a particular triangle can be small, and the searching of these potential Cartesian cells can be fast. Therefore, the total time needed to determine the intersection between the flow field Cartesian cells

and the boundary surface triangles can be reduced by a large amount. Ideally, a total time of order $O(mlogn)$ can be achieved.

One way of organizing the Cartesian cells for fast searching, would be using the Alternative Digital Tree (ADT) structure as discussed in [16].

3.2 Cut-cell Volume Calculation

The determination of the cut-cell volume is based on the very simple but robust Monte Carlo method. For a Cartesian cell C_i which is cut through by triangles $T_1, T_2, T_3, ..., T_{i_m}$, let the volume of the remaining part that lies inside the flow field region be V_{i_c}, and the volume of the original Cartesian cell be V_i. Randomly generate N_0 number of points in the whole region of the Cartesian cell C_i. Count the total number of points that lie inside the flow field region, and let it be N_i. Then, the volume of V_{i_c} can be estimated by the following expression:

$$V_{i_c} = V_i \frac{N_i}{N_0} \tag{3.5}$$

As the number of randomly generated points increases, the calculated value of volume V_{i_c} converges to its exact value. When a total number of N_0 points are used, the average error between exact value and calculated value are inversely proportional to the square root of the total number of points used N_0, or error $\propto \frac{1}{\sqrt{N_0}}$.

The key issue is to determine whether a point randomly generated in the Cartesian cell is contained in the closed region (regions) formed by the triangulated boundary surface, or is inside the flow field part. The detailed procedure of determing this and therefore the cut-cell volume are as follows:

(1) Loop over each Cartesian cell that is a cut-cell.

(2) Choose one of the vertex (vertices) which is (are) determined to lie inside the flow field part from all the eight vertices of the cut-cell (As long as a Cartesian cell is a cut-cell, there is at least one vertex of it that is inside the flow-field part),

and denote the vertex as Q. Determing which vertex (vertices) will satify such condition is achieved through the same processes as below. However, the whole list of triangles should be tested here.

(3) Randomly generate N_0 number of points in the entire domain of the Cartesian cell. Set the number of points in the flow field part to be zero. In other words, set $N_i = 0$.

(4) Loop over each point P_i, $i = 1,\ 2,\ ...,\ N_0$.

[a] Let $f = 0$ (f is used to record the number of intersections between the line segment QP_i and all the triangles which are contained in the cut-cell). For a point P_i, loop over all the triangles $T_1, T_2, ..., T_{i_k}$ that have been determined in Section 3.1 to intersect with the Cartesian cell. Determine whether the line segment QP_i intersects with a triangle.

[b] If the line segment QP_i intersects with a triangle, let $f = f + 1$.

[c] After all the triangles have been examined with line segment QP_i, check f. If f is odd, then point P_i lies inside the closed boundary surface, and P_i does not lie in the flow field part. If f is even, then point P_i lies in the flow field part.

(5) If point P_i lies inside the flow field, then let $N_i = N_i + 1$. Go to (4) for the next point.

(6) After the positions of all the randomly generated points are determined, calculate the cut-cell volume using Eq. 3.5.

(7) Continue to the next cut-cell, go to (2), repeat (2), (3), (4), (5), (6) to calculate its volume.

(8) The volume of all the cut-cells have been calculated, finish the volume calculation.

The above process is shown in Fig 3.6. Q is chosen to be one of the vertices which lie outside the closed surface. Line QP_1, QP_2 and QP_3 intersect with the surface 1 time,

2 times and 3 times, respectively. So we can determine that point P_1 and P_3 lie outside the flow field and lie in the closed surface region, while point P_2 lies outside the surface region and lies in the flow field.

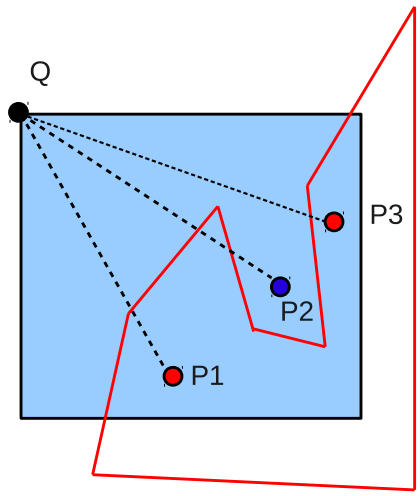


Figure 3.6: Determining whether a point is in the flow field region

The method discussed above is simple and can get high accuracy with the use of a moderate N_0. The problem is that the time spent to calculate the volume of each cut-cell depends on the number of points N_0 used. If N_0 is small, the time spent would be acceptable, while the calculated volume may have large error from exact value. On the other hand, the use of a large N_0 will result in accurate cut-cell volume, while time spent is also large. The number of N_0 that should be used would depend on the accuracy of the cut-cell volume we hope to get. The accuracy of the cut-cell volume will affect the collision probability calculation, which would affect the simulation result. However, it should be noted that such volume errors are more tolerable in a DSMC simulation than in a CFD simulation, where stability is an issue.

Other approaches of calculating the cut-cell volume exist. One of them is discussed in [11]. Another more direct way would be finding all the intersection points between the

Cartesian cell and the triangles. And then using the intersection points, the faces of the Cartesian cell and the triangles that intersect with it to form a closed polyhedron of the remaining flow field part of the Cartesian cell. Calculate the volume of the polyhedron by dividing it to simpler geometries, such as tetrahedron, or use the divergence theorem on the complex polyhedron mentioned in [7].

Chapter 4

Simulation Results

The simulations conducted in this chapter serve the following purposes:

- Verify and test techniques that has been used in this thesis to deal with the various issues encountered in the development of a Cartesian grid based DSMC code capable of AMR, variable time step and the decoupling of the flow field Cartesian mesh and the triangulated boundary surface mesh.
- Provide physical understandings of both the problems that are being simulated and the DSMC method that is being used.

Two simulations will be discussed in detail:

- Hypersonic flow over a blunt body.
- Planetary probe under high Mach number.

4.1 Hypersonic Flow Past a Blunt Body

4.1.1 Simulation Condition and Setup

A hypersonic flow of N_2 with Mach number $M = 15$ over a vertical wall acting as a blunt body, is simulated in this section, the detailed conditions for the simulation are

listed in Table 4.1.

Gas	M_∞	$T_\infty(K)$	$\rho_\infty(kg/m^3)$	$V_\infty(m/s)$	Vertical wall Length L (m)	T_w
N_2	15	80	$6.5 * 10^{-6}$	2735.5	0.16	1500

Table 4.1: Condition for hypersonic flow past a blunt body

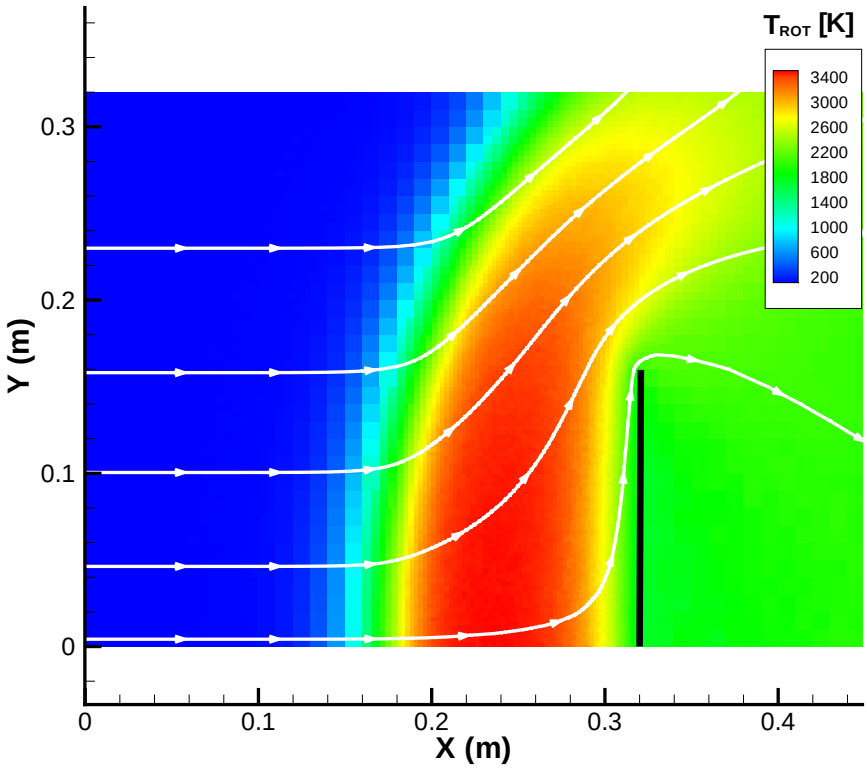


Figure 4.1: Rotational temperature contour of hypersonic flow past a blunt wall

Due to the high Mach number $M = 15$, there will be a strong bow shock formed in front of the vertical wall, which will cause a large density and temperature increase across the shock. Due to the existence of the boundary layer and a relatively low temperature on the vertical wall surface ($T_w = 1500K$) compared to the temperature after the bow shock (around $3800K$), a density increase will also occur near the wall

region. While in the wake region behind the vertical wall, density will be low. The combination of these two effects will cause a large variation of density across the flow field. If a fixed mesh is used, the difficulty on the generation of a mesh with high quality will be severe. However, with the use of AMR, this can be easily solved.

During the simulation, 4 AMRs are used, each including a 1500 simulation time steps of waiting before sampling to allow the flow to reach a steady state and a 3500 time steps of sampling. After the AMR, the cell size is adapted to half the local mean free path, and the simulation time step for each cell is also adapted to around $\frac{1}{5}$ of the local mean collision time.

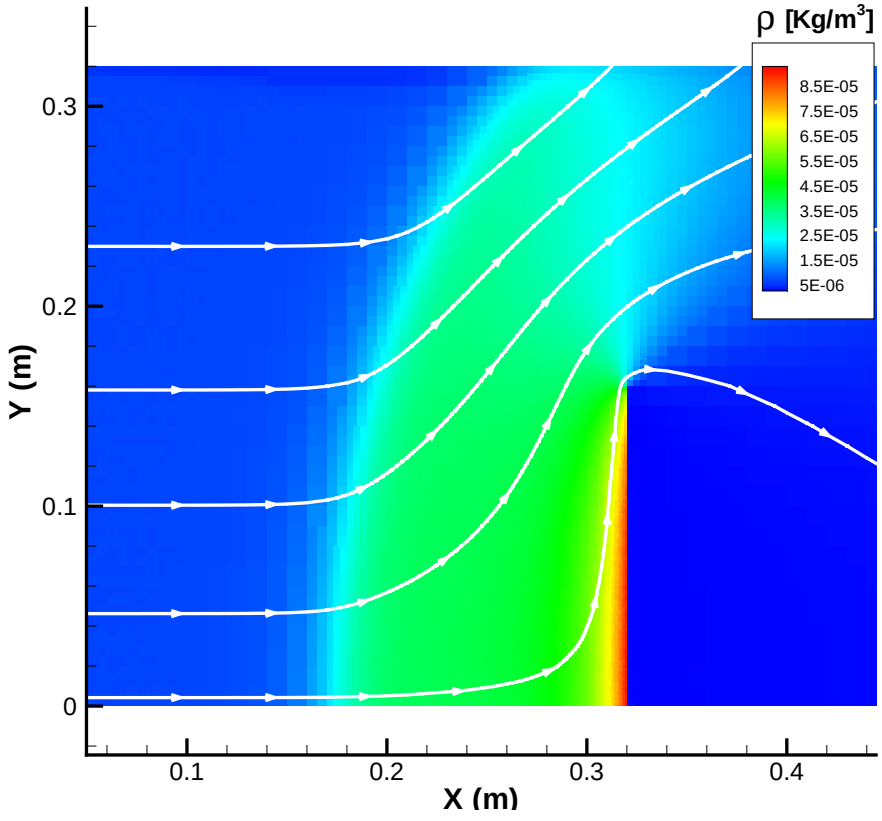


Figure 4.2: Density contour of hypersonic flow past a blunt wall

The idea here is to examine the performance of the AMR method used in this thesis,

and also validate the code for simulation involving high temperature, which is of great importance in hypersonic flows.

4.1.2 Blunt Body Simulation Result

The resulting flow field quantities are given in Fig 2.5, Fig 4.1, and Fig 4.2 respectively. The translational temperature and the adapted mesh are shown in Chapter 2 for comparison with results using a uniform mesh.

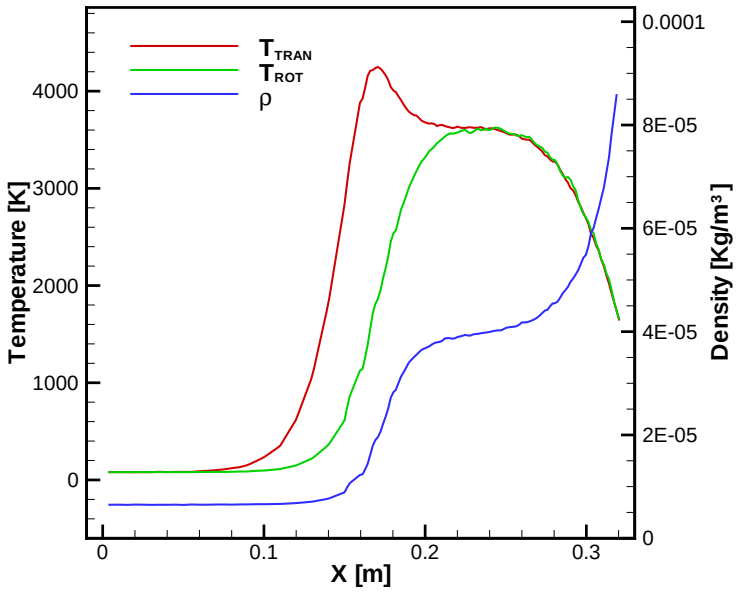


Figure 4.3: Temperature and density distribution along the stagnation stream line

It can be seen from Fig 2.5, due to the large variation in density across the flow field, which is shown in Fig 4.2, the mesh size in different regions of the flow field are far from uniform. The mesh size will first increase by almost a factor of 2 just across the shock due to the increase in temperature. Because the increase of density across the shock, the mesh size will then decrease toward the wall. The cold wall condition will cause an increase in density across the boundary layer, which corresponds to an decrease in mesh

size. Near the wake region, the flow has relatively low density, which will cause a larger mesh size.

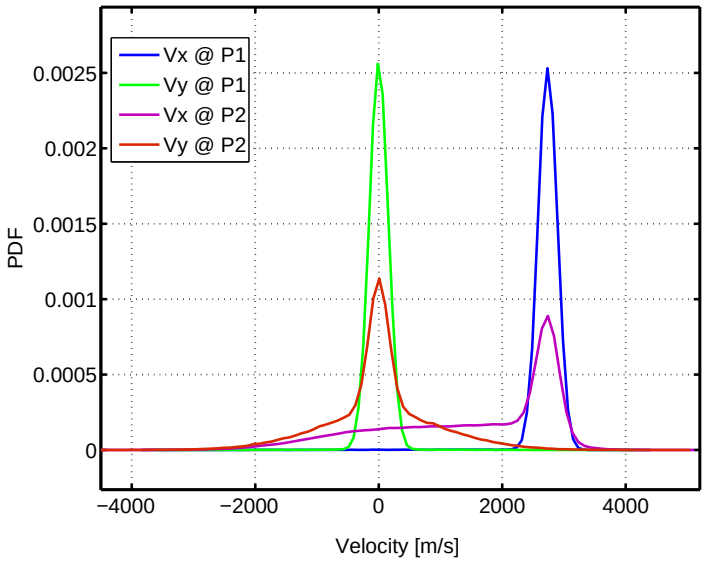


Figure 4.4: Velocity distribution function at two locations

The distribution of translational temperature T_{TRAN}, rotational temperature T_{ROT} and density ρ along the stagnation stream line are also plotted in Fig 4.3. As can be seen from Fig 4.4, the velocity distribution functions before and in the shock, due to the finite distance, the particles cannot reach an equilibrium distribution. As a result, the rotational temperature will lag the translational temperature. Also, the translational temperature will have an overshoot across the shock due to the existence of a bimodal velocity distribution. Because the existence of the boundary layer and a cold wall, the density will continue to increase from post shock condition toward the wall.

The two position in Fig 4.4 are located at $x = 0.08m$ and $x = 0.16m$.

4.2 Planetary Probe Simulation

4.2.1 Simulation Condition and Setup

A hypersonic flow of N_2 with Mach number $M = 20.2$ over a planetary probe is simulated in this section, the detailed conditions for the flow are listed in Table 4.2. The shape of the probe is sketched in Fig 2.6. The condition is the same as SR3 experiment [13]. The base diameter of the probe is $0.05m$. The difference from the experiment geometry is that, here the probe doesn't have a sting, while for the SR3 experiment, the probe is fixed on a sting. However, due to the high Mach number of the flow and a relatively low angle of attack, the sting will have little effect on the flow field except for the wake regions. Moreover, the impact on surface properties along the fore-body will be negligible. This makes the comparison between the results from SR3 experiment and the DSMC simulation results here reasonable.

The intersection between the surface of the planetary probe and the background Cartesian mesh should be determined, thus allow the surface of the probe acting as the wall boundary. However, due to a relatively small Cartesian mesh size in the near probe wall region, the intersection will be extremely complex in this situation, as can be seen from Fig 4.8. The time required to determine the intersection between the probe surface triangular mesh and the background Cartesian mesh will also become an important factor in the simulation. Thus, the automatic and efficient determination of intersection between flow field Cartesian mesh and surface triangular mesh will be very important. At a result, this problem can act as a good case, to test the capability of the cut-cell method of intersection determination developed in this thesis. The time required to perform a full cut-cell algorithm for this problem is in the order of several hundred simulation time steps, which is acceptable for this simulation. However, time efficiency of the cut-cell method developed here need to be improved for problems with larger scales.

Gas	M_∞	$T_\infty(K)$	$\rho_\infty(kg/m^3)$	$V_\infty(m/s)$	T_w
N_2	20.2	13.3	$1.73 * 10^{-5}$	1502	300

Table 4.2: Condition for planetary probe simulation

4.2.2 Planetary Probe Simulation Results

Two angle of attack $\alpha = 0^\circ$ and $\alpha = 10^\circ$ with the above condition are simulated. The translational temperature T_{TRAN} on the symmetry plane of the Planetary probe, and the heat transfter coefficient C_H on the probe surface are plotted in Fig 4.6 for angle of attack $\alpha = 0^\circ$ case. The total temperature of the free stream flow is $T_0 = 1100K$. However, across the shock, the translational temperature has a maximum value around $1200K$. This is due to the fact that the molecules are in a non-equilibrium state across the shock, same as the situation faced in Section 4.1. The mesh after AMR is also plotted for the symmetry plane. The heat transfer coefficient C_H is defined as:

$$C_H = \frac{q}{\frac{1}{2}\rho v^3} \tag{4.1}$$

For the case of angle of attack $\alpha = 10^\circ$, the translational temperature along the symmetry plane of the probe and the surface heat transfer coefficient C_H on the probe surface are plotted in Fig 4.5. In this case, the stagnation point is not in the center of the sphere surface, but away from it. This can be seen from the heat flux coefficient C_H distribution along the probe in the same figure.

The temperature and density along the stagnation stream line is plotted in Fig 4.7 for the case with angle of attack $\alpha = 0^\circ$. Due to the relative small shock stand-off distance from the stagnation point on the surface, there is no clear interface between the shock and the boundary layer, which is different from Section 4.1. This can be clearly seen from the density profile. In Fig 4.7, the density ρ continuously increase along the stagnation stream line, while in Fig 4.3, the density first increases across the strong normal shock, then keeps as a constant along the stagnation stream line untill

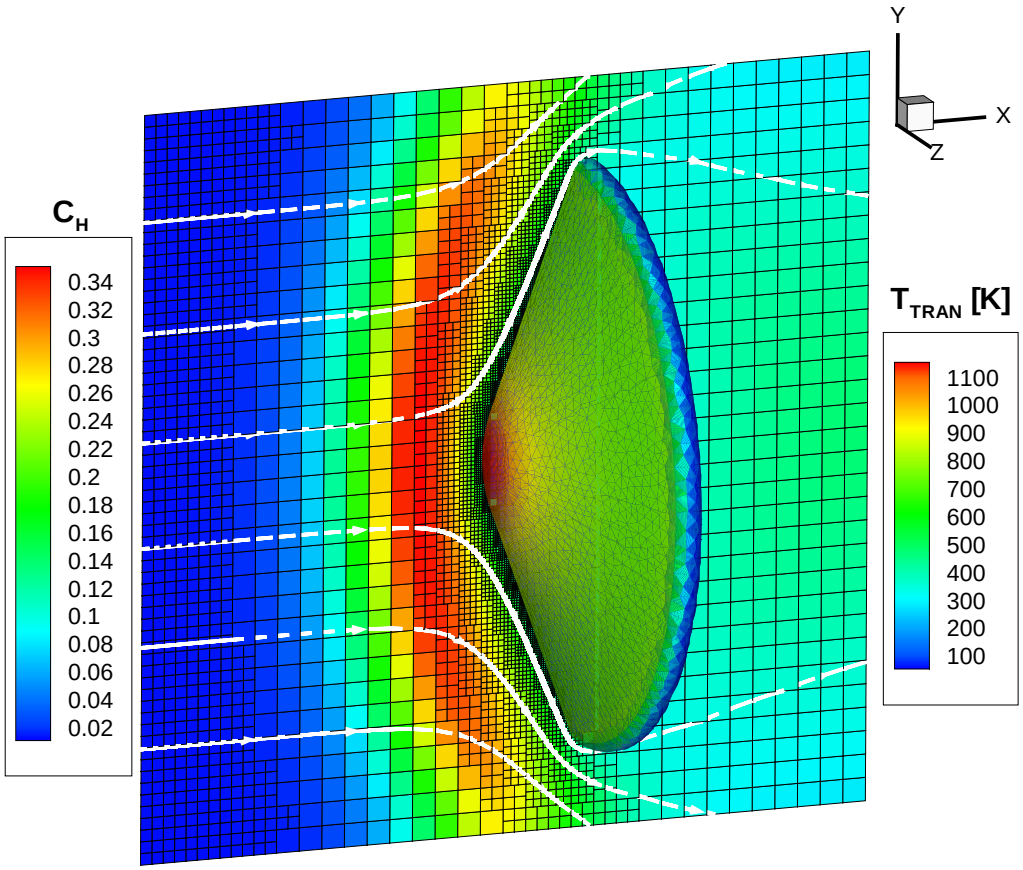


Figure 4.5: T_{TRAN} on the center plane, and C_H on the probe surface with $\alpha = 0°$

reaching the boundary layer edge, and then the density increases again. However, for both cases, the increase of density along the stagnation stream line toward the wall is due to the existence of the boundary layer. While the pressure in the boundary layer is near constant, the temperature decreases toward the wall due to the low wall temperature, so the density will increase toward the wall. The gas is in a non-equilibrium state across the shock, though the highest temperature available in this case is relatively low. This can be seen by the lag of rotational temperature from translational temperature.

The mesh on the symmetry plane of the probe and on one cross section perpendicular

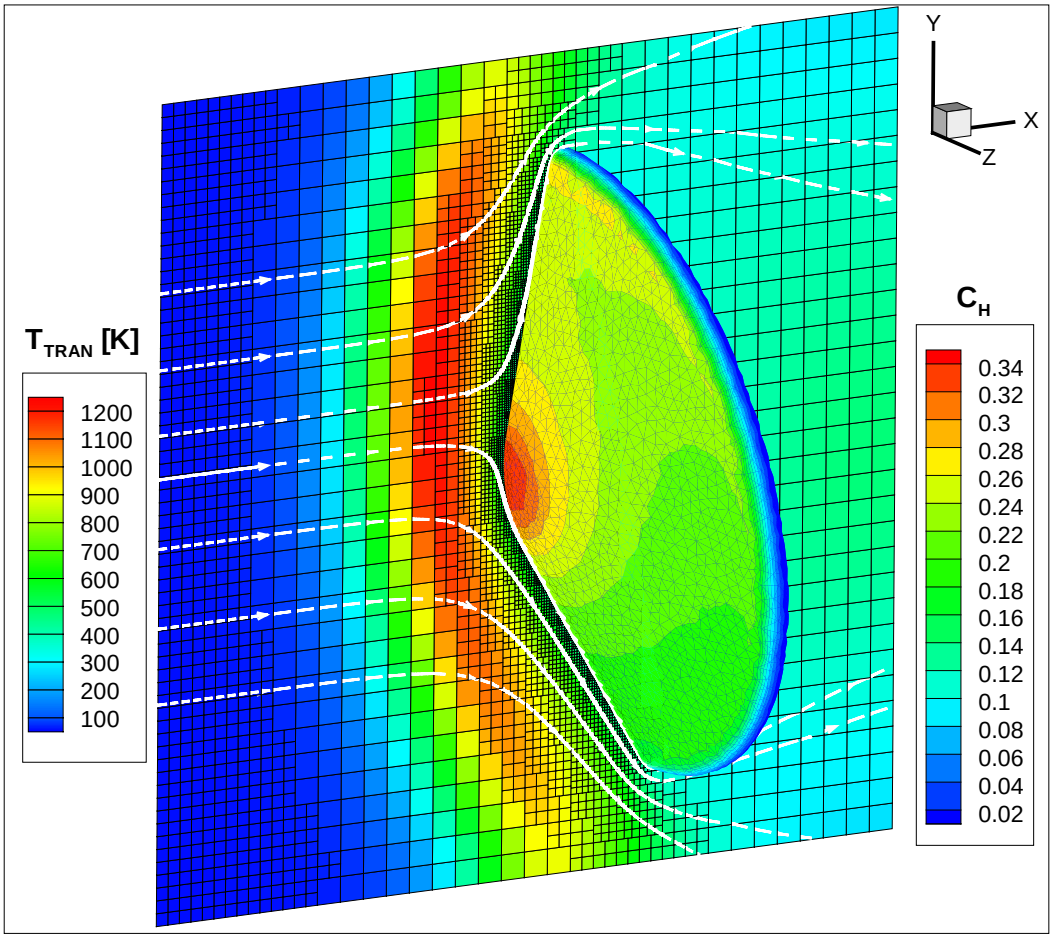


Figure 4.6: T_{TRAN} on the center plane, and C_H on the probe surface with $\alpha = 10°$

to x direction are plotted in Fig 4.8 for angle of attack $\alpha = 10°$ case. The size of the surface triangular mesh are larger by several factor than the flow field Cartesian mesh. This is allowed in the current cut-cell method framework. However, it should be noted that for body-fitted mesh, which uses surface mesh as a boundary constrain to generate flow field mesh, the resulting flow field mesh near the boundary region has the same size as the surface mesh. Due to the large increase of density toward the wall (because of the boundary layer), the mesh size decreases from far field region toward the wall region substantially, as can be seen from the mesh on the plane perpendicular to x direction

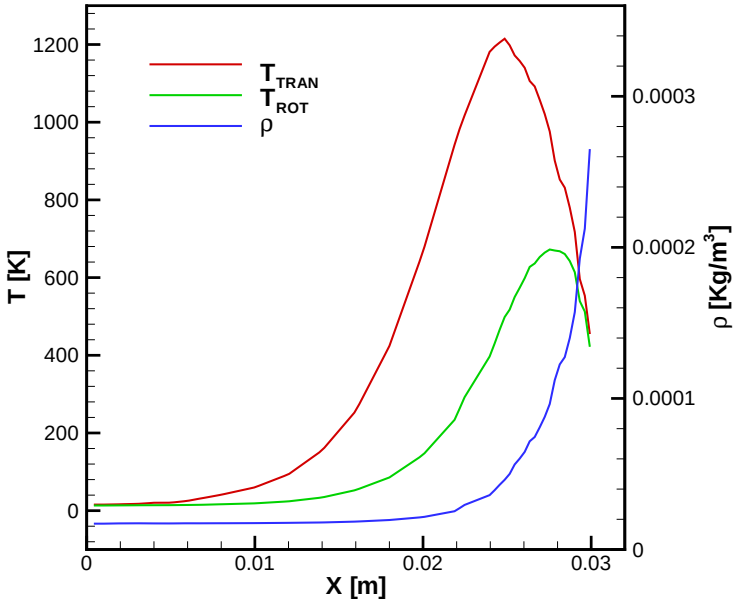


Figure 4.7: Temperature and density distribution along the stagnation stream line

in Fig 4.8.

The heat flux q for angle of attack $\alpha = 0°$ and $\alpha = 10°$ cases are plotted in Fig 4.9. The SR3 experimental results are also plotted in the same figure for comparision. The results are along the windward side starting from the center of the nose of the planetary probe. It can be seen that, the DSMC results for the two cases are in good agreement with experimental results.

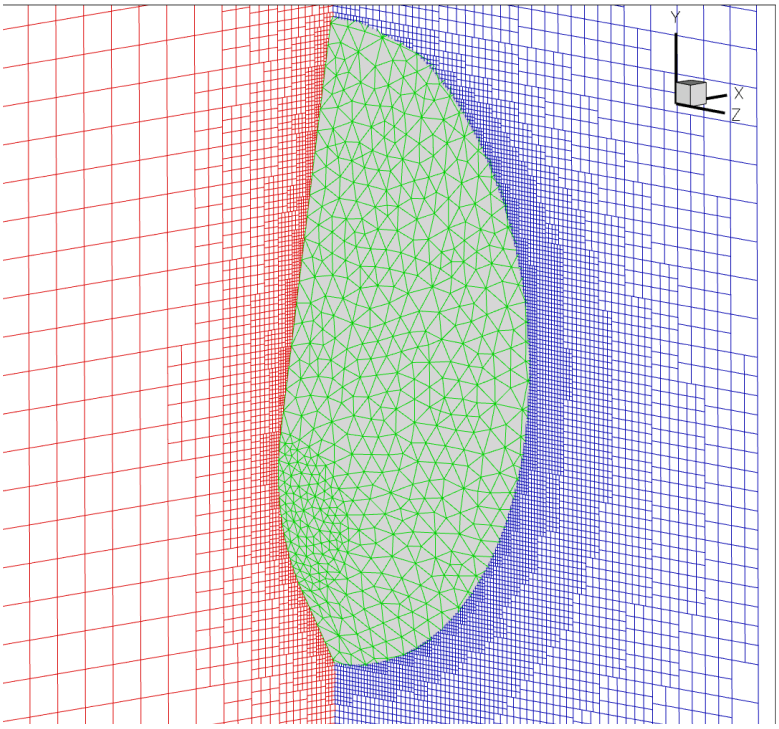


Figure 4.8: Flow field mesh on two cross sections at the angle of attack $\alpha = 10°$

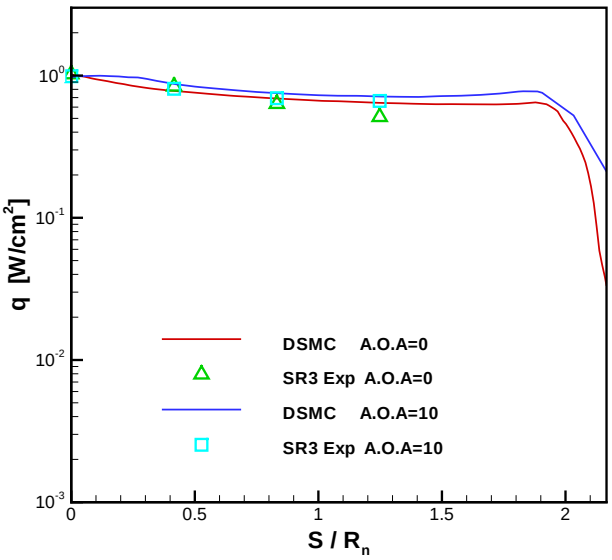


Figure 4.9: Surface heat transfer q at the angle of attack $\alpha = 0°$ and $\alpha = 10°$

Chapter 5

Conclusions

In this thesis, the implementation details of the adaptive mesh refinement and cut-cell algorithms for a three-level Cartesian grid based DSMC method is presented. Results are given to both verify the algorithms developed in this thesis, and demonstrate the capability of the current Cartesian grid based DSMC method.

5.1 Conclusions

(1) An automatic AMR algorithm was developed and implemented for a three-level Cartesian grid in this thesis.

(2) A General cut-cell algorithm was developed and implemented for a three-level Cartesian mesh.

(3) Simple Cartesian-based AMR provides precise cell refinement efficiently, even for 3D flows. The time required for the AMR is in the order of 10 simulation time steps regardless of the problem scale.

(4) General cut-cell method is able to imbed complex 3D geometry into the flow field Cartesian mesh. The Monte Carlo volume calculation algorithm is very simple and robust. The time required for the full cut-cell algorithm is in the order of

several hundred simulation time steps depending on the number of the surface triangles and the number of flow field Cartesian cells.

(5) 3D simulations of Planetary probe reproduce experimental heat flux measurements.

5.2 Future Works

(1) Problems with larger scale need to be conducted to test the performance of the currently developed AMR and cut-cell algorithms. The robustness and time efficiency of the current AMR and cut-cell algorithms need to be further tested and improved.

(2) Physical models should be added to the general DSMC code structure to deal with internal energy exchange between different energy modes and chemical reactions associated to such process.

References

[1] G. A. Bird. Monte carlo simulation of gas flows. *Annu. Rev. Fluid Mech.*, Vol. 10:11–31, 1978.

[2] G. A. Bird. *Molecular Gas Dynamics and the Direct Simulation of Gas Flows*. Oxford University Press, Oxford, UK, 1994.

[3] E. S. Oran, C. K. Oh, and B. Z. Cybyk. Direct simulation monte carlo: Recent advances and applications. *Annu. Rev. Fluid Mech.*, Vol. 30:403–441, 1998.

[4] W. G. Vincenti and Jr C. H. Kruger. *Introduction to Physical Gas Dynamics*. Wiley, New York, 1967.

[5] D. Gao, C. Zhang, and T. E. Schwartzentruber. A three-level cartesian geometry based implementation of the dsmc method. *AIAA Paper 2010-450*, Jan. 2010.

[6] G. J. LeBeau. A parallel implementation of the direct simulation monte carlo method. *Computer Methods in Applied Mechanics and Engineering*, Vol. 174:319–377, 1999.

[7] M. J. Aftosmis, M. J. Berger, and J. E. Melton. Robust and efficient cartesian mesh generation for component-based geometry. *AIAA Journal*, Vol. 36, No. 6:952–960, 1998.

[8] G. A. Bird. Definition of mean free path for real gases. *Physics of Fluids*, Vol. 26:3222–3223, 1983.

[9] M. J. Berger. Data structures for adaptive grid generation. *SIAM J. Sci. Stat. Comput*, Vol. 7, No. 3:904–916, 1986.

[10] J. Bell, M. Berger, J. Saltzman, and M. Welcome. Three-dimensional adaptive mesh refinement for hypersonic conservation laws. *SIAM J. Sci. Comput*, Vol. 15, No. 1:127–138, 1994.

[11] A. L. Garcia, J. B. Bell, W. Y. Crutchfield, and B. J. Alder. Adaptive mesh and algorithm refinement using direct simulation monte carlo. *Journal of Computational Physics*, 154:134–155, 1999.

[12] S. Dietrich and I. D. Boyd. Scalar and parallel optimized implementation of the direct simulation monte carlo method. *Journal of Computational Physics*, Vol. 126:328–342, 1996.

[13] J. N. Moss and J. M. Price. Survey of blunt body flows including wakes at hypersonic low-density conditions. *Journal of Thermophysics and Heat Transfer*, Vol. 11, No. 3:321–329, 1997.

[14] T. E. Schwartzentruber, L. C. Scalabrin, and I. D. Boyd. Multiscale particle-continuum simulations of hypersonic flow over a planetery probe. *Jounal of Spacecraft adn Rockets.*, Vol. 45, No. 6, 2008.

[15] K. C. Kannenberg and I. D. Boyd. Strategies for efficient particle resolution in the direct simulation monte carlo method. *Journal of Computational Physics*, Vol. 157:727–745, 2000.

[16] J. Bonet and J. Peraire. An alternating digital tree (ADT) algorithm for 3d geometric searching and intersection problems. *International Journal for Numerical Methods in Engineering*, Vol. 31:1–17, 1991.