國立交通大學

機械工程學系

博 士 論 文

應用非結構性網格之通用平行化三維 **DSMC** 程式**(PDSC)**的研究與發展

Development of a General-Purpose Parallel Three-Dimensional DSMC Code (PDSC) Using Unstructured Tetrahedral Mesh

研 究 生：曾坤樟

指導教授：吳宗信　博士

中 華 民 國 九 十 四 年 七 月

應用非結構性網格之通用平行化三維 DSMC 程式(PDSC)的研究與發展

Development of a General-Purpose Parallel Three-Dimensional DSMC Code (PDSC) Using Unstructured Tetrahedral Mesh

研 究 生：曾坤樟　　Student：Kun-Chang Tseng

指導教授：吳宗信　　Advisor：Dr. Jong-Shinn Wu

國 立 交 通 大 學
機 械 工 程 學 系
博 士 論 文

A Thesis
Submitted to Department of Mechanical Engineering
College of Engineering
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy
in

Mechanical Engineering

July 2005

Hsinchu, Taiwan

中華民國九十四年七月

Development of a General-Purpose Parallel Three-Dimensional DSMC Code (PDSC) Using Unstructured Tetrahedral Mesh

Student: Kun-Chang Tseng Advisor: Dr. Jong-Shinn Wu

Department of Mechanical Engineering
National Chiao-Tung University

Abstract

A general-purpose parallel three-dimensional direct simulation Monte Carlo code (PDSC) using unstructured tetrahedral mesh is developed and validated in this thesis. Important features of this PDSC include parallel processing with dynamic domain decomposition, combination of variable time-step scheme with solution-based adaptive mesh refinement, conservative weighting scheme for treating trace species and chemical reaction functions for hypersonic air flows. A multi-level graph-partitioning technique is employed to adaptively decompose the computational domain according to the workload distribution among processors during runtime, which can alleviate the unbalancing loading before it becomes a problem. A three-dimensional *h-refined* unstructured adaptive mesh with simple mesh quality control, based on a preliminary DSMC simulation, is used to obtain suitable mesh resolution to increase the accuracy of the DSMC solution. A variable time-step method using the concept of fluxes (mass, momentum and energy) conservation across the cell interface is implemented to reduce the number of simulated particles and the number of iterations of transient period to reach steady state, without sacrificing the solution accuracy. A conservative weighting scheme, ensuring exact momentum and near exact energy conservation during each particle collision, is incorporated into the PDSC to efficiently treat flows with trace species. This method is validated and shows it can greatly reduce both the number of particles and computational time. Chemical reaction module, including dissociation, recombination and exchange reactions, is incorporated into the PDSC for treating reactive flows. It is verified by comparing probability, degree of dissociation and mole fractions of a single 2-D cell with theoretical data. Completed PDSC is then applied to compute several complicated, challenging flow problems to demonstrate its superior computational capability. The results are also validated with experimental data or previous simulation data wherever available.

Organization of this thesis is briefly described as follows. Chapter 1 describes the background, motivation and objectives of the current study. Chapter 2 describes the general DSMC method and overview of the current implementation of the PDSC. Chapter 3 introduces the variable time-step scheme in combination with solution-based adaptive mesh refinement on unstructured tetrahedral mesh. Chapter 4 describes the parallel implementation and performance of the DSMC method using dynamic domain decomposition. Chapter 5 describes the conservative weighting scheme and its superiority in treating flows having trace species. Chapter 6 describes the chemical reaction functions for treating hypersonic air flows along with its validation using single-cell simulation. Chapter 7 describes the results of simulating several challenging flow problems using the PDSC. Chapter 8 concludes and summarizes the important findings of the current study, along with recommended future studies.

Keywords: direct simulation Monte Carlo, unstructured tetrahedral mesh, parallel processing, dynamic domain decomposition, variable-time-step, adaptive mesh refinement, conservative weighting scheme, chemical reaction, graph-partitioning

應用非結構性網格之通用平行化三維 DSMC 程式(PDSC)的研究與發展

學生：曾坤樟　　　　　　　　　　指導教授：吳宗信　博士

機械工程學系博士班

國立交通大學

中文摘要

本論文發展並驗證一通用平行化之三維直接模擬蒙地卡羅程式(PDSC)於非結構性四面體網格。這個 PDSC 程式有幾個重要的特色：以動態區域切割之平行化處理、結合變時步方法於可調適網格、可有效率處理稀少氣體之權值守恆法(conservative weighting scheme)以及處理高速流之化學反應等等。此程式利用多層式圖形切割技術，在模擬的過程中根據每顆處理器的負載做動態區域切割。此外，利用 DSMC 的初步結果，三維非結構可調適網格(*h-refined*)法以及簡易的網格品質控制可以用來增加 DSMC 的準確性。變時步法則利用通過界面的流量(質量、動量以及能量)守恆之觀念，在避免犧牲結果準確性的前提下，同時減少模擬分子數目以及到達穩態的疊代次數。確保動量及能量在每次碰撞後均能守恆的權值守恆法(CWS)可以用來有效的模擬具有稀少氣體的流場。這個方法被證實可以大量減少模擬分子的數目及計算時間。分離(Dissociation)、結合(Recombination)以及交換(Exchange)的化學反應亦被結合在 PDSC 中來處理有化學作用的流場。最後利用發展完全的 PDSC 模擬數個複雜、具有挑戰性的流場來顯示它優越的計算能力。計算結果與實驗結果或前人的模擬結果進行比較，驗證程式的正確性。

本論文的結構簡述如下：第一章描述此研究之背景、動機以及目的。第二章為 DSMC 法之簡介及目前 PDSC 軟體結構內容之概要描述。第三章介紹結合變時步法之非結構性可調適四面體網格。第四章為利用動態區域切割之平行化 DSMC 程式。第五章介紹權值守恆法以及其處理稀少氣體流場之優點。第六章為化學反應於高速流以及利用單一網格之驗證。第七章為利用 PDSC 模擬幾個具有挑戰性

的應用及結果。第八章為此研究的重要發現，以及對未來發展的建議。

關鍵字：直接模擬蒙地卡羅法，非結構四面體網格，平行化處理，動態區域切割，變時步法，可調適網格，權值守恆法，化學反應，圖形切割

Acknowledgements

I would like to thank my research advisor, Professor Jong-Shinn Wu, for his support and guidance for this work. He gave me continuous encouragement and assistance to my research and life. It lets me has interests and keep studying during this research. I also learn how to figure problems out by my own and have a chance of one year study in the United States. It is real a turning point of my life.

My graduate colleagues and research associates are also unforgettable. Our continuous discussions are significantly efficient and helpful to improve the performances of this research. You also make my research span became more colorful with your companies.

I am grateful to Professor I. D. Boyd and those colleagues in his group at the University of Michigan in the United States for giving me chance to study chemical reaction function of the DSMC method. It is a wonderful year to study with them.

I also want to thank the National Science Council (NSC) and the National Center for High-Performance Computing (NCHC) in Taiwan. These simulations with immense computing cost cannot be finished without your supports.

Finally, a thankyou gives to all my friends and family for all their supports. This thesis would not complete without you.

Table of Contents

List of Tables

List of Figures

Nomenclature

λ	： mean free path
ρ	： density
δ	： mean molecular spacing
α	： is a penalty parameter
	： symmetric factor
	： degree of dissociation
	： attack angle
σ	： the differential cross section
ω	： viscosity temperature exponent
τ	： surface skin-friction force
ϕ	： m, mv, $mv^2/2$ or other internal energy
	： free-stream parameter
	： weighting ratio between cells
θ	： characteristic temperature of dissociation
	： circumferential angle
ϕ_0	： preset value of free-stream parameter for adaptive mesh refinement
ρ_d	： characteristic density for dissociation
ε_{rot}	： rotational energy
ε_v	： vibrational energy
ζ_{rot}	： rotational degree of freedom
ζ_v	： vibrational degree of freedom
$\triangle t$	： time-step
$\triangle x$	： cell size
σ_T	： the total cross section
A_c	： cell area for 2-D
B	： balance factor
c	： the total velocity

c'	：random velocity
c_o	：mean velocity
c_r	：relative speed
C	：repartitioning cost
C_f	：surface skin-friction coefficient
C_h	：surface heat transfer coefficient
C_p	：surface pressure coefficient
d	：molecular diameter
D	：the throat diameter of the twin-jet interaction
$d\Omega$	：element of solid angle
d_{ref}	：reference diameter
E	：edge cut of graph
	：energy
E_a	：activation energy
E_c	：cutting edge
f	：particle phase-space distribution function
F	：external force per unit mass
F_N	：particle weight
G	：graph
I	：degree of imbalance
k	：the Boltzmann constant
k_f	：forward rate constant
k_r	：reverse rate constant
Ke	：equilibrium constant
Kn	：Knudsen number
Kn_c	：local cell Knudsen number
Kn_{cc}	：preset value of cell Knudsen number for adaptive mesh refinement
L	：characteristic length;
	：degree of load imbalance

m	：molecule mass
M_∞	：free-stream mach number
n	：number of processor;
	：speedup of parallel computing;
	：number density
N	：fluctuating number of simulated particles
$\tilde{N}$	：average number of simulated particles
N_p	：the number of simulated molecules of p^{th} cell
N_x	：the number of sub-domains along x-direction
N_y	：the number of sub-domains along y-direction
N_z	：the number of sub-domains along z-direction
p	：surface pressure
P	：momentum
	：reaction probability
q	：surface heat flux
r	：a distance
Re	：Reynolds number
S	：weight of a sub-domain
t	：time
T^*	：potential well-depth temperature
T_o	：stagnation temperature
T_{ref}	：reference temperature
T_{rot}	：rotational temperature
T_{tot}	：total temperature
T_{tr}	：translational temperature
T_v	：vibrational temperature
T_w	：wall temperature
u	：molecular velocity

V	：vertex of graph
	：cell volume for 3-D
W	：degradation function
	：particle number of each sub-domain
W_p	：the particle weight
Zr_∞	：limiting rotational collision number

Chapter 1
Introduction

1.1 Background and Motivation of the Thesis

Gas flows are often described correctly using the Navier-Stokes equations. However, in some flow regimes, the Navier-Stokes equations fail to approximate the gas dynamics behavior and the particle nature of the matter must be taken into account. One of these is the rarefied gas flow ($Kn \geq 0.01$, see Bird [11]), which the mean free path becomes comparable with, or even larger than, the characteristic length of flows. These high Knudsen number flows are now of practical scientific and engineering importance. For example, the examples include the pumping characteristics of turbo-molecular drag vacuum pump [1-3], the low-pressure plasma etching and chemical vapor deposition (LPCVD) [4], the micro-filter [5], and the micro-electro-mechanical-system (MEMS), [6-8], *etc*. Due to their importance in practical applications, accurate and efficient numerical modeling of these phenomena, rather than solving the Navier-Stokes equations, becomes necessary for understanding the underlying physics.

It is well known that the Boltzmann equation is more appropriate for all flow regimes; it is, however, rarely used to numerically solve the practical problems because of two major difficulties. They include higher dimensionality (up to seven) of the Boltzmann equation and the difficulties of correctly modeling the integral collision term. An alternative method, known as Direct Simulation Monte Carlo (DSMC) method, was proposed by Bird to solve the Boltzmann equation using direct simulation of particle collision kinetics, and the associated monograph was published in 1976 [10] and 1994 [11].

Later on, both Nanbu [12] and Wagner [13] were able to demonstrate mathematically that the DSMC method is equivalent to solving the Boltzmann equation as the simulated particle numbers become large. This method has become a widely used computational tool for the simulation of gas flows in which molecular effects become important. The advantage of using particle method under these circumstances is that molecular model can be implemented directly to the calculation of particle collisions. It has been applied very successfully to compute rarefied hypersonic flows [8, 14], and other fundamental scientific problems, such as flow instabilities [15, 16]. In addition to the space science applications, it has also been utilized in the analysis of ultra-high vacuum technology [1-3]. Very recently, it was applied to rarefied internal gas flow problems such as channel, pipe, ducted slider air bearing flows and the results generally agree very well with experiments [8, 14, 17].

Figure 1.1 [11] illustrates the effective limits of major approximations in the DSMC method. The dilute gas assumption requires that δ/d>>1 and δ/d=7 has been chosen as the limit, as shown as a vertical solid line. δ and d are mean molecular spacing and molecular diameter, respectively. The longer tilted dash line represents Kn=0.1 which is the demarcation between the continuum approach and the particulate approach. On the right-hand side of this line, continuum approach, such as the Navier-Stokes equations, is valid; while it is necessary to consider the particle nature of the flow on the left-hand side of this line. The shorter tilted dash line has been chosen as L/δ=100 as the criterion for the onset of significant statistical fluctuations.

With the increasing importance of the rarefied gas dynamics, an efficient simulation tool becomes necessary to understand and engineer rarefied gas dynamics. Thus, in the current study we intend to develop a general-purpose DSMC code, which can be used efficiently for either fundamental or practical understanding of general rarefied gas dynamics in the modern technological development.

1.2 Reviews of the DSMC Method

1.2.1 General Features

The direct simulation Monte Carlo (DSMC) method [10, 11] is a particle method for solving the Boltzmann equation describing gas flows. The gas is modeled at the microscopic level using simulated particles, which each represents a large number of physical molecules or atoms. The physics of the gas are modeled through the motion of particles and collisions between them. Mass, momentum and energy transports between particles are considered at the particle level. The method is statistical in nature and depends heavily upon pseudo-random number sequences for simulation. Physical events such as collisions are handled probabilistically using largely phenomenological models, which are designed to reproduce real fluid behavior when examined at the macroscopic level. General procedures of the DSMC method consist of four major steps: moving, indexing, collision and sampling. In the current study, we either use Variable Hard Sphere (VHS) or Variable Soft Sphere (VSS) molecular models [10, 11] to reproduce real fluid behavior as well as No Time Counter (NTC) method [11] for the collision mechanics. Details of the procedures and the consequences of the computational approximations regarding DSMC can be found in Bird [10, 11].

In the following section, we will introduce several well-known DSMC codes, which include *Visual DSMC Program*, *DAC*, *MONACO*, *SMILE* and *PDSC*. The details of the standard DSMC method along with the current implementation will be introduced in Chapter 2.

1.2.2 Existing DSMC Software

There are relatively few general-purpose DSMC codes available in the public domain. Table 1.1 summarizes the list of the main features of the four DSMC codes, including Visual *DSMC Program*, *DAC*, *MONACO* and *SMILE*. They are briefly introduced in the following in turn. Interested readers can refer to the listed references for details.

1.2.2.1 Visual DSMC Program

Professor G. Bird, who invented the DSMC method back in 1958, at the University of Sydney in Australia, designed the *Visual DSMC Program*. It is a stand-alone Window version that can be used for two-dimensional, axis-symmetric and three-dimensional flows. It is equipped with the real-time animation of moving particles and displays of macroscopic property contours. It is an excellent tool for understanding the basics of the DSMC method and the physics of rarefied gas dynamics. However, it is not practical to utilize it for practical engineering problem due to its speed of simulation. The related information can be found in http://www.gab.com.au/.

1.2.2.2 DAC

For simulating rarefied gas dynamics environment, the **D**SMC **A**nalysis **C**ode (*DAC*) [18] is a well-known DSMC software designed by Gerald J. LeBeau at the National Aeronautics and Space Administration (NASA). It was awarded by NASA as the *2002 Software of the Year* Award and has been used extensively to support numerous space related programs, projects and missions. For example, it has been used to simulate plume impingement of space shuttle, Russian Mir Space Station, aerodynamics of re-entry space shuttle and Hubble Space Telescope (HST) servicing mission. It is a 2-D/Axi-symmetirc/3-D simulator having features such as parallel computing using dynamic load balancing, adaptive meshing and chemical reaction functions. Both distributed-memory and shared-memory can be used to run the *DAC*. Structured background grid coupled with unstructured triangular surface grid is used to treat complicated geometry of objects.

1.2.2.3 MONACO

MONACO is a well-developed DSMC code by Professor I. D. Boyd at the University of Michigan in the United States [19]. It can be run on both workstation- and PC-class clusters. It has been applied to compute plume flows, hypersonic flows and materials processing. It is currently under development of coupling CFD and DSMC methods. The *MONACO* code also can be used for 2-D/Axi-symmetirc/3-D simulation. It can be run by serial or parallel computation with unstructured mesh. Important features include chemical reactions for hypersonic air flows, variable time-step scheme and “manual” dynamic domain decomposition. Some preprocessors were developed to ease the preprocessing tasks.

1.2.2.4 SMILE

SMILE, stands for **S**tatistical **M**odeling **I**n the **L**ow-density **E**nvironment, is a software system based on the DSMC method, which is developed by Professor M. Ivanov at the Institute of Theoretical and Applied Mechanics, Novosibirsk, Russia [20]. It is a parallel version with dynamic domain decomposition and several numerical techniques to save the computational time. By now *SMILE* has been used to compute high-altitude aerodynamics problems, nozzle and plume interactions, to name a few. The detailed features are listed in Table 1.1.

In the following, we will introduce and review some important aspects of implementing DSMC in practice, which more or less explain our preference to some specific choices than others.

1.2.3 Structured and Unstructured Mesh in DSMC

Most applications of the DSMC have used *structured grids* [10, 11] to discretize the physical domain. For problems with complicated geometry, multi-block meshing techniques were developed first by Bird [11], which involved two steps: dividing the flow field into several blocks followed by discretizing each block into quadrilateral (2-D) or hexahedral (3-D) cells. Subsequent related research has been directed to develop alternative meshing techniques such as the coordinate transformation method by Merkle [21], the body-fitted coordinate system by Shimada and Abe [22] and the transfinite interpolation method by Olynick *et al.* [23]. However, all of these still used structured grids. It is much easier to program the code using structured grids; however, it often requires tremendous problem specific modification. To alleviate such restriction, an *unstructured grid* system is an alternative choice. Many physical problems involve very complicated geometry of objects; thus, unstructured mesh has been recommended to take advantage of the flexibility of handling this situation, although it might be computationally more expensive. In addition, using unstructured mesh has the flexibility of applying graph-partitioning technique for parallel implementation of the DSMC method [24-26]. Boyd's group [27-29] has applied such technique to compute thruster plumes produced by spacecraft and found that the results are very satisfactory. Wilmoth *et al.* [30] have used two types of grid (unstructured tetrahedral and structured Cartesian grids) to compute the low-density, hypersonic flows about reusable launch vehicle. Both methods were shown to give comparable results. Wu's group [8, 31] has also developed 2-D and 3-D codes to compute nozzle plume and vacuum pump, respectively. It was concluded that unstructured grid has certain advantages in grid refinement as compared with structured grid.

1.2.4 Adaptive Mesh Refinement

The success of the DSMC method relies on the proper distribution of simulated particles and cells. The collision partners of each cell are selected from each cell without considering their relative positions. This assumption is reasonable as long as the cell dimension is less than a mean free path. Ideally, the cell size has to be small enough, e.g., one third of the mean free path [10, 11]. Thus, solution-based mesh adaptation becomes critical step in obtaining accurate flow solution using DSMC, especially for flows having highly non-uniform density variations. The development of mesh adaptation in CFD and DSMC are described in the following for the purpose of comparison.

CFD: For the past decade, the development of CFD using unstructured adaptive meshes has greatly extended the capability of predicting complex flow fields. Several adaptive mesh techniques have been developed to increase the resolution of “important” region and decrease the resolution of “unimportant” region within the flow field, as reviewed by Powell *et al.* [32].

In general, mesh adaptation can be categorized into three methods [33]: (1) re-meshing (mesh generation), (2) mesh movement, and (3) mesh enrichment (or *h-refinement*). For the first method, a solution based on the initial mesh is obtained, and then the mesh is regenerated, which the mesh points are more concentrated on where resolution of the solution is needed. This new mesh may contain more or fewer mesh points than the original mesh. For the second method, the total mesh points remain the same in the computational domain. It is common to use a spring analogy, in which the nodes of the mesh are connected by springs whose stiffness is proportional to certain measure of solution activity over the spring. The mesh points are moved closer into the region where solution gradients are relatively large. This is often applied to the spatial adaptation of a structured mesh. For the third method, mesh enrichment, mesh points are added or embedded into the regions where relatively large solution gradients are detected, while the global mesh topology remains intact. It is generally regarded that mesh enrichment method has certain advantages over the first two methods [33, 34]. One of the most important advantages is that the mesh enrichment technique is in general many times faster and robust than the re-meshing technique [34]. In Ref. [33], it is mentioned that the disadvantage, however, is that the implementation of mesh enrichment involves a significant modification to existing numerical schemes due to the appearance of hanging nodes [33]. This can be easily overcome, however, by some simple methods through the elimination of hanging nodes, as proposed by Kallinderis and Vijayan [35].

DSMC: The corresponding development and the application of the adaptive mesh technique in particle method, such as the DSMC method, has been largely ignored. Applying adaptive mesh technique in the DSMC method, as in CFD, not only improves the flow field resolution without increasing the computational cost much, but also more or less equalizes the statistical uncertainties in the averaging process of obtaining the macroscopic quantities.

Among the very few studies about this subject, Wang and Harvey [36] have first applied a solution-based, re-meshing adaptive grid technique (mesh regeneration using the advancing front method) in unstructured mesh to study the hypersonic flow field with highly non-uniform density variations involving shocks. Later on, in the same group, Robinson [25] has applied a similar technique combining a parallel DSMC method to compute a hypersonic flow over compression ramp at different Knudsen numbers. However, some unexpected results such as lower accuracy for a refined mesh, as compared with a coarse mesh, arose due to smaller particle-per-cell caused by too many cells.

Cybyk *et al.* [37] have developed a technique using the Monotonic Lagrangian Grid (*MLG*) in the DSMC method, which provides a time-varying grid system that automatically adapts to local number densities within the flow field. However, the application of this *MLG* technique to external gas flows is not promising due to the particle sorting problems inhered in the scheme. Additionally, this technique highly restricts the time-step size as compared with the traditional DSMC method, which makes the cost of obtaining the steady-state solution comparably high.

Garcia and Bell [38] have developed an adaptive mesh and algorithm refinement (*AMAR*) embedding the DSMC method within a continuum method (N-S equations solver) at the finest level of an adaptive mesh refinement (*AMR*) hierarchy. This method can cope with problems involving physics in several orders of magnitude of length scale.

Recently, Wu's group [39, 40] also proposed 2-D and 3-D DSMC methods with adaptive mesh using *h-refinement* technique. Density was used as the adaptation parameter considering the statistical nature of the DSMC method. However, highly skewed cells often appear since no quality control of the mesh has been implemented. Thus, some policy during mesh adaptation should be added to improve the mesh quality.

<u>Concerns Related to Mesh Adaptation</u>

Before implementing the adaptive mesh techniques, several concerns need to be considered as pointed out by the excellent review article by Powell *et al.* [32] and the references cited therein. These concerns are mainly applied to CFD; however, most of these are true to the DSMC method as well. These include the data structure, the initial mesh generation, the mesh adaptation procedure, the adaptation parameters and criteria, and the effect of mesh adaptation on computational algorithm. These are briefly described in turn in the following from the perspectives of the DSMC method.

Data structure

There exists strong relationship between the selected mesh adaptation and the data structure to be used. The majority of the DSMC codes apply structured mesh as mentioned previously. Structured mesh allows the particle tracking relatively easy and accesses the mesh information in memory more directly; however, it lacks the flexibility on mesh adaptation. There are two ways of adapting the structured mesh. One way, called *r-refinement*, is to "distort" the mesh distribution, so that mesh redistributes more crowded in the region where it needs mesh refinement. Another way, called *h-refinement*, is to add mesh in the localized region in both x- and y-directions; however, it increases unexpectedly the mesh population in other unexpected regions as well, where they do not require at all. To allow for the addition (or deletion) of mesh in the computational domain avoiding the problem outlined in the above, more sophisticated mesh data structure, rather than the simple structured mesh, has to be adopted. Unstructured mesh cannot be mapped onto a computational space with structured (*i,j*) indexing. Instead, the connectivity information of mesh has to be stored, which makes the mesh data access indirectly. However, the spawning of the mesh in the regions of interest is much easier as compared to structured mesh due to the un-ordered data structure. It was thus concluded that unstructured mesh is superior to structured mesh considering the advantages and implementation of mesh adaptation.

Initial Mesh Generation

The initial un-adapted mesh required for computation is generated via either advancing front method or Delaunay triangulation if unstructured mesh is used. The detailed descriptions of these two methods can be found in Lohnern and Parikh [41] and Baker [42], respectively, and are not repeated here.

Mesh Adaptation Procedure

We have to decide how to adapt the mesh. For the unstructured mesh, there are generally two ways of adapting the mesh: re-meshing and embedding [36]. For the re-meshing procedure, generation of connectivity information for all the mesh is required at each adaptation step. This is expensive in general as mentioned previously. For the embedding technique, local *h-refinement* is used to introduce new mesh points and only the mesh in the immediate vicinity of new mesh need to be connected. Hence, it requires less computational effort. In addition, we also have to decide if isotropic or an-isotropic refinement is used. Generally, the quality of the mesh using isotropic refinement is superior to that using an-isotropic refinement. Thus, some mesh quality control policies may have to be implemented to ensure the good quality of cells.

Adaptation parameters and criteria

The decision of where to refine or coarsen the mesh is one of the very critical issues in mesh adaptation scheme. For the DSMC method, it often requires that the computational mesh size is much smaller (at least 1/3~1/2) than the local mean free path [10, 11], which is inversely proportional to the local number density. However, density is naturally a parameter to be considered. In addition, the choice of density as the adaptation parameter helps equalize the statistical uncertainties throughout the cells due to the sampling process for obtaining the macroscopic quantities.

Effect on Computational Algorithm

The effects of mesh adaptation on DSMC are trivial since the cell is used mainly for selecting collision partners and sampling the particles. Thus, the computational procedure is exactly the same except the number of the cells increases after mesh adaptation.

1.2.5 Variable Time-Step Scheme

The accuracy of a DSMC simulation is directly related to the number of simulated particles per cell throughout the cells. As the number of simulated particles increases, the statistical uncertainties of the macroscopic properties reduce due to better collision condition. The number of simulated particle per cell is shown to inversely proportional linear and square of gas density for two- and three-dimensional flows, respectively. That is, the simulated molecules are fewer in higher density regions, while lower density regions are over resolved. More computational time is spent calculating the lower density regions than is needed. A strategy to increase the computational speed without jeopardizing the accuracy of the solution is to reduce the number of simulated particles by using cell/particle weighting, but maintaining near-uniform particle distribution per cell, e.g., cell weighting for axisymmetric flow [10, 11], particle weighting and variable time-step [43, 44]. Kannenberg and Boyd [43] presented strategies for efficient particle resolution in DSMC. The authors manipulated variations of particle weight, variations of time-step and grid arrangement to obtain a more uniform particle count throughout the flow field. It was shown that careless use of cell/particle weighting often introduces some detrimental effects to the statistical accuracy, which is caused by repeatedly cloning the particles in the flow field [43]. Nevertheless, variable time-step method represents one of the simplest and most efficient ways of particle weighting that avoids the problem of particle cloning, if careful grid manipulation is done [43]. To obtain a more uniform distribution of model particles per cell throughout the computational domain, a variable time-step scheme is highly recommended.

Markelov and Ivanov [44] proposed a method with zoned variable time-steps in the DSMC method to simulate an axisymmetric shock wave/laminar boundary layer interaction by the DSMC method and to analyze the influence of flare length on the separation region extent. The authors employed the *SMILE* code to obtain adaptive mesh. The simulation domain was divided into sub-domains with different time-steps. This technique allows controlling the number of particle in each sub-domain. A relatively uniform particle distribution and 30% decrease in the total number of molecules was reported, correspondingly, to a higher computational efficiency.

Moss *et al.* [45] have developed a local time-step scheme in the DSMC method and apply to compute a 5-deg wedge flow. A grid generation and adaptation procedure is also incorporated to insure the cell size requirement of the DSMC method. In this method, only one time-step is used throughout the flow field for unsteady flow. If the flow is steady, then the computational effort can be facilitated by subdividing computational domain into an arbitrary number of regions with different local time-steps. Thus, the number of particles and overall computational time for steady state are reduced using the local time-step for each sub-domain.

Usami [46] also has developed a different time-step scheme to simulate supersonic jet expansion at a very large pressure ratio. The flow filed has been divided into 12 blocks and classified into seven classes, where the further block from the orifice has a longer time-step. If the particle passes through the interface of each domain, the time-step of the further block from the orifice becomes twice as the closer block.

1.2.6 The Parallel DSMC Method

The DSMC method has become a widely used computational tool for the simulation of gas flows in which molecular effects become important. The advantage of using a particle method under these circumstances is that molecular model can be applied directly to the calculation of particle collisions, while the continuum methods use macroscopic averages to account for such effects. Therefore, particle methods can predict these effects with much higher accuracy. Also, it is the only viable tool for analyzing the gas flows in the transitional regime. Nevertheless, the main drawback of such direct physical method is its high computational cost. That is why the DSMC method was only used for analyzing high Knudsen number flows (or transitional flows). For lower Knudsen number gas flows near the continuum regime, the computational cost is prohibitively high, even with the most advanced computer nowadays. Hence, it is important to increase the computational speed to extend the application range of the DSMC method.

1.2.6.1 Domain Decomposition

Generally, these are two methods to partition the simulated domain, which are geometry-based and graph-based domain decomposition. Geometry-based method is usually faster but provided poor edge cut (E_c) since they pay no heed to the connections of the point in the mesh. Many physical problems can be expressed within the framework of *graph theory*, such as discrete optimization problems and matrix reordering. Sketch of graph and mesh is shown in Fig. 1.2. A small portion of a triangular grid, which is usually made by commercial mesh software, is shown as the thinner lines. The bold solid circles and bold lines represent the vertices and edge cuts of the graph, respectively. The graph $G(V, E)$ is the collection of these vertices (V) and edge cuts (E) on the basis of the connectivity between the cells. One of the advantages in expressing the problem in terms of a graph is that each of the edges and vertices can be assigned a *weight* to account for the specific numerical application. For example, in DSMC, the vertex (cell center) can be weighted with the particle numbers with all edges having unitary weight. The brief overviews of each partition method are described in the following.

Geometry-based method

Geometry-based methods use spatial (or coordinate) information of mesh to partition the domain. These methods are usually simple and fast but provide poor E_c (cutting edge) and poor load balancing.

Coordinate Partitioning

This method is the simplest way of geometric partition that is only working for rectangular and cubic domain. In this method, the domain is split into $N_x \times N_y$ and $N_x \times N_y \times N_z$ sub-domain, which is based on their geometric position for 2-D and 3-D mesh, respectively. The chief advantages are that the partition domain is easy to obtained, and the cheaper computational cost. Examples of this method are given in Haug *et al.* [47].

Recursive Coordinate Bisection (RCB)

This method is similar to the coordinate partitioning that attempts to minimize the boundaries between the sub-domains. The procedure is to find out the longest coordinate direction of the domain and then split the mesh in half. And then each sub-domain is recursively divided by the same method. This method is also fast but the quality is low and sometimes obtains the disconnected sub-domains. Furthermore, it is only useful in splitting the domain by the normal direction of the coordinate axes. Examples of this method are given in Simon [48].

Inertial Recursive Bisection (IRB)

This method is a modification from the *RCB* method due to the geometry of the domain is not always orthogonal to the coordinated axes. The concept of this method is to find out the longest inertial axis. The domain is split in half to produce bisection and then applied recursively. A smaller sub-domain boundary is created than the *RCB* method. *IRB* has been implemented within the context of dynamic load balancing for DSMC by Diniz et al [49].

Graph-based method

A conventional graph-partitioning problem is to subdivide the *n* vertices between the *NP* sub-domains while minimizing the number of edge cuts, E_c, and balancing the weight in each sub-domain. For example, in DSMC, each vertex can be weighted with the number of particles with all edges that connect cell centers, having unitary weight. A truly dynamic load balancing technique is required for DSMC because the load (approximately proportional to the number of particles) in each sub-domain changes frequently, especially during the transient period of simulation. Domain decomposition in DSMC may become very efficient by taking the advantage of successful development in graph partitioning. However, it is well known that it is *NP* complete, which means that the optimal solution of this problem is impossible to compute in polynomial bounded time. Instead, it is relaxed to seek near-optimal solutions within reasonable time. In computer science, there are several methods developed for achieving near-optimal solutions to this problem. Related description and reviews of graph partitioning can be found in the references [48, 50-53].

Greedy Partitioning

This method is first proposed by Farhat [50] to decompose the domain of the finite element computation. This method is started from determining a vertex in the domain. This vertex "bites" the neighboring vertices until appropriate proportion of the graph is formed. And the next vertex of the other sub-domain is seeded on the border of the previous sub-domain and the processes are repeated until the whole domain is partitioned. The variant employed here differs from Farhat [50] only in that it works with graph based rather than the node and elements of a finite element mesh.

Recursive Spectral Bisection (RSB)

Simon [48] first presents this graph partition by computing the Laplacian matrix of the graph. Then, the eigenvector (*Fiedler* vector) corresponding to second largest eigenvalue, when associated with the vertices of the graph, gives a measure of the distance between the vertices. Once the measures of distance are computed for all vertices, they can be sorted by their value and then split into two parts. This method seens to produce good quality partitions, but it is relatively computationally expensive on calculating the *Fiedler* vector.

Multilevel Scheme

Barnard and Simon [51] first proposed multilevel partitioning to accelerate the convergence in CFD simulations. The process of this partitioning is first coarsing to obtain a small graph; it is easier and finds a good partition of a small graph than a large graph. The second stage is computing a high-quality bisection, that is, small edge-cut, of the coarse graph. And finally the refinement process is operated here to get the partition domain and repeat these three processes until the partition reaches comparable quality. Multilevel schemes now are considered state of the art static partitioners. *METIS* [52], developed at the University of Minnesota, is a variant of the multilevel scheme of graph partitioning. The idea of multilevel scheme draws from the multi-grid technique. Reported performance was impressive in terms of CPU time.

Two-Step Method

Two-step methods may be seen as relatives of the multi-level partition methods. The concept of two-step partition method is to refine the initial domain by a cheap and high-speed partition method. Moving the vertices on partition boundaries optimizes the domain decomposition. A cost function F is proposed to determine the minimum value, which is, $F = \alpha E_c + (1-\alpha)L$, where E_c the edge cut associated with the decomposition, L is the degree of load imbalance and α is a penalty parameter. *JOSTLE* [54] uses an initial domain decomposition (generated by greedy partitioning) and successively adjusts the partition by moving vertices lying on partition boundaries. In this method, vertex shedding is localized since only the vertices along the partition boundaries are allowed to move, not the vertices anywhere in the domain. Hence, this method possesses a high degree of concurrency and has the potential of being applied in the dynamic domain decomposition in the event of load imbalancing across the processor array.

<u>Others</u>

In addition, Plimpton and Bartel [55] have proposed a random partitioning method for the DSMC method. The computational cells are randomly picked up to form separate sub-domains. This is the simplest and a fast method of domain decomposition, resulting in maximum edge cuts and disconnected domains, although it may give moderate load balance.

Thus far, there seems no report that matured graph-partitioning tool has been incorporated in the parallel DSMC method on an unstructured mesh. Thus, it is interesting and technically important to learn that if the graph partition tools can be used efficiently in conjunction with the DSMC method. Thus, we use *PJOSTLE* [56] to dynamically decompose the computational domain for the parallel DSMC simulation in the early stage of the study and, recently, we use *ParMETIS* [57] instead for its robustness and much bigger user community around the world, which is often the driving force to improve the tool.

1.2.6.2 Static Domain Decomposition (SDD)

In the past, several studies on parallel implementation of DSMC have been published [58-62] using static domain decomposition and structured mesh. Message passing was used to transfer molecules between processors and to provide the synchronization necessary for the correct physical simulation. Results showed reasonable speedup and efficiency could be obtained if the problem is sized properly to the number of processors.

Recently, Boyd's group [29, 63] designed the parallel DSMC software named *MONACO*, which emphasized high data locality to match the hardware structure of modern workstations, while maintains the code efficiency on vectorized supercomputers. In this code, unstructured grids were used to take the advantage of flexibility of handling complex object geometry. Static domain decomposition technique was used to distribute cells among processors. Interactive human interruption is required to redistribute the cells among processors to maintain workload balance among processors, which is indeed unsatisfactory from practical viewpoint. Timing results show the performance improvement on workstations and the necessity of load balancing for achieving high performance on parallel computers. Maximum 400 IBM-SP2 processors have been used to simulate flow around a planetary probe with approximately 100 million particles, which parallel efficiency of 90% has been reached by manually redistributing the cells among processors during simulation. However, the parallel efficiency for *n* processors is unusually defined as the ratio of computational time to the sum of computational and communicational time, rather than which is normally defined as the ratio of the true speedup to the ideal speedup (*n*) for *n* processors.

1.2.6.3 Dynamic Domain Decomposition (DDD)

Until very recently, dynamic domain decomposition technique was used in conjunction with the parallel implementation of DSMC method. Ivanov's group [64] has developed a parallel DSMC code called *SMILE*, which implements both the static and dynamic load balancing techniques. *SMILE* has united the background-structured cells into groups, so-called "cluster", which is the minimum spatial unit, and are distributed and transferred between the processors. The dynamic domain decomposition algorithm is scalable and requires only local knowledge of the load distribution in a system. In addition, the direction and the amount of workload transfer are determined by the concept of heat diffusion process [65]. In addition, an automatic granularity control is used to determine when to communicate the data among processors [65].

Around the same period of time, dynamic load balancing technique, using Stop At Rise (SAR) [66], which compares the cost of re-mapping the decomposition with the cost of not re-mapping, based on a degradation function, was used in conjunction with the parallel implementation of the DSMC method [60, 62]. In the study [60], they used a runtime library, *CHAOS*, for data communication and data structure manipulation on a structured mesh. Results show that it yields faster execution times than the scalar code, although only 25% of parallel efficiency is achieved for 64 processors. LeBeau [67] reported that parallel efficiency up to 90% is achieved for 128 processors for the flow over a sphere. It is not clear how they implemented the dynamic load balancing, although they did mention they have used the concept of heat diffusion [65]. In LeBeau's study [67], surface geometry is discretized using an unstructured triangular grid representation. A two-level embedded Cartesian grid is employed for the discretization of the computational domain.

In summary, studies about DSMC using both purely unstructured mesh and dynamic domain decomposition were relatively few in the past [24-26], although using unstructured mesh exhibits higher flexibility in handling objects with complicated geometry and boundary conditions. Robinson [24-26] has been the first to develop a heuristic, diffusive, hybrid graph-geometric, localized, concurrent scheme, *ADDER*, for repartitioning the domain on an unstructured mesh. Dramatic increase of parallel efficiency was reported as compared with that of static domain decomposition. However, Robinson [24-26] has shown that the parallel efficiency begins to fall dramatically as the number of processors increases to some extent due to the large runtime of the repartitioning the domain relative to the DSMC computation. Thus, the utilization of a more efficient repartitioning runtime library is essential to improve the performance of a parallel DSMC method.

Based on previous reviews, the development of the parallel DSMC method has not taken the advantage of the great success in graph partition. Considering the nature of DSMC, a truly dynamic domain decomposition technique is required because the load (approximately proportional to the particle numbers) in each sub-domain changes frequently, especially during the transient period of simulation. However, such implementation is definitely not an easy task to accomplish.

1.2.7 Conservative Weighting Scheme

For most applications, each species of the flows has the same order of particle density. It is unfortunately that sometimes the flow involves trace particle situation. One typical example is the hypersonic flow in aerospace engineering, which includes launching and re-entry processing. From the MSIS-E-90 Atmosphere Model (http://nssdc.gsfc.nasa.gov/space/model/models/msis.html], the components of the atmosphere at 80 km altitude are nitrogen molecules (N_2), oxygen molecules (O_2) and oxygen atoms (O). The ratio of mole fraction oxygen atom to nitrogen molecule is less than 10^{-5}. This means that to simulate one oxygen atom in a computational cell would require 100,000 nitrogen molecules, if the same particle weighting is used for both species. Another example is the chemical vapor decomposition (CVD) in materials processing. Chemical species of most importance in CVD process often occurs in very small amount of quantities. The amount of the trace gas, for example, SiH_2, existing in CVD is usually very small for low-pressure environment, on the order of 10^{-4} or even smaller for the mole fraction. The above two examples present a major difficulty to the DSMC algorithm. To simulate one particle of the trace gas in a computational cell at this mole fraction would require simulation of 10,000 other particles, if constant weighting scheme is applied for all species. Flows with trace species will lead to computational waste for those non-trace particles. Even with a numerically efficient DSMC code implemented on parallel computers, this simulation would require hundreds of hours or even more of execution time.

To circumvent this difficulty, a conservative weighting scheme (CWS) was developed by Boyd [68] to deal with the trace species often involved in some non-reactive physical processes, which is otherwise considered computationally impossible using the conventional DSMC method. This conservative weighting scheme improves greatly the statistical uncertainties by decreasing the weighting factors of trace-species particles, while it ensures the conservation of both momentum and energy between two colliding particles with large difference of weighting factors.

1.2.8 Chemical Reactions

So far, we have reviewed or discussed features of the DSMC method in simulating non-reactive flows. As the flow speed is very high and with appreciable temperature, consideration of chemical reactions becomes necessary to faithfully simulate the gas flows. Thus, how to efficiently model the chemical reactions under the framework of DSMC is important for simulating hypersonic flows.

The total collision energy model (TCE) is proposed by Professor G. A. Bird [10, 11], which indicates the reaction probability is a function of total collision energy and coefficients of Arrhnius equation. Dogra *et al.* used this chemical model to simulate a

1.6-meter-diameter sphere in hypersonic rarefied flow [69]. Professor Boyd also developed a chemistry model, which is so called vibrationally-favored dissociation (VFD). A three-body recombination reaction is also present in this paper [70]. A new modified reaction model is also proposed to figure out systematic errors when used with discrete energy modes by Ivanov's group [71]. A hypersonic flow over spacecraft in the Martian and Earth atmosphere is simulated to study the flow field with chemistry.

1.3 Objectives of the Thesis

Based on previous reviews, the ultimate goal of the current study is to develop a general-purpose three-dimensional DSMC code (PDSC) with the following features:

1. Using unstructured tetrahedral mesh for better treatment of complex geometry of boundaries.
2. Parallel processing with dynamic domain decomposition for a fast and load-balancing simulation.
3. Using variable time-step scheme, in combination with solution-based adaptive mesh, to reduce the simulated particles and to increase the accuracy of the numerical solution.
4. Using conservative weighting scheme to efficiently treat gas flows with trace species.
5. Simulating hypersonic air flows considering chemical reactions including dissociation, exchange and recombination reactions.

The thesis begins with descriptions of the conventional DSMC method and overview of the current implementation of the PDSC in Chapter 2. Mesh adaptation and variable time-step scheme for an unstructured mesh will be discussed in Chapter 3. Parallel computing of DSMC are presented in Chapter 4. Chapters 5 and 6 are description and verification of conservative weighting scheme and molecular chemical reactions of PDSC, respectively. Chapter 7 presents some applications of the current DSMC implementation to some realistic flow fields. Finally, conclusions of the current study and the recommended future studies are summarized in Chapter 8.

Chapter 2
An Overview of the Current Implementation of the DSMC Method

2.1 The Boltzmann Equation

As mentioned previously, the Boltzmann equation is valid for all flow regimes. It describes the statistical distribution of particles in a fluid. It is one of the most important equations of non-equilibrium statistical mechanics, the area of statistical mechanics that deals with systems far from thermodynamics equilibrium. The Boltzmann equation is used to study how a fluid transports physical quantities such as heat and current, and thus to derive transport-related properties such as viscosity, thermal conductivity and electrical conductivity. There are some assumptions made in the derivation of the Boltzmann equation which define limits of applicability.

1. Molecular chaos is an essential component which is valid when the intermolecular forces are short range. It allows the representation of the two particles distribution function as a product of the two single particle distribution functions.
2. Distribution functions do not change before particle collision. This implies that the encounter is of short time duration in comparison to the mean free collision time.
3. Assume all collisions are binary collisions.
4. Particles are uninfluenced by intermolecular potentials external to an interaction.

According to these assumptions, the Boltzmann equation is derived and shown as Eq. (2.1);

$$\underbrace{\frac{\partial(nf)}{\partial t}}_{1}+\underbrace{u_i\frac{\partial(nf)}{\partial x_i}}_{2}+\underbrace{F_i\frac{\partial(nf)}{\partial u_i}}_{3}=\underbrace{\left.\frac{\partial f}{\partial x_i}\right|_c}_{4}=\underbrace{\int_{-\infty}^{\infty}\int_{0}^{4\pi}n^2(f^{'}f_1^{'}-ff_1)g\sigma d\Omega dU}_{4} \qquad (2.1)$$

The meaning of particle phase-space distribution function f is the number of particles with center of mass located within a small volume d^3r near the point $\boldsymbol{r}$, and velocity within a range d^3u, at time t. F_i is an external force per unit mass, t is the time and u_i is the molecular velocity. σ is the differential cross section and $d\Omega$ is an element of solid angle. The prime denotes the post-collision quantities and the subscript 1 denotes the collision partner. The meaning of each term in Eq. (2.1) is described in the following;

1. The first term on the left hand side of the equation represents the time variation of the distribution function of the particles.
2. The second term gives the spatial variation of the distribution function.
3. The third term describes the effect of a force on the particles.
4. The term at right hand side of the equation is called the collision integral. It is the source of most of the difficulties in obtaining solutions of the Boltzmann equation.

However, the Boltzmann equation is rarely used to numerically solve the practical problems because its higher dimensionality (up to seven) and the difficulties of modeling the integral collision term. Instead, the DSMC method has been used to simulated problems involving rarefied gas dynamics, which is the main topic in the current thesis.

2.2 General Description of the Standard DSMC

Due to the expected rarefaction caused by the very small size of micro-scale devices or the rarefied gas flows, the current research is performed using the DSMC method [10, 11], which is a particle-based method. The basic idea of DSMC is to calculate practical gas flows through the use of a method that has a physical rather than a mathematical foundation, although it has been proved that the DSMC method is equivalent to solving the Boltzmann equation [12, 13]. The assumptions of molecular chaos and a dilute gas are required by both the Boltzmann formulation and the DSMC method [10, 11]. The molecules move in the simulated physical domain so that the physical time is a parameter in the simulation and all flows are computed as unsteady flows. An important feature of DSMC is that the molecular motion and the intermolecular collisions are uncoupled over the time intervals that are much smaller than the mean collision time. Both the collision between molecules and the interaction between molecules and solid boundaries are computed on a probabilistic basis and, hence, this method makes extensive use of random numbers. In most practical applications, the number of simulated molecules is extremely small compared with the number of real molecules. The general procedures of the DSMC method are described in the next section, and the consequences of the computational approximations can be found in Bird [10, 11].

In real molecular collision, the force between molecules is strongly repulsive at short distance and weakly attractive at larger distance. Models for analytical and numerical studies involve some degree of approximation. These models are developed to imitate the real particle collision according to experiment. There are three molecular collision models, which are the Hard Sphere (HS), Variable Hard Sphere (VHS) and Variable Soft Sphere (VSS) molecular models, in the standard DSMC method [11]. The total collision cross section of the hard sphere model is proportional to the square of the constant diameter. It has the advantage of easily calculated collision mechanics because of the isotropic scattering that means all directions are equally possible for the post-collision velocity in the center-mass frame of reference. But the cross-section should vary with relative velocity in reality. The variable hard sphere (VHS) model proposes the collision diameter is a function of relative speed, which can predict the viscosity more accurately. The cross-section is determined from the viscosity coefficient, but the ratio of the momentum to the viscosity cross-section follows the hard sphere value. Thus, the variable soft sphere (VSS) model is developed to predict the correct viscosity and diffusion coefficients, which the scattering of post-collision is not isotropic anymore.

The procedure for the collision is based on the cell, which collision pairs are chosen from the cells. The correct probability of collision between two particles is proportional to the product of their relative speed and total collision cross-section. The collision pairs then are chosen by the acceptance-rejection method. The no time counter (NTC) method is an efficient method for molecular collision. This method yield the exact collision rate in both simple gases and gas mixtures, and under either equilibrium or non-equilibrium conditions.

Note that the corresponding molecular data including reference diameter (d_{ref}), reference temperature (T_{ref}), and the viscosity temperature exponent (ω) for each species are taken from those listed in Ref. [11]. Solid walls for all cases considered in this study are assumed to be fully diffusive (100% thermal accommodation), unless otherwise specified.

2.3 The Standard DSMC Procedures

Figure 2.1 is a general flowchart of the DSMC method. Important steps of the DSMC method include setting up the initial conditions, moving all the simulated particles, indexing (or sorting) all the particles, colliding between particles and sampling the molecules within cells to determine the macroscopic quantities. The details of each step will be described in the following subsections.

2.3.1 Initialization

The first step to use the DSMC method in simulating flows is to set up the geometry and flow conditions. A physical space is discretized into a network of cells and the domain boundaries have to be assigned according to the flow conditions. A point has to be noted is the cell dimension should be smaller than the mean free path, and the distance of the molecular movement per time-step should be smaller than the cell dimension. After the data of geometry and flow conditions have been read in the code, the numbers of each cell is calculated according to the free-stream number density and the current cell volume. The initial particle velocities are assigned to each particle based on the Maxwell-Boltzmann distribution according to the free-stream velocities and temperature, and the positions of each particle are randomly allocated within the cells.

2.3.2 Particle Movement

After initialization process, the molecules begin move one by one, and the molecules move in a straight line over the time-step if it did not collide with solid surface. For the standard DSMC code by Bird [10, 11], the particles are moved in a structured mesh. There are two possible conditions of the particle movement. First is the particle movement without interacting with solid wall. The particle location can be easy located according to the velocity and initial locations of the particle. Second is the case that the particle collides with solid boundary. The velocity of the particle is determined by the boundary type. Then, the particle continues its journey from the intersection point on the cell surface with its new absolute velocity until it stops. Although it is easier to implement by using structured mesh, it is difficult for those flows with complex geometry.

2.3.3 Indexing

The location of the particle after movement with respect to the cell is important information for particle collisions. The relations between particles and cells are reordered according to the order of the number of particles and cells. Before the collision process, the collision partner will be chosen by a random method in the current cell. And the identities of the collision partners can be easy determined according to this numbering system.

2.3.4 Gas-Phase Collisions

The other most important phase of the DSMC method is gas phase collision. The current DSMC method uses the no time counter (NTC) method to determine the correct collision rate in the collision cells. The number of collision pairs within a cell of volume V_c over a time interval Δt is calculated by the following equation;

$$\tfrac{1}{2} N \overline{N} F_N (\sigma_T c_r)_{\max} \Delta t / V_c \qquad (2.2)$$

N and $\overline{N}$ are fluctuating and average number of simulated particles, respectively. F_N is the particle weight, which is the number of real particles that a simulated particle represents. σ_T and c_r are the cross section and the relative speed, respectively. The collision for each pair is computed with probability

$$(\sigma_T c_r)/(\sigma_T c_r)_{\max} \qquad (2.3)$$

The collision is accepted if the above value for the pair is greater than a random fraction. Each cell is treated independently and the collision partners for interactions are chosen at random, regardless of their positions within the cells. The collision process is described sequentially as follows:

1. The number of collision pairs is calculated according to the NTC method, Eq. (2.2), for each cell.
2. The first particle is chosen randomly from the list of particles within a collision cell.
3. The other collision partner is also chosen at random within the same cell.
4. The collision is accepted if the computed probability, Eq. (2.3), is greater than a random number.
5. If the collision pair is accepted then the post-collision velocities are calculated using the mechanics of elastic collision. If the collision pair is not to collide, continue choosing the next collision pair.
6. If the collision pair is polyatomic gas, the translational and internal energy can be redistributed by the Larsen-Borgnakke model [72], which assumes in equilibrium.

The collision process will be finished when all the collision pairs are handled for all cells and then progress to the next step.

2.3.5 Sampling

After the particle movement and collision process finishes, the particle has updated positions and velocities. The macroscopic flow properties in each cell are assumed to be constant over the cell volume and are sampled from the microscopic properties of each particle within the cell. The macroscopic properties, including density, velocities and temperatures, are calculated in the following equations [10, 11];

$$\rho = nm \tag{2.4a}$$

$$c_o = \bar{c} = \overline{c_o} + \overline{c'} \tag{2.4b}$$

$$\frac{3}{2}kT_{tr} = \frac{1}{2}m(\overline{u'^2} + \overline{v'^2} + \overline{w'^2}) \tag{2.4c}$$

$$T_{rot} = \frac{2}{k}(\varepsilon_{rot}/\zeta_r) \tag{2.4d}$$

$$T_v = \frac{2}{k}(\varepsilon_v/\zeta_v) \tag{2.4e}$$

$$T_{tot} = (3T_{tr} + \zeta_{rot}T_{rot} + \zeta_v T_v)/(3 + \zeta_{rot} + \zeta_v) \tag{2.4f}$$

n, *m* are the number density and molecule mass, receptively. *c*, c_o, and *c'* are the total velocity, mean velocity, and random velocity, respectively. In addition, T_{tr}, T_{rot}, T_v and T_{tot} are translational, rotational, vibrational and total temperature, respectively. ε_{rot} and ε_v are the rotational and vibrational energy, respectively. ζ_{rot} and ζ_v are the number of degrees of freedom of rotation and vibration, respectively. If the simulated particle is monatomic gas, the translational temperature is regarded simply as the total temperature. Vibrational effect can be neglect if the temperature of the flow is low enough.

The flow will be monitored if steady state is reached. If the flow is under unsteady situation, the sampling of the properties should be reset until the flow reaches steady state. As a rule of thumb, the sampling of particles starts when the number of molecules in the calculation domain becomes approximately constant.

2.4 Overview of the Current Implementation of PDSC

So far, there have only several well-developed DSMC codes which mentioned in Section 1.2.2. It is very helpful to develop a general-purpose parallel DSMC code for simulating problems of rarefied gas flows. Thus, development of this numerical solver is the main subject of this thesis. To make the PDSC friendlier to use, an applicable preprocessor and postprocessor are necessary. Figure 2.2 shows the overview of the planed PDSC, which includes a graphic-based preprocessor, main numerical solver and a postprocessor. They are described in the next sections in tern.

2.4.1 MuST Visual Preprocessor

A graphical user interface (GUI), which names MuST Visual Preprocessor (Fig. 2.3(b)), is developed by Professor Wu's group [73]. It is used to ease the parameter settings of boundary conditions, initial conditions and parallel processing because these procedures are tedious and complicated as can be shown in Fig. 2.3(a). This preprocessor can be either used for cell-based (e.g. DSMC) or for node-based (Finite Element Method) numerical methods. Firstly, the MuST Visual Preprocessor will ask for mesh connectivity information, which is easy to obtain from mesh generators. The mesh types can be quadrilaterals, triangles, hexahedrons, tetrahedrons, prisms and pyramids. Secondly, the preprocessor will transfer mesh format into graph and process the initial partition according to the weighting of each cell. Thirdly, the processor can create the boundary surfaces automatically and we can assign the boundary types very easily to create the input files for the PDSC.

2.4.2 PDSC

Parallel **DS**MC **C**ode (PDSC) is the main solver developed in this thesis, which utilizes unstructured tetrahedral mesh. Figure 2.4 is the features of PDSC and brief introduction is listed in the following paragraphs.

Unstructured Tetrahedral Mesh

Reasons of PDSC using unstructured tetrahedral mesh are: (a) it can be easily used for flows with complicated boundary conditions, (b) parallel processing can be easier implemented via graph-partitioning technique, which can handle irregular inter-processor boundary of dynamic domain decomposition, (c) it can be coupled with unstructured node-based numerical method (e.g. N-S equations).

According to these advantages of using unstructured mesh, a special particle ray-tracing technique has to be designed to efficiently track the particle movement for the special grid system, unstructured grid, which we use in the current study. Briefly speaking, the movement of a particle is determined by the velocity and initial position of the particle. If the intersecting face is an I/O boundary, the particle will be removed. If not, then process the interaction according to the specified wall boundary condition. The details of particle ray-tracing techniques of two- and three-dimensional domain are described in Ref. [8, 74].

Collision Cross-Section Data

As mentioned in Section 2.2, the variable soft sphere (VSS) model can reproduce the viscosity and diffusion coefficients correctly. The relevant parameters of using VSS model for the DSMC method can be found in Bird's book [10, 11]. This reference provides some usual gaseous species. When the flow involves some special species, it has problem to obtain the relevant parameters of the VSS molecular model. To overcome this problem, a quantum chemistry method is proposed to calculate the intermolecular energy surface according to the distance between molecules [75]. Then the simulated intermolecular energy potential is fitted through the Lennard-Jones (L-J) potential to obtain the constants. Based on these constants and gas kinetic theory, the transport coefficients, which are viscosity and diffusion coefficients, are derived. Finally, the parameters of the VSS model are derived by fitting these computed coefficients to those derived from the VSS model.

Pressure Boundary Treatment

In order to perform accurate simulation for inflow/outflow pressure boundaries, general procedure for treating these conditions by using the concept of particles flux conservation is developed in PDSC [76]. This function is useful for applications of micro-manifold, micro-nozzle and slider air bearing.

Unstructured Adaptive Mesh with Variable Time-Step Scheme

To obtain accurate simulated results, two- and three-dimensional *h-refinement* adaptive mesh with variable time-step scheme is developed [39, 40]. Some parameters are used to determine the adaptive level and a simple cell quality control can prevent the creation of high aspect ratio cells. This module is not only valid for PDSC, but also suitable for other numerical simulators. The detail of adaptive mesh refinement and variable time-step scheme can be found in Chapter 3.

Parallel with Dynamic Domain Decomposition

To save the enormous computational cost of the standard DSMC code, a parallel DSMC with dynamic domain decomposition. Message passing interface (MPI) is used for data communication. This function can automatically repartition the graph domain according to the loading of each processor, which is the particle number of each cell, to achieve the load balancing of the simulation. It also can be used for other particle simulation and equation solvers. The detail of this feature is presented in Chapter 4.

Conservative Weighting Scheme

When the flow involving trace particle species, the simulation needs lots of simulated particles to satisfy the DSMC limit, which will lead to immense computational time. A weighting scheme is developed to deal with this kind of flows [77]. The basic concept is assigning the lower weight for trace particle species to create more simulated particles. This method does not use particle cloning and destroying to avoid the statistical error. The detail of conservative weighting scheme is introduced in Chapter 5.

Molecular Chemical Reaction

Finally, PDSC also has the function to simulate flows with chemical reactions. Chemistry is important and needs to be considered when the flow velocity and temperature is very high. The chemistry in PDSC is developed with help of Professor I. D. Boyd at the University of Michigan in United States. It has dissociation, exchange and recombination reactions in PDSC. Chapter 6 is a section of detailed introduction and validation of this feature.

2.4.3 Postprocessor

After the DSMC solver (PDSC), it needs a postprocessor to view the result of the simulation. The output data of PDSC can be transferred from cell-based data into node-base data easily and then importing into both Tecplot and Grapher for displaying purpose.

2.5 Concluding Remarks

This chapter has presented an overview of the current implementation of the DSMC method in brief. The first part of this chapter was concerned with the general description of the standard DSMC method, which is proven that it does provide solutions that are consistent with the Boltzmann equation. The detail of the DSMC method can be found in Ref. [10, 11]. The second part describes an overview of the current implementation of PDSC. A graphical user interface, which names MuST Visual Preprocessor, is developed to deal with complicated and tedious procedures. This preprocessor not only can be used for cell-based simulators (e.g. DSMC or PIC), but also for some node-based solvers (e.g. CFD or FEM). Then, several important features of the PDSC are also introduced to deal with different problems of flows. The next chapter will introduce the unstructured adaptive mesh with variable time-step scheme in DSMC method to improve the accuracy and efficiency of the computation.

Chapter 3
Unstructured Adaptive Mesh Refinement with Variable Time-Step Scheme

The accuracy of the DSMC method depends on the number of simulated particles and a suitable mesh. It is necessary to adapt the existing mesh according to the flow filed to achieve higher resolution of flow properties and apply the variable time-step scheme to obtain a more uniform particle distribution and efficient computation. Thus, in the current section, the general features of the proposed PDSC with variable time-step scheme combining mesh adaptation will be described in detail.

3.1 Variable Time-Step Scheme

The number of simulated particles per cell is related to the number density, cell volume and particle weight by the following relation;

$$N_P = \frac{nV}{W_P} \tag{3.1}$$

N_p is the number of simulated molecules of pth cell. W_p is the particle weight which is defined as ratio of the number of real particles to the number of simulated molecules, and n and V are the number density and the volume of the computational cell, respectively. If the number density is assumed to be a constant, the simulated particle count decreases by decreasing the cell volume or increasing the particle weight. As mentioned previously, mesh refinement can help to obtain a better cell-size distribution, ideally on the order of the local mean free path. The volume of a cell can then be related to density by the fact that the mean free path is inversely proportional to the number density. Thus, the relationship between these variables is shown as Eq. (3.2),

$$\Delta x \propto \lambda \propto n^{-1} \tag{3.2}$$

For two-dimensional flow, the cell volume is given by

$$\begin{aligned} V &\propto \Delta x \times \Delta x \\ &\propto \lambda^2 \propto n^{-2} \end{aligned} \tag{3.3}$$

For three-dimensional flow, the cell volume is given by

$$\begin{aligned} V &\propto \Delta x \times \Delta x \times \Delta x \\ &\propto \lambda^3 \propto n^{-3} \end{aligned} \tag{3.4}$$

Substituting Eqs. (3.2)~(3.4) into Eq. (3.1), give the following relation between number of simulated particles and flow density (assuming constant particle weight):

$$\begin{aligned} N_p &\propto n^{-1} \propto \Delta x \,, \quad 2D \\ N_p &\propto n^{-2} \propto \Delta x^2, \quad 3D \end{aligned} \tag{3.5}$$

According to Eq. (3.5), the number of simulated particle is inversely linear and square proportional to the number density with respect to the two- and three-dimensional flows. That is, the lower density regions have larger simulated particle numbers and the higher density regions have fewer simulated molecular numbers. This effect is more obvious in three-dimensional simulation. This will lead to computational waste and incorrect results at lower and higher density regions, respectively. To avoid this problem, a variable time-step scheme is proposed to obtain a more uniform particle distribution as follows:

From Eq. (3.5), the density distribution is inversely proportional to the dimension of the cell. Thus, the first step of variable time-step scheme is to find out the cell, which has the minimum cell volume ($V_{\min}$), and to calculate the local time-step of the cell as Eq. (3.6). The time-step is also proportional to Δx and inversely proportional to the number density,

$$\begin{aligned} \Delta t_{\min,2D} &= \frac{\sqrt{V_{\min}}}{3\times(U_{mean}+\sqrt{2kT/m})} \propto \Delta x \propto n^{-1}, \quad 2D \\ \Delta t_{\min,3D} &= \frac{\sqrt[3]{V_{\min}}}{3\times(U_{mean}+\sqrt{2kT/m})} \propto \Delta x \propto n^{-1}, \quad 3D \end{aligned} \tag{3.6}$$

U_{mean} and $\sqrt{2kT/m}$ are the mean and thermal velocity, respectively. Then, each local time-step Δt_j of each cell can be assigned based on $\Delta t_{\min}$ and the cell volume V_j as Eq. (3.7).

$$\begin{aligned} \Delta t_i &= \sqrt{\frac{V_i}{V_{\min}}} \times \Delta t_{\min,2D}, \quad 2D \\ \Delta t_i &= \sqrt[3]{\frac{V_i}{V_{\min}}} \times \Delta t_{\min,3D}, \quad 3D \end{aligned} \tag{3.7}$$

where V_i and Δt_i are the volume and local time-step of i^{th} cell. Sketch of the concept of the variable time-step scheme for a simulated particle moves across the cell interface is illustrated in Fig. 3.1.

Basic idea of variable time-step method in PDSC is to enforce the flux conservation (mass, momentum and energy) of moving simulated particle when crossing the interface between two neighboring cells. If we scale the local cell time-step to the local cell size (or local mean free path), then the best way to enforce flux conservation is to change the particle weight factor without destroying or cloning the particles during particle movement across the cell interface. The cloning of particle can generally induce unpredictable random-walk effects in a statistical simulation like DSMC. One of the advantages in implementing the variable time-step scheme is to reduce both the simulated particle numbers and transient time-step to steady state, when the sampling normally starts in DSMC. This will result in appreciable time saving for the steady DSMC simulation. The net flux of the physical particles, including mass, momentum and kinetic energy, should be enforced conservation when a simulated particle crosses the cell interface form the cell 1 and to the cell 2. Thus,

$$f_1 = \frac{N_1 W_1 \Phi_1}{A \times \Delta t_1} u = \quad f_2 = \frac{N_2 W_2 \Phi_2}{A \times \Delta t_2} u \tag{3.8}$$

where W's, Φ's (=m, mv, $mv^2/2$ or other internal energy) and Δt's are the particle weight, conserved flux quantity and time-step, respectively, and the numbers at subscript represents cell numbers. Note that A represents the area of cell interface between cell 1 and 2. N_2 is number of the simulated particle in cell 2, which originated from cell 1. There are several choices of the corresponding parameters to satisfy Eq. (3.8), with which we can play. The best choice is to set N_2=1 (without particle cloning or destroy) and to keep $\Phi_1 = \Phi_2$ without changing the velocity across the cell interface, Eq. (3.8) can be rewritten as Eq. (3.9)

$$\frac{W_1}{\Delta t_1} = \frac{W_2}{\Delta t_2} \tag{3.9}$$

In other words, $\frac{W_i}{\Delta t_i}$ will be the same for all cells throughout the computational domain.

Inserting Eqs. (3.7)~(3.9) into Eq. (3.1), the number of simulated particles per cell is,

$$\begin{aligned} N_P &= \frac{nV}{W_P} \propto \frac{\Delta x^{-1} \times \Delta x^2}{\Delta t} \propto \frac{\Delta x}{\Delta t} \propto 1, \qquad 2D \\ N_P &= \frac{nV}{W_P} \propto \frac{\Delta x^{-1} \times \Delta x^3}{\Delta t} \propto \frac{\Delta x^2}{\Delta t} \propto \Delta x, \qquad 3D \end{aligned} \tag{3.10}$$

Using this approach, resulting number of simulated particles per cell for the three-dimensional flow scales with Δx ($\sim\sqrt[3]{V_c}$, V_c is the cell volume) [43] if cell size is proportional to the local mean free path, which otherwise scales with $(\Delta x)^2$. In doing so, the simulated particle will only have to adapt its weight that is proportional to the size of time-step, which is approximately commensurable to the local mean free path if solution-based adaptive mesh is used. Of course, the remaining time for a simulated particle, when crossing cell interface, should be rescaled according to the ratio of time-steps in original and destination cells. In the PDSC, the procedure of variable time-step scheme is listed in the following;

1. Chose a minimum cell volume to calculate the reference time-step, Eq. (3.6).
2. Assign the time-step for each cell based on the cell size, Eq. (3.7).
3. Determine the particle weight for each cell by Eq. (3.9).
4. The time-step has to be modified if the particle crosses the cell interface.

By using this variable time-step scheme, the simulated particle number and transient time will be reduced to speed up the computing.

3.2 Unstructured Adaptive Mesh Refinement

3.2.1 General Features

Based on the reviews in previous chapter and considering the applications of DSMC, the general features of an adaptive mesh generation scheme are proposed as follows: (1) unstructured mesh (triangular, quadrilateral and hybrid in two-dimensional mesh and tetrahedral in three-dimensional mesh); (2) *h-refinement* with mesh embedding; (3) local adaptation criteria and free-stream parameter (relative local density ratio to free-stream value) as the mesh adaptation parameters; (4) upper limit on maximum number of levels of mesh adaptation. And the variable time-step scheme is incorporated into the unstructured adaptive mesh to reduce the number of simulators.

All mesh adaptation methods need some means to detect the requirement of local mesh refinement to better resolve the features in the flow fields and hence to achieve more accurate numerical solutions. This also applies to DSMC. It is important for the adaptation parameters to detect a variety of flow features but does not cost too much computational time. Often gradients of properties, such as pressure, density or velocity, are used as the adaptation parameters to detect rapid changes of the flow-field solution in traditional CFD. However, by considering the statistical nature of the DSMC method, density is adopted instead as the adaptation parameter. Using density as the adaptation parameter in DSMC is justified since it is generally required that the mesh size be much smaller than the local mean free path to better resolve the flow features, as mentioned previously. Thus, density is used to determine the mesh adaptation in the current study.

3.2.2 Adaptation Parameter and Criteria

There are two adaptation criteria to use if the cell should be refined or not. One is the local cell Knudsen number, Kn_c, based on density. The other is the free-stream parameter ϕ_i. The details are described in the following.

Firstly, to use the density as an adaptation parameter, a local cell Knudsen number is defined as

$$\begin{aligned} Kn_c &= \frac{\lambda_c}{\sqrt{A_c}}, \quad 2D \\ Kn_c &= \frac{\lambda_c}{\sqrt[3]{V_c}}, \quad 3D \end{aligned} \tag{3.11}$$

where λ_c is the local cell mean free path based on HS model and A_c and V_c are the magnitude of local cell area and local cell volume of two- and three-dimensional domain, respectively. When the mesh adaptation module is initiated, the local cell Knudsen number for each cell is computed and compared with a preset value, Kn_{cc}. If this value is less than the preset value, then mesh refinement is required. If not, check the next cell until all cells are checked. This adaptation parameter is expected to be most stringent on mesh refinement (more cells are added); hence, the impact to PDSC computational cost might be high, but is required to obtain an accurate solution.

Secondly, considering the practical applications of mesh adaptation in external flows, we have added another constraint, $\phi_i \geq \phi_0$, where ϕ_i, free-stream parameter of each cell, is defined as

$$\phi_i = \frac{\rho_i}{\rho_\infty} \tag{3.12}$$

and ϕ_0 is a preset value. Not only the above constraint helps to reduce the total refined cell numbers to an acceptable level by reducing the cell numbers in the free-stream region a great deal, but also reduces the total computational time up to 30% as can be shown later.

3.2.3 Adaptation Procedures

In our simulations, the *h-refinement* has been developed to obtain a reasonable adaptive mesh. The concepts of the adaptation procedures of two- and three-dimensional simulation are similar, but the details are complicated and non-trivial. Before outlining the procedures of mesh adaptation, three general rules are described as follows:

(1) *Isotropic mesh refinement is employed for those cells, which flag for mesh refinement.* A new node is added on each edge (face) of a *parent cell* and connecting them to form *child cells*. In general, this will create hanging nodes in the interfacial cell, which is not refined, next to the isotropically refined cell. Existence of hanging node(s) not only complicates the particle movement, but also increases the cost of the cell-by-cell particle tracing due to the increase of face numbers. Hence, a remedy is proposed as follows in item (2).

(2) *An-isotropic mesh refinement is utilized in the (interfacial) cells next to those cells have just been isotropically refined.* The child cells are formed no matter what type of the interfacial cell is, considering the generality of the practical programming. However, some special treatment is required. The removal of hanging node(s) in the interfacial cells does increase the computational cost; it is, however, trivial as compared with the disadvantages caused by the hanging node(s).

(3) *Isotropic mesh refinement is used to remove the cells of high aspect ratio.* In the interfacial regions between refined and unrefined cells, bad quality cells (large aspect ratio) do occur. In DSMC, this will introduce large errors when particles are collided in the cell if it is treated the same as other normal cells. The remedy to remove these high aspect ratio cells is recorded these cells, which has been refined by an-isotropic adaptation. If these labeled cells suffering the second an-isotropic refinement, these cells will be forced to refine by isotropic adaptation.

The mesh adaptation procedures are performed after enough samples of data at each original cell are gathered. As a rule of thumb, about 50,000 particles sampled in a cell are considered enough for the mesh adaptation purpose. The mesh adaptation module is initiated and checks through all the cells to determine if mesh enrichment is required based on the specific adaptation parameter, which was explained previously. If mesh enrichment is conducted, associated neighbor identifying arrays are updated or created, coordinates and number of face for new cells are recorded, and sampled data on the coarse *parent cell* are redistributed (based on the magnitude of cell size) to the finer *child cells* accordingly. The above procedures are repeated until the prescribed maximum number of adaptation levels has been reached or no mesh enrichment is required for all the cells in the computational domain. Before preceding the DSMC computation using the most updated mesh, all sampled data are reset to zero. Finally, the DSMC computation is then conducted on the final refined mesh to accumulate enough samples for obtaining the macroscopic properties in the cells.

In summary, the following steps summarize the procedures for mesh refinement:

1. Set up initial grids and input data.
2. Process computation until enough sampled data are gathered at each cell.
3. Compute the adaptation parameters of each cell by using Eq. (3.11) and Eq. (3.12).
4. Refine all the cells in which both the Kn_c is less than the preset Kn_{cc} and ϕ_i is larger than ϕ_0 by conducting isotropic mesh refinement. If the adaptation criteria are not met, go to step (8).
5. Create and update the neighbor identifying arrays, coordinates, face numbers, and distribute sampled data to child cells, respectively. Reduce the simulation time-step to half.
6. Check if there are any hanging nodes in the interfacial cells. If it does, then conduct an-isotropic mesh refinement. Also create and update associated cell data as described in step (5).
7. Return to (2) if both the accumulated adaptation levels are less than the preset maximum value and mesh refinement is required.
8. If the accumulated adaptation levels are greater than the preset value or no mesh refinement is required, then reset all sampled data to zero and precede the DSMC computation as normal.

The corresponding flowchart is illustrated in Fig. 3.2 Note that the proposed mesh adaptation is capable of refining the mesh close to the body surface following the real surface geometry if the surface contour can be cast into parametric function format. The details of the adaptation procedure are present as follows.

3.2.3.1 Two-Dimensional Adaptation

These rules applied to quadrilateral and triangular mesh in two-dimensional simulation, as illustrated in Fig. 3.3 and Fig. 3.4, respectively. The mesh adaptation can be categorized into isotropic and an-isotropic.

1. *Isotropic adaptation:*

   If the cell is suggested to be isotropic adaptation, quadrilateral and triangular child cells are formed with respect to the parent cell. In general, this will create one to three hanging nodes in the interfacial cell, which is not refined, next to the isotropically refined cell. For example, for a quadrilateral interfacial cell with three or four hanging nodes (Fig. 3.3), an isotropic cell refinement is conducted and forms four quadrilateral parent cells.

2. *An-isotropic adaptation:*

   For the an-isotropic adaptation, triangular child cells are formed no matter what type of the interfacial cell is, considering the generality of the practical programming. Typical methods of interfacial mesh refinement in the quadrilateral and triangular cells are also shown in Figs. 3.3 and 3.4, respectively. However, some special treatment is required. For example, for a quadrilateral interfacial cell with one or two hanging nodes (Fig. 3.3), an isotropic cell refinement is conducted to form three or four triangular cells, respectively.

3.2.3.2 Three-Dimensional Adaptation

The basic ideas of refining the 2-D unstructured mesh, first by isotropic refinement, then by an-isotropic refinement, remain the same for the three-dimensional tetrahedral unstructured mesh, although more complicated conditions will be encountered. The adaptation has two kinds of refinement, which are isotropic and an-isotropic refinements as 2-D adaptation. When the cell is determined to be refined, it will process the isotropic refinement, that is, the original cell will be divided to eight subcells equally. This process will create hanging nodes for the neighboring cells. In order to remove those hanging node, an-isotropic mesh is processed to over come the difficulty of this problem. The tree diagram of mesh adaptation is shown as Fig. 3.5.

No matter the parent cell is adapted by isotropic or an-isotropic mesh refinement. The child cells are all formed as tetrahedral cell. As Fig. 3.6 illustrates, there are six types with different hanging node numbers. The tetrahedral interfacial cell having one to five hanging node always forms tetrahedral child cells by adding appropriate node(s) on the edge(s) of the interfacial cell. The number of refined cells in the interfacial tetrahedral cell is either two, four or eight depending upon the distribution of the hanging nodes. Rules of removing hanging nodes can be briefly summarized as follows:

1. *An-isotropic adaptation:*

   a. *One hanging node (Type 1)*: Forming two tetrahedral child cells by directly connecting the hanging node to the other two nodes in the interfacial cell.

   b. *Two and three coplanar hanging nodes (Type 2a, 3b)*: In the case of the two coplanar hanging nodes, one node is added to the same coplanar face. The cell, with these two types with different hanging nodes, will be refined as four tetrahedral child cells by an-isotropic adaptation.

2. *Isotropic adaptation:*

   a. *Others (Type 2b, 3a, 4, 5, 6)*: Forming eight tetrahedral child cells by adding appropriate nodes to fill up the six edges of the interfacial cell and then form the child cells as that in isotropic mesh refinement described earlier.

Usually, no matter for the two- or three-dimensional adaptation, the an-isotropic adaptation will form two or four child cells with high aspect ratio. These cells with bad quality maybe lead to poor simulated results. Thus, some modifications must be operated to improve the cell quality. Isotropic mesh refinement is again used to remove the cells having high aspect ratio, as shown in Fig 3.7, which represents the simplest approach of mesh quality we can think of. The method is first marked the cell with an-isotropic adaptation. Then, these cells will be forced to refine by isotropic adaptation if these labeled cells suffering second an-isotropic adaptation. It is effective in improving the mesh quality, although it is simple. Besides, one other important step during the mesh adaptation procedures is to modify the old cell neighbor-identifying arrays and add the new cell neighbor-identifying arrays accordingly.

3.3 Verifications of Unstructured Adaptive Mesh Refinement and Variable Time-Step Scheme

The test cases include two-dimensional and three-dimensional simulations. A hypersonic cylinder flow and a 15^{o}-compression ramp flow are simulated in two-dimensional domain. A three-dimensional supersonic flow over a sphere is also presented as a benchmark test. Related issues about the advantages of using variable time-step method and unstructured adaptive mesh will be discussed along with the presentation of simulated results. Results are then compared with experimental data and previous simulation wherever available. In addition, physics of the flow field related to these problems will be described as brief as possible, since we are interested in demonstrating the code capability at this moment.

3.3.1 Two-Dimensional Flows

Hypersonic Flow over a Cylinder

<u>Flow and Simulation Conditions</u>

Flow conditions are the same as those of Koura and Takahira [78] and represent the experimental conditions of Bütefisch [79]. This flow problem is chosen to demonstrate the capability of resolving the expected high density in the stagnation region and the high-density gradient across the detached bow shock around the cylinder. For completeness, they are briefly described here as follows: Variable Hard Sphere nitrogen gas, free-stream Mach number M_{∞}=20, free-stream number density n_{∞}=5.1775E19 particles/m^3, free-stream temperature T_{∞}=20 K, fully thermal accommodated and diffusive cylinder wall with T_w/T_0=0.18, where T_w and T_0 are the wall and stagnation temperatures, respectively. Temperature dependent rotational energy exchange model of Parker [80] is used to model the diatomic nitrogen gas with the following parametric setting: limiting rotational collision number Zr_{∞}=21, potential well-depth temperature T^*=79.8 K. Resulting Knudsen number based on free-stream condition is 0.025, based on the free-stream mean free path and diameter of the cylinder. The diameter of the cylinder is 1m. An unstructured triangular mesh is used for the simulation. Sketch of computational domain and complete listing of physical and VHS parameters are shown in Fig. 3.8 and Table 3.1. The variable time-step (VTS) and constant time-step (CTS) schemes with adaptive mesh are used in this simulation.

Mesh Adaptation Concerns

Corresponding adaptation criteria for mesh adaptation is Kn_{cc}=1.1 with maximum number of adaptation levels equal to 4. Additional constraint, free-stream parameter, ϕ_0 =1.05, which reduces greatly the final refined total cell numbers, is used not to refine those cells with normalized density ratio close to unity (within 5% in this case). The side effect of this constraint might increase the skew of the interfacial cells between un-adaptive free-stream cells and adaptive cell; however, the reduction of computational cost is appreciable and up to approximately 20-30%. Final free-stream cell size is expected to be much longer than the local mean free path; however, the solution is not expected to deteriorate since nearly uniform flow properties prevail in the free-stream region. Thus, initial 7,025 triangular cells are used for the simulation.

Evolution of adaptive mesh at each level (only 0,2,4 shown) with cell quality control is presented in Fig. 3.9 and corresponding results are summarized in Table 3.2. In Table 3.2, the cell numbers increase from 7,025 to 75,099 after four levels of mesh adaptation, the number of cells is much less than that used by Koura and Takahira [78], which had 200,000 cells, but the positioning of the cells in the present study may be superior to theirs due to the mesh adaptation scheme applied. As illustrated in Figs. 3.9, the mesh is refined across the strong bow shock around the cylinder as well as the stagnation region in front of the cylinder. It is clearly that the proposed mesh adaptation method captures the important flow features such as the bow shock in this case. We would expect the results in these mesh-refined regions to be better than those without mesh adaptation.

Comparisons of the Results Using Different Meshes

Figure 3.10 is a normalized density contour with different meshes. The upper and the lower figures are the density contours with the original and the level-4 adaptive meshes, respectively. There are some results that we need to note. First, a rather strong bow shock stands off at some distance away from the cylinder. The flow is highly compressed across the nearly normal shock to the stagnation point, where density increases tremendously. Second, the flow is slightly compressed across the oblique shock away from the cylinder and then is slightly expanded further downstream. A relatively rarefied region (as compared with free-stream) with the size of cylinder diameter is formed with density ratio less than 0.5 behind the cylinder since most gas particles are directed away from the cylinder across the oblique shock as discussed earlier. Third, maximum values of normalized density are 14.4 and 39.3 at the stagnation point due to the highly refined mesh in this region, while minimum values of 0.279 and 0.283 are observed just behind the cylinder with the initial and adaptive mesh, respectively. Figure 3.11 presents the comparisons of the contours of normalized temperatures with different resolution meshes. The distributions of translational and rotational temperatures can be demonstrated clearly in this figure, where T_{tr} and T_{rot} represent the translational and rotational temperature, respectively. In general, the trends of the results with different meshes are similar, but the distributions of value are different. These figures show that these temperatures are increased according to the bow shock and reached to a maximum value at the stagnation point. And then it decreases away from the bow shock and the cylinder by expansion effect. Clearly, strong temperature non-equilibrium exists in the bow shock especially for the regions near the stagnation line.

Centerline Properties Distribution

Results of normalized number density (n/n_∞), and normalized translational and rotational temperatures ($(T-T_\infty)/(T_o-T_\infty)$) along the stagnation line are presented in Fig. 3.12. Previous experimental data of Bütefisch [79] and DSMC data of Koura and Takahira [78] are also included in these figures for comparison. In these figures, normalized density and temperature are nearly the same and the agreements are remarkable, except for the data in front of the cylinder. Also appreciable thermal non-equilibrium occurs in the wake and shear layer around the cylinder. The simulated results with initial mesh are under predicted due to the fact that the cell is too coarse for solving the flow field. However, thermal non-equilibrium due to complicated flow field is well resolved using the adaptive mesh.

Comparisons of Adaptive Mesh With or Without Cell Quality Control

Distribution of the adaptive mesh with or without cell quality control after fourth level adaptation is presented in Fig. 3.13. As this figure illustrated, the refined cells with high aspect ratio are removed by isotropic adaptation with cell quality control. Figure 3.14 illustrates normalized density and temperatures along the stagnation line. These results are approximately the same and agree with the previous experimental and simulated data. Although the discrepancy of the present results is not obvious, it is intuitive that the results with good cell quality should be more accurate.

Comparisons of the Results Using CTS and VTS Schemes

The constant time-step scheme (CTS) and variable time-step scheme (VTS) are both used with the same adaptive mesh (level-4) in this simulation. Both the simulation time-steps are set as 20,000. Variation of particle distribution is reduced greatly as illustrated in Fig. 3.15, which shows the comparison of distribution of averaged number of particles per cell on level-4 adaptive mesh using constant time-step (CTS) and variable time-step (VTS) methods. In this figure, the upper part is the result using constant time-step scheme. It is clear that the particle distribution is extremely non-uniform. The free-stream and wake regions have larger particle numbers, while fewer particle numbers at the bow shock region, due to the cell dimension and density effect as mentioned in Section 3.1. The particle distribution seems more uniform and desirable which prevents the waste of computational time. In addition, using VTS method, the reduction of averaged number of particles per cell can be as large as 10-fold and 30-fold in the regions of oblique bow shock and wake region behind the cylinder, respectively. This is achieved by keeping the particle weight the same in the reference cell (minimum cell, near the stagnation point in this case) for both VTS and CTS methods. In the other words, the statistical uncertainty in the minimum cell is kept the same for comparing both methods. Resulting simulated particles at steady state are reduced from 0.85 million, using constant time-step (CTS), to 0.16 million, using variable time-step (VTS) in this case (Fig. 3.16). The number of simulated particles using variable time-step scheme can be reduced about five times of the constant variable time-step scheme. In addition, another benefit of applying VTS method to unstructured adaptive mesh is that it can reduce dramatically the number of iterations of transient period towards steady state, as illustrated in Fig. 3.16. In addition, the required number of iterations for transient period, when VTS is applied, is only about 25% of that if CTS is applied. In the current case, if we expect roughly the same statistical uncertainty in the minimum cell to obtain macroscopic properties for both CTS and VTS methods, the combination of VTS and unstructured adaptive mesh could save the computational time up to one order of magnitude.

Density Distributions

Figures 3.17 and 3.18 illustrate comparisons of normalized density and temperature contours with different time schemes, respectively. The upper and lower regions are the results using CTS and VTS methods, respectively. All the properties distributions are almost the same except for the density distribution at the wake region due to a relative fewer particles are applied by variable time-step scheme.

Centerline Properties Distribution

Normalized density and temperatures along the stagnation line are shown in Fig. 3.19. The results of variable time-step method seem more agreeable with previous experimental by Bütefisch [79] and simulated results by Koura and Takahira [78] than constant time-step scheme. Therefore, the variable time-step scheme not only can reduce the computational time but also obtains a more accuracy results in this case.

Hypersonic Flow Over a 15^o-Compression Ramp

Flow and Simulation Conditions

A hypersonic flow over a flat plate with a 15°-compression ramp is investigated by Holden and Moselle [81] and Robinson [24]. The corresponding boundary settings for simulation and general features of flow field are depicted in Fig. 3.20. The flow conditions are briefly described in the following; Variable Hard Sphere nitrogen gas, free-stream Mach number M_∞=14.36, free-stream density ρ_∞ and temperature T_∞ are 5.221E-4 kg/m^3 and 84.83 K, the length of the cylinder, X_c, and ramp, X_r, are 43.891 cm and 36.86 cm, respectively, fully thermally accommodated and diffusive flat and ramp wall with T_w =294.4 K. Resulting Knudsen numbers and Reynolds numbers based on X_c are 0.0002 and 1.04E5, respectively. Constant rotational energy exchange model is used with the rotational collision number Zr=5. Vibration energy transfer is neglected due to the low temperature involved. The complete listing of physical and VHS parameters are summarized in Table 3.3

<u>Mesh Adaptation Concerns</u>

In this simulation, free-stream parameter ϕ_0 and adaptation criteria Kn_{cc} for mesh adaptation are set to 1.05 and 1.1, respectively. The cell number of the initial triangular mesh is 15,063 and the maximum number of adaptation levels equal to 2. The simulation procedure is the same as the cylinder flow.

Figure 3.21 is the evolution of adaptive mesh at each level with cell quality control and corresponding results are summarized in Table 3.4. The number of cells increases from 15,063 to 83,776 after two levels mesh adaptation. The number of cells is larger than that used by Robinson [24], which had 66,482 cells. The numbers of simulated particles are about 350,000 and 640,000 of the present and Robinson, respectively. And the simulation time-steps are 24,000 and 160,000 of the present and Robinson, respectively. The mesh is refined across a weak leading-edge shock stands off at several mean free path from the tip of the plate. At the same time, a viscous boundary layer grows downstream along the flat plate due to the low Reynolds number laminar flow (Re_L=1x10^5). As mentioned earlier, the layer is thickened greatly by the pressure rise caused by the compression ramp. It is clearly captured in this figure. We would expect the results in these mesh-refined regions are better than those without mesh adaptation.

<u>Comparisons of the Results Using Different Meshes</u>

Results of normalized density (ρ/ρ_∞) contour with initial and level-2 adaptive meshes are presented in Fig. 3.22. Both the contours have the similar trend but the distributions are much different especially at the weak leading-edge shock and the portion of the ramp. The maximum density ratio occurs at the mid portion of the ramp, where the leading-edge shock and the compression ramp shock interact with each other (Type VI interaction in Fig. 3.20). These ratios of initial and level-2 adaptive meshes are 6.39 and 8.79, respectively. The minimum value of density ratio above the plate is lower than ambient value in the early portion of the plate due to the leading-edge shock, and the values are 0.28 and 0.25 with respect to initial and level-2 adaptive meshes.

Figures 3.23 illustrates the pressure ($C_p = \dfrac{p - p_\infty}{\frac{1}{2}\rho u_\infty^2}$), shear stress ($C_f = \dfrac{\tau}{\frac{1}{2}\rho u_\infty^2}$) and heat transfer ($C_h = \dfrac{q}{\frac{1}{2}\rho u_\infty^3}$) coefficient distributions, along the solid wall, with initial and level-2 adaptive meshes. Previous experimental data of Holden and Moselle

[81] and the DSMC data of Robinson [24] are also included in these figures for comparison. In these figures, the circle hollow symbols and the solid line represent the experimental by Holden *et al.* and simulated data by Robinson, respectively. The lines with hollow triangle, hollow quadrilateral are the present data with respect to use initial and adaptive meshes. As shown in Fig. 3.23(a), there are something needed to note. First, the pressure distribution generally increases with the distance from the leading edge, reaches a maximum value at approximately the position of x=0.76 m, and then decreases to some value before the ramp corner. Second, the result of level-2 adaptive mesh is more agreeable to experimental data of Holden and simulation of Robinson due to the reasonable mesh is obtained after refinement. But the present results are worse than Robinson's because the numbers of simulated particles and sampling are not enough. In addition, shear stress and heat transfer coefficient distributions are shown in this figure, although there is no experimental and simulated result available. This represents that there have a large improvement after mesh adaptation.

<u>Comparisons of Adaptive Mesh With or Without Cell Quality Control</u>

Mesh refinements with or without cell quality control are also discussed in this simulation again. Figures 3.24 and 3.25 are the level-2 adaptive mesh distribution and normalized density contour with or without cell quality control, respectively. From Fig. 3.24(b), there exist some cells with high aspect ratio in font of the leading edge and the region after the leading edge shock. These high aspect ratio cells are removed by cell quality control (in Fig. 3.24(c)). The local normalized density contour with a value equal to 0.66 presents more scatter distribution than the data with quality control in Fig. 3.25. Hence, a suitable adaptive mesh with good cell quality can obtain accurate simulated results.

3.3.2 Three-Dimensional Flows

Supersonic Flow over a Sphere

<u>Flow and Simulation Conditions</u>

A supersonic flow past a sphere is simulated to demonstrate the applicability of the current mesh refinement and variable time-step scheme to three-dimensional flow problem. Simulation is conducted for one to sixteenth of a sphere by taking advantage of the inherent axial symmetry of this problem. Sketch of one to sixteenth sphere is shown in Fig. 3.26. Simulation by this geometry can reduce vast particles and cell numbers and then it will be proven the domain is enough to the simulation. Considering a supersonic flow over a sphere, related flow conditions are listed as follows: Variable

Hard Sphere nitrogen gas, free-stream Mach number M_∞= 4.2, free-stream number density n_∞ = 9.77E20 particles/m^3, free-stream temperature T_∞= 66.25 K, stagnation temperature T_o = 300 K, fully thermal accommodated and diffusive sphere wall with the temperature T_w (equal to T_o). The corresponding free-stream Knudsen number Kn_∞ is 0.1035, based on the free-stream mean free path and diameter of the sphere. The diameter of sphere is 1.28 cm. These flow conditions represent the numerical simulation and experiments conducted by Russell et al [82] and list in Table 3.5. Figure 3.27 presents normalized density contour with three different cross sections, including two symmetric planes and the center plane between these two symmetric planes, to identify the computational domain is enough to this simulation. It is clear the results are almost the same.

Mesh Adaptation Concerns

The local cell Knudsen number criterion (Kn_{cc}) and free-stream parameter (ϕ_0) for mesh adaptation are chosen as 2 and 1.05, respectively, since we are dealing with an external flow. Evolution of adaptive surface mesh at each level (0-2) with cell quality control is presented in Fig. 3.28 and corresponding results are summarized in Table 3.6. The number of cells is increase from initial mesh (5,353) to the level-2 adaptive mesh (164,276) and the increased scale is larger then two-dimensional case resulting from that a parent cell can be divided to 2, 4, or 8 child cells. Hence, the number of cells and adaptation level should be careful to avoid too many cells are formed. In Figure 3.28(c) (level-2 adaptive surface mesh), it illustrates that the high-density region due to the bow shock around the sphere is clearly captured with much finer mesh distribution. In addition, the mesh distribution in the free-stream region is comparably coarse as compared with the local mean free path due to application of the free-stream parameter mentioned earlier. Application of the free-stream parameter effectively reduces the number of cells in the free-stream region, where the cell-size requirement ($\Delta x < \lambda$) could be relaxed due to nearly uniform density distribution in this region.

Comparisons of the Results Using Different Meshes

The mesh distribution and normalized density contour of initial and level-2 adaptive meshes at x-y plane are illustrated in Fig. 3.29. In these figure, the upper and lower parts are represented as the results of initial and level-2 adaptive mesh, respectively. A strong bow shock is observed in front of the sphere in Fig. 3.29(a), and the density contour distribution are obvious different due to the mesh adaptation. A maximum value of 4.26 and 5.14 are observed at the stagnation point, while a minimum value of 0.0215 and 0.016 are observed just behind the sphere with respect to initial and level-2 adaptive meshes. Comparisons of normalized temperatures at x-y plane are also presented in Fig. 3.30. The temperatures are increased when the flow interacts the bow shock and reach to a maximum at the stagnation point, and then decreased after the shock and the sphere. The distributions are quite different especially at the wake regions, which justify the use of adaptive mesh in the current study. In addition, normalized density along the stagnation line with different meshes is compared in Fig. 3.31 with experimental data by Russell [82]. Results show that the better agreement with experimental data is found using level-2 adaptive mesh near the stagnation region in front of the sphere.

Comparisons of Adaptive Mesh With or Without Cell Quality Control

Comparisons of adaptive mesh and the normalized density contour with or without cell quality control at x-y plane are presented in Fig. 3.32. The upper figure in Fig. 3.32(a) is the adaptive mesh without cell quality control. As this figure illustrated, the cells of high aspect ratio are formed especially at the interface of adaptive and un-adaptive regions due to the first type an-isotropic adaptation. The refined cells with high aspect ratio are exactly removed by the cell quality control procedure at the lower figure. Figure 3.32(b) is the comparison of the flow density at x-y plane and normalized to the free-stream density. Roughly, the results are very similar, but the refinement mesh with cell quality control has better solutions in Fig. 3.32(b). In this figure, the value of 1.05 is located on the interface of adaptive and un-adaptive region. The contour lines with cell quality control are smoother and straight. Figure 3.33 presents normalized density distribution along the stagnation streamline. Both results agree excellent with experiment.

Comparisons of the Results Using CTS and VTS Schemes

Comparisons of constant time-step scheme and variable scheme are also discussed in this simulation by using level-2 adaptive mesh (164,276). Variation of particle distribution is reduced greatly as illustrated in Fig. 3.34, which shows the comparison of distribution of averaged particles per cell on level-2 adaptive mesh using constant time-step (CTS) and variable time-step (VTS) methods. In this figure, using VTS method, the reduction of averaged number of particles per cell can be as large as 10-fold and 30-fold in the regions of oblique bow shock and wake region behind the sphere, respectively. This is achieved by keeping the particle weight the same in the reference cell (minimum cell, near the stagnation point in this case) for both VTS and CTS methods. In the other words, the statistical uncertainty in the minimum cell is kept the same for comparing both methods. The simulated particle count with different scheme is shown in Fig. 3.35. As two-dimensional simulation, the particle numbers of variable time-step scheme is obvious fewer than constant time-step scheme. Resulting simulated particles at steady state are reduced from 1.7 million, using constant time-step (CTS), to 0.34 million, using variable time-step (VTS) in this case (Fig. 3.35). In addition, another benefit of applying VTS method to unstructured adaptive mesh is that it can reduce dramatically the number of iterations of transient period towards steady state, as illustrated in Fig. 3.35. In this figure, the required number of iterations for transient period if VTS is applied is about only 25% of that if CTS is applied. In the current case, if we expect roughly the same statistical uncertainty in the minimum cell to obtain macroscopic properties for both CTS and VTS methods, the combination of VTS and unstructured adaptive mesh could save the computational time up to one order of magnitude. In this simulation, the total computation time of variable time-step and constant time-step are about 31.4 and 5.7 hours, respectively. It saves about 5.5 times computational time with the same time-steps.

<u>Density Distributions</u>

Figures 3.36 and 3.37 are the contours of normalized density and temperatures with different time schemes, respectively. The upper and lower regions are the results using CTS and VTS scheme, respectively. The trends are similar but there is a little different at the wake regions. Figure 3.38 illustrates normalized density along the stagnation line using CTS and VTS schemes. The experimental results are also included in this figure for comparison. Both of the results agree very well with the experiments. The results are almost the same although the particle number of variable time-step scheme is fewer about five times of constant time-step scheme.

3.4 Concluding Remarks

In contrast, this chapter is introduced the DSMC method combined adaptive mesh and variable time-step scheme, which is implemented in two- and three-dimension simulations. Accurate results and more efficient computation can be achieved by a uniform particle distribution. The results serve both to validate the DSMC code and to provide some insight into the complicated nature of these flows. Take the 3-D supersonic sphere flow as an example, the adaptive mesh refinement (AMR) can refine those cells near the bow shock region. The variable time-step (VTS) scheme can reduce the simulated particles from 1.7 million, which is by using constant time-step scheme (CTS), to 0.34 million and the required transient time decreases to only 25% of the constant time-step scheme. The next chapter is going to introduce the DSMC method incorporating the parallel computing to reduce the computational cost and speed up the efficiency of the computations. This exploitation will bring the DSMC method to wider applications of flow problems.

Chapter 4
Parallel Computing of DSMC

The direct simulation Monte Carlo method (DSMC) is a particle method for the simulation of gas flows. The method is statistical in nature and the gas is modeled at the microscopic level using simulated particles which each represents a large number of physical molecules or atoms. Physical events such as collisions are handled probabilistically using largely phenomenological models, which are designed to reproduce real fluid behavior when examined at the macroscopic level. It has been proven that the DSMC method is valid for rarefied gas flows even for near-continuum flows, but the computational cost is vast to afford. Thus, to parallelize the DSMC method is necessary, which is introduced in this section.

4.1 The Parallel DSMC Method

The DSMC algorithm is readily parallelized through the physical domain decomposition. The cells of the computational grid are distributed among the processors. Each processor executes the DSMC algorithm in serial for all particles and cells in its own domain. Parallel communication occurs when particles cross the domain (processor) boundaries and are then transferred between processors. High parallel performance can only be achieved if communication is minimized and the computational load is evenly distributed among processors. To minimize the communication for domain decomposition, the boundaries between sub-domains should more or less lie along the streamlines of the flow field; however, it is nearly impossible to achieve this partition for most practical flows. In practice, we can only minimize the number of edge cuts, E_c, under the framework of graph theory. Fortunately, the advancement of networking speed has reduced the communication time between processors to an acceptable level. For the DSMC algorithm, the workload (or equivalently the number of particles) in each processor changes frequently, especially during the transient period of a simulation; while the workload attains a roughly constant value during the steady-state sampling. Thus, a truly dynamic (or adaptive) domain decomposition technique is required to perfectly balance the workload among the processors.

Figure 4.1 shows a simplified flowchart of the parallel DSMC method proposed in the current study, which incorporates the multi-level graph-partitioning technique. In general, this algorithm not only works for the DSMC method, but also it is suitable for other particle-based methods, such as Molecular Dynamics (MD), Particle-In-Cell (PIC) and Monte Carlo methods in plasma physics. Note that processors are numbered from **0** to **np-1** in this figure. Before detailing the proposed procedures (Fig. 4.1), we will instead discuss the preprocessing required for this parallel implementation. In this implementation, an unstructured mesh is first constructed by a commercial code, *HyperMesh™* [83] or other equivalent meshing tool. Then, a preprocessing code is used to reorder the fully unstructured mesh data into the *globally sequential but locally unstructured* mesh data for each processor in conformation with the partitioning information from graph partitioning tool (*JOSTLE*) [54], as schematically presented in Fig. 4.2. In addition to the above, another important information output from this preprocessor is the cell-neighboring information, which is needed for particle tracing on an unstructured mesh. Original algorithm [84] used to obtain the information of cell neighbors, using the concept of loops over cells by identifying repeated node number, has been found to be very inefficient as the total number of cells increases up to several tens of thousand. Instead, we have replaced it by a very efficient algorithm, using the concept of loops over nodes by searching through the cells sharing the node, which turns out to be very efficient. For example, for preprocessing 3 million unstructured 3-D cells, it takes less than 20 minutes on a 1.6-GHz (Intel) personal computer. Preliminary results show that the preprocessing time increases approximately linearly with the number of cells. Parallel processing to speed up this preprocessing is currently in progress and will be incorporated into the parallel DSMC code in the very near future.

Note that the partition information from *JOSTLE* provides the cell numbers ($\mathbf{m_n}$ for the $\mathbf{n}$th sub-domain, where **n=0** to **np-1**) and mapping of cells in each partitioned sub-domain. After the cell-number reordering, the cells in each sub-domain are renumbered such that the corresponding global starting and ending cell numbers for the $\mathbf{n}$th sub-domain are $\sum_{i=0}^{n-1} m_i + 1$ and $\sum_{i=0}^{n} m_i$, respectively. In each processor, the cell numbering is unordered (unstructured), but both the starting (smallest) and ending (largest) cell numbers increase with processor numbers. We term this as "*globally sequential but locally unstructured*". Thus, in each processor the memory is only needed to record the starting and ending cell numbers for all processors, in addition to the cell related data in each processor. The mapping between global and local cell data, however, can be easily obtained by a simple arithmetic operation due to this special cell-numbering design. The required array size for cell related data is approximately the same as the number of cells in each sub-domain. For example, if there are one million cells totally in the simulation with 100 processors, each processor will only be required to store the array on the order of 10,000. The memory cost reduction will be approximately 100 times in this case. This simple reordering of cell numbers dramatically reduces the memory cost otherwise required for storing the mapping between the local cell number in each processor and the global cell number in the computational domain if un-reordering unstructured cells are used.

In addition, a processor neighbor-identifying array is created for each processor from the output of the preprocessor, which is used to identify the surrounding processors for those particles crossing the inter-processor boundaries (IPB) during simulation. From our practical experience, the maximum number of processor-neighbor is on the order of 10 at most; therefore, the increase of memory cost due to this processor neighbor-identifying array is negligible. The resulting *globally sequential but locally unstructured* mesh data with the partition information is then imported into the parallel DSMC code as the initial mesh distribution.

Again referring to Fig. 4.1, after reading the preprocessed cell data on a master processor (CPU **0**), the cell data are then distributed to all other processors according to the designated initial domain decomposition. All the particles in each processor then start to move as in sequential DSMC algorithm. The particle related data are sent to a buffer and are numbered sequentially when hitting the inter-processor boundary (IPB) during its journey within a simulation time-step. After all the particles in a processor are moved, the destination processor for each particle in the buffer is identified via a simple arithmetic computation, owing to the previously mentioned approach for the cell-numbering scheme, and are then packed into arrays. Considering communication efficiency, the packed arrays are sent as a whole to its surrounding processors in turn based on the tagged numbers recorded earlier. Once a processor sends out all the packed arrays, it waits to receive the packed arrays from its surrounding processors in turn. This "send" and "receive" operation serves practically as a synchronization step during each simulation time-step. Received particle data are then unpacked and each particle continues to finish its journey for the remaining time-step. The above procedures are repeated twice since there might be some particles cross the IPB twice during a simulation time-step. Theoretically it could be more than twice, but in our practical experience it is generally at most twice for "normal" domain decomposition and by carefully choosing the simulation time-step. After all particles on each processors have come to their final destinations at the end of a time-step, the program then carries out the indexing of all particles and the collisions of particles in each computational cell in each processor as usual in a sequential DSMC code. The particles in each cell are then sampled at the appropriate time.

High parallel efficiency can only be achieved if communication is minimized and the computational load is evenly distributed among processors. To minimize the communication for static domain decomposition, the boundaries between sub-domains should lie along the streamlines of the flow field [63] as mentioned previously; however, it is nearly impossible to achieve this partition for most practical flows. Fortunately, the advancement of networking speed has reduced the communication time between processors to a tolerable amount.

4.1.1 Static Domain Decomposition (SDD)

In the previous research about the parallel DSMC code, three static domain decomposition methods are used by considering the estimated particle weight on each vertex (cell center) [84]. This represents the first application of the graph partition techniques in DSMC to the best of the author's knowledge. The weight on each vertex is estimated by running the sequential DSMC code with about 10% or fewer (for some cases only 3% is used) of the total particle number, which is used for a real parallel simulation. The variations of weight on vertices obtained this way can be shown later that they are roughly equivalent to those obtained by running all particles (100%). For the simple coordinate partition, the speedup (efficiency) of 25 processors is 13.6 (54.5%), while it increases to 15.6 (62.5%) for the multi-level method (*JOSTLE*) and 15.5 (62.2%) for the multi-level scheme (*METIS*). Note that the percentage number appearing in the parenthesis right after the speedup represents the corresponding parallel efficiency.

Although the DSMC possesses nearly 100% parallelism (except for initialization and final output), and an estimated particle distribution is used as a weight to get a suitable partition domain, both the values of speedup and efficiency by using static domain decomposition are expected to be lower than the ideal values due to the load unbalancing and communication as mentioned previously. Dynamic domain decomposition can improve the parallel performance by balancing the load of each processor and are described in the next section.

4.1.2 Dynamic Domain Decomposition (DDD)

This section presents an overview of the algorithms implemented of dynamic load decomposition scheme. As mentioned above, the parallel performance will become worse resulting from the communication and the load unbalancing. Dynamic domain decomposition scheme for an unstructured mesh is implemented in this thesis to speed up the parallel computing. It is a non-trivial task and it is the main contribution of the PhD research. Basic concept is the domain will be repartition when the loading of each processor is becoming unbalancing. The simulator aims to balance the number of particles on the sub-domains. The flowchart with dynamic domain decomposition is shown as Fig. 4.3. The procedures of the flowchart are almost the same except some processes of dynamic domain decomposition method. There are three main processes, which are decision policy for repartitioning, repartition the domain and cell/particle migration, for dynamic domain decomposition. The details are described in the following.

4.1.2.1 Decision Policy for Repartitioning

DSMC represents a typical dynamic (or adaptive) irregular problem, i.e., workload distributions are known only at runtime, and can change dramatically as simulation proceeds, leading to a high degree of load imbalance among the processors. Thus, some decision policy is required to determine when to repartition the computational domain, since the repartition is often expensive computationally. It has been shown that, for some problems using DSMC, remapping the domain at fixed intervals leads to poor parallel performance [7, 60]. Therefore, it is highly desirable to either pre-determine the optimal interval for repartitioning, or using a clever monitoring policy to decide when to repartition. The former choice is definitely impractical since pre-runtime analysis is generally required to determine this optimal choice. Therefore, in the current study, a decision policy, Stop At Rise (SAR) [66], is employed to determine when to repartition the domain. SAR, a “greedy” repartitioning policy, attempts to minimize the long-term processor idle time and tries to solve the load imbalance in advance before it becomes a problem since the last repartitioning. SAR assumes the processors synchronize at every time-step. And then the idle time of each processor is known. The quality

$$\frac{1}{n}\sum_{j=1}^{n}(T_{\max}(j)-\overline{T}(j)) \qquad (4.1)$$

represents the average idle time per processor per time-step since the last re-balance. This decision policy chooses to repartition the computational domain based on the value of a degradation function $W(n)$ at the n^{th} time-step, which is defined as follows:

$$W(n) = \frac{\sum_{j=1}^{n} (T_{\max}(j) - \overline{T}(j)) + C}{n} \tag{4.2}$$

where $T_{\max}(j)$ is the maximum amount of time required by any processor to complete the j^{th} time-step, $T_{\text{avg}}(j)$ is the average time required by a processor to complete the j^{th} time-step, and C is the amount of time required to complete the repartitioning operation. This degradation function represents the average idle time for each processor including the cost of repartition. In general, $W(n)$ tends to decrease with the increasing value of n. The summation term in Eq. (4.2) will eventually increase as the workload unbalance develops, while the repartitioning cost, C, is approximately constant during simulation. Repartitioning is not performed until the time that $W(n)>W(n\text{-}1)$, i.e., when the first local minimum of degradation function is detected. This decision policy for repartitioning the domain is inherently advantageous over the fixed-interval scheme in that no prior knowledge of the evolution of the problem is necessary for the determination of the repartitioning interval, and the repartitioning can be expected to follow the dynamics of the problem without wasting computing resources. A detail analysis of the behavior of degradation function is proposed by Nicol and Saltz [66].

4.1.2.2 Repartition the Domain

In the current study, we have incorporated the parallel runtime library, *PJOSTLE* [56], as the repartitioning module in our parallel DSMC code. *JOSTLE* [54], a serial version of *PJOSTLE,* uses the multi-level implementations that match and combine pairs of adjacent vertices to define a new graph and recursively iterate this procedure until the graph size falls under some threshold. The coarsest graph is then partitioned and the partition is successively refined on all the graphs starting with the coarsest and ending with the original. At evolution of levels, the final partition of the coarser graph is used to give the initial partition for the next finer level. *PJOSTLE* [56], a parallel version of *JOSTLE* [54], uses an iterative optimization technique known as relative gain optimization, which both balances the workload and attempts to minimize the inter-processor communication overhead. This parallel algorithm runs on single program multiple data (*SPMD*) paradigm with message passing in the expectation that the underlying unstructured mesh will do the same. Each processor is assigned to a sub-domain and stores a double-linked list of the vertices (cell centers in DSMC) within that sub-domain. However, each processor also maintains a “halo” of neighboring vertices in other sub-domains. For the serial version, the migration of vertices simply involves transferring data from one linked-list to another. In parallel implementation, this process is far more complicated than just migrating vertices. The newly created halo vertices must be packed into messages as well, sent off to the destination processors, unpacked, and the pointer based data structure recreated there. This provides an extremely fast solution to the problem of dynamically load-balancing unstructured mesh [53].

In DSMC simulation, the workload of each processor is approximately proportional to the number of particles in the corresponding sub-domain. Thus, we can assign the weight of each vertex in graph as particle numbers in the corresponding cell in estimating the workload during simulation. *PJSOTLE* [56] will try to maintain perfect load balance while optimizing the partitions based on pre-determined balance factor. This factor, which affects the partitioning quality and cost, is defined by $B=S_{max}/S_{opt}$, where S_{max} is the largest allowable weight of the sub-domains and S_{opt} is the optimum sub-domain size which equal to the average weight of these sub-domains. B is 1.03 in the current study, unless otherwise specified. Simulated results have shown a fairly even particle distribution among processors is obtained using the above setting, which can be seen later.

4.1.2.3 Cell/Particle Migration

After repartitioning the domain, relationship between cells and sub-domains has to be updated according to the new partition. Any cell may be assigned to a processor, which is different from the original processor it belongs to. Thus, cell/particle associated data need to migrate to their new parental processor properly. Theoretically, the multi-level graph-partitioning scheme is much faster than the hybrid graph-geometric partitioning scheme developed by Robinson [24-26], in which only the "halo" cells of each sub-domain are allowed to move among processors after each repartition.

In addition, the original cell neighbor-identifying array, *nbr*(*face_number*, *local_cell_number*)=*local_cell_number*, in a sequential code has been changed to *nbr*(*face_number*, *local_cell_number*)=*global_cell_number* in the parallel code. Thus, a conversion array between local and global cell numbers is required to access the data efficiently. Note that the global cell numbers associated with each cell is not changed throughout the simulation. Only the local cell numbers for each cell in each processor has to be updated according to the new partition. Of course, the conversion array between local and global cell numbers has to be changed accordingly for those cells involved in migrating among processors. Thus, the update of neighbor-identifying array after cell data transferred between processors becomes very easy. Only the local cell numbers for the transferred cells have to be changed with negligible computational cost.

The detail procedures of cell/particle migration after the repartition is shown in Fig. 4.4 and briefly summarized as follows;

1. Pack into buffer arrays the to-be-transferred particle related data particle by particle. They include positions, velocities, internal energies and the new local cell numbers in the destination processor, to which the particle shall reside. The new local cell numbers is assigned as the value, which is the sum of one and the most updated pre-partitioned maximum local cell numbers in pre-partitioned destination processor. Having packed the to-be-transferred particle data, they are then removed from the source processor they belong to.
2. Pack into buffer arrays the to-be-transferred cell related data cell by cell. The procedures for each cell are described in detail as follows. *First*, record the data of the to-be-transferred cell, including new local cell number in the destination processor (as in step 1), cell/node coordinates and sampled data in the cell and related cell face. *Second*, update the relation between the local and global numbers, i.e., the data of the to-be-transferred cell numbers are then replaced by the data of the maximum cell numbers in the source processor. Then, subtract one from the maximum local cell numbers in the source processor.
3. Migrate both the particle- and cell-data in the buffer arrays as a whole to the destination processors.
4. Receive and unpack these data from the buffer.
5. Reorder and change the neighboring cell numbers accordingly.

After all the subroutines of dynamic domain decomposition processed, the general parallel DSMC method is applied immediately and then the next repartition is called if the loading becomes unbalance again.

The current parallel code, in *SPMD* (Single Program Multiple Data) paradigm, is implemented on the IBM-SP2 machines (distributed memory system) using message passing interface (MPI) to communicate information between processors. It is thus essentially no code modification required to adapt to other parallel machines (e.g. IBM-SP2, IBM- SMP, PC-clusters) with similar distributed memory system once they use the same MPI libraries for data communication.

In the next section, we are going to compare the performance of the static and dynamic domain decomposition method for simulating the driven cavity flow. In the current study, we use the state-of-the-art graph partitioning tools, *JOSTLE* [54], to the static domain decomposition, and use the parallel graph partitioning tool, *PJOSTLE* [56], to the dynamic domain decomposition. Although the current repartitioning tool we use is ParMetis [57].

4.2 Parallel Performance of the Parallel DSMC Method

In order to test and study the parallel performance of the current parallel implementation of DSMC using dynamic domain decomposition, we have used a two-dimensional, high-speed, bottom lid-driven cavity flow at different problem sizes as the test problem, similar to that used in Robinson [24]. We have tested the implementation on different parallel machines, including IBM-SP2, IBM-SMP and PC cluster system, which are all memory-distributed machines. In this thesis, we will only describe the results on two commercial parallel machines at National Center for High-performance Computing (NCHC) in Taiwan. The simulations are operated on IBM-SP2 and IBM-SMP parallel machines up to 64 and 128 processor numbers, respectively. The detail configures of each system are shown in Table 4.1. In what follows, the test flow conditions, preliminary simulated results of the lid-driven cavity flow, dynamic domain decomposition, parallel performance and time breakdown analysis of the parallel DSMC code will be reported in turn.

4.2.1 Flow Conditions of Driven Cavity Flow

In order to test the performance of the proposed parallel DSMC algorithm, we use a 2-D high-speed driven cavity flow with bottom plate moving to the right with the speed of eight times the most probable speed as similar to that used in Robinson [24]. Figure 4.5 illustrates the sketch of the driven cavity flow. Related flow conditions include argon gas, the wall temperatures are all set to be 300K and diffusive (100% full thermal accommodation), and lower wall velocity is eight times the most probable speed based on wall temperature. Initial gas temperatures within the enclosure are set to be 300K. The corresponding Knudsen number, based on the initial uniform mean free path and length of the square box, is equal to 0.04 cavity flow as the benchmark problem [24]. The No Time Counter (NTC) method and Variable Hard Sphere (VHS) molecular model [10, 11] are used for collision kinetics and reproduction of real fluid properties, respectively. Data in the cell are sampled every two time-steps in the current study, unless otherwise specified. Different problem sizes, including small, medium and large problem size, are considered for simulation (Table 4.2), where the number of particles and the number of the cells are in the range of 225,000~3,600,000 and 11,250~180,000, respectively. Average number of particles per cell is kept approximately at 20 particles for three test problem sizes. Nearly uniform triangular mesh is used throughout the study. Simulations are run for 50,000 time-steps with time-step about 1/2 of the initial mean collision time-step. DSMC code is implemented on IBM-SP2 and IBM-SMP with processor numbers in the range of 1~64 and 1~128, respectively.

Preliminary Simulated results

An ultra high-density region appears at the very right-hand bottom corner due to the high-speed moving plate at the bottom of the cavity. Also the densities at the two top corners are higher than the initial value. In addition, most of the region above the moving plate is rarefied as compared with the initial value since the particles are "entrained" to collide with the moving plate. It is a good benchmark to test the parallel performance because it will lead to large load unbalancing.

Parallel Performance

Static and dynamic domain decomposition methods are both applied in this simulation to investigate the parallel performance of our PDSC. *JOSTLE* [54] partition library is used to provide the initial decomposition by assigning constant particle weight on each vertex (cell center). In other words, the number of cells of each sub-domain is approximately the same initially within the predetermined balance factor, which is 1.03 in the current study, unless otherwise specified. For dynamic domain decomposition, *PJOSTLE* runtime library [57] is incorporated in the code to repartition the domain based on SAR policy as the DSMC simulation proceeds. In this study, different strategies of activating SAR policy at intervals of 2Δt, 10Δt and 20Δt are implemented and compared, which will be shown later.

Two parameters often used to measure the performance of the parallel implementation are *speedup* and *efficiency*. These two parameters are defined as follows;

Speedup is defined as the ratio of the required running time for a particular application using *one* processor to that using *N* processors, i.e.,

$$Speedup \equiv \frac{t_1}{t_N} \tag{4.3}$$

Efficiency is then defined as

$$Efficiency \equiv \frac{Speedup}{N} \tag{4.4}$$

and is just the ratio of the true speedup to the ideal speedup, N, and hence its value lies between zero and one. The speedup and efficiency of ideal parallel algorithm are the number of the processors and unity, respectively.

4.2.2 Simulations on IBM-SP2

Speedup and Efficiency

Results of parallel speedup and efficiency of the cavity-flow computation, at different problem sizes on IBM-SP2 machine, as a function of the number of processors are presented in Fig. 4.6. In these figures, there are four curves with circle, triangle, quadrilateral and cross symbols with respect to SDD, DDD by calling SAR at intervals of 2Δt, 10Δt and 20Δt. And the linear dash line represents the ideal case. As expected, the parallel performance of those using dynamic domain decomposition is much better than those using static dynamic domain decomposition. Several trends for different problem sizes are described in detail as follows.

Small Problem Size

Super-linear speedup (efficiency > 100%) occurs clearly for number of processors less than or equal to 16, if dynamic domain decomposition is applied (Fig. 4.6(a) and Fig. 4.6(d)). This is mainly attributed to both the cache effects and better load balancing among processors. However, the efficiency decreases with increasing number of processors (up to 64) as expected, due to load unbalancing among processors, if static domain decomposition is applied. Figure 4.7 shows that the computational time per particle as a function of particle numbers on a single processor (IBM-SP2), in which the minimum computational time per particle occurs at approximately 4,000 particles and increases with increasing particle numbers as particle numbers is greater than 4,000. As the number of processors increases over 16, negative effects of load unbalancing and communication increase among processors begin to play a more important role than the positive effects of cache effects. Thus, the parallel efficiency decreases monotonously with increasing number of processors (up to 64) even if dynamic domain decomposition is used, as shown in Fig. 4.6(a) and Fig. 4.6(d).

In addition, results show that parallel performance for applying SAR scheme less frequently is generally better than applying SAR scheme more frequently for number of processors less than 64. As the number of processors increases up to 64, all three strategies of applying SAR scheme result in roughly the same value of parallel efficiency, mainly due to the relative load unbalancing developed among processors levels the advantages gained by repartitioning the domain less frequently. Also, in the small problem size, it might be difficult for the repartitioning library to re-decompose the domain more exactly since too few particles stay within a processor if the number of processors is high, e.g., only approximately 3,500 particles per processor with 64 processors. Nevertheless, for the small problem size the parallel efficiency using dynamic domain decomposition improves appreciably in the range of 30-50%, as compared with static domain decomposition.

Medium Problem Size

Similarly, super-linear speedup exists for the medium problem extending even up to 48 processors, if dynamic domain decomposition is activated (Fig. 4.6(b) and Fig. 4.6(e)). This unusual extended super-linear speedup should be attributed to the relatively small cache size available on the super-scalar workstation, in which the array data can be accessed very fast. However, the super-linear speedup is not seen at all if the dynamic domain decomposition is deactivated, which nevertheless demonstrates the effectiveness of implementing dynamic domain decomposition. As the number of processors is over 48, this super-linear speedup disappears due to increasing communication among processors. For the medium problem size, parallel performance (Fig. 4.6(b) and Fig. 4.6(e)) using dynamic domain decomposition is generally 50-100% higher than that using static domain decomposition as the number of processors is less than or equal to 64. Note that approximately 90% of parallel efficiency can be reached at processor numbers of 64 for the medium problem size. In addition, advantage of activating SAR scheme less frequently is diminishing for the medium problem size due to the increasing problem size, in which the repartitioning cost becomes comparatively less important than useful particle computation.

Large Problem Size

For the large problem size, super-linear speedup generally disappears even if the dynamic domain decomposition is activated (Fig. 4.6(c) and Fig. 4.6(f)). The only exception is the case, which activates SAR scheme more frequently (at interval of 2 Δt), which balances the workload among processors more efficiently than the other two cases. Parallel efficiency of 107% can still be reached for number of processors of 64 when activating SAR scheme at interval of 2 Δt.

In conclusion, the results are shown that the simulation with larger problem size can obtain higher parallel performance. The simulations by using dynamic domain decomposition method present a better performance then using static domain decomposition and dependent to the problem size. And the optimal frequency of activating SAR scheme generally increases with increasing problem size.

Dynamic Domain Decomposition

Typical evolution of dynamic domain decomposition using graph partitioning technique are shown in Fig. 4.8 and Fig. 4.9 for the large problem size using 16 and 64 processors, respectively, when activating SAR scheme at intervals of 2Δt. As mention above, *JOSTLE* is used to form the initial partition by assigning the unitary weight on each vertex, and *PJOSTLE* is used to repartition when the loading is unbalance. It is clear that region covered by each sub-domain (processor) changes as the simulation proceeds due to repartitioning among processors when the initial size of each domain is approximately the same. There exists a smallest sub-domain in the right-hand lower corner of the cavity due to the presence of highest density in this region (Figs. 4.8(c) and 4.9(c)). In addition, the size of the sub-domains above the moving plate is generally larger as compared with others due to the rarefied conditions caused by the fast moving plate (Figs. 4.8(c) and 4.9(c)). It clearly demonstrates that the current implementation of dynamic domain decomposition is very effective in following the dynamics of the flow problem under study.

Figure 4.10 illustrates both the number of particles in each processor and the partition count as a function of the number of simulation time-steps for the large problem size using 16 processors when activating SAR at intervals of 2Δt. It only shows the time history of both quantities in the early stage of simulation in each processor for the clarity of presentation. In this figure, we are not trying to identify the evolution of particle numbers in any specific processor, although different lines represent different processors. Results show that number of particles in each processor approaches to the average number of particles per processor (225,000) right after the repartition, which shows the load balancing takes effect once the repartitioning functions. Note that we have preset the balance tolerance value to be 3% in *PJOSTLE* library [57], which represents that load imbalance among processors less than this value, shall not repartition the domain, even the SAR scheme decides to do so. In addition, the deviation of the number of particles in some processors from the average value deteriorates faster in the early stage right after the repartition, where the flow changes dramatically from initially uniform distribution, as shown in Fig. 4.10. As the flow reaches steady state, the repartitioning is less frequently as expected, which can be seen clearly with smaller value of slope in Figs. 4.11 and 4.12. These figures show the typical domain repartition counts as a function of simulation time-steps for 16 and 64 processors. Each solid symbol indicates a repartitioning of the computational domain. Generally, at first it increases more rapidly (larger value of slope) with simulation time during transient period and then increases less rapidly (smaller value of slope) with simulation time as flow approaches steady state (~10,000 steps in Fig. 4.12). Similar trends can be found for other flow conditions. Note that the repartition history for larger problem size after 20,000 steps is skipped due to the limitation of accessing time to the parallel machine. An interesting feature of small problem is the remapping process are called at every opportunity after steady state, that is, indicating the flow field is unable to balance the load. The instability named "flip-flop" will occurred when the number of processors is increased or the problem size is too small. When the weight of the transferred vertices would reach such a level that their transfer would take a sub-domain from an over-loaded state to an under-loaded state, or vice-versa. In such condition, the repartition is repeating. The problem dominates on 64 processors even for the simulation with medium problem size in Fig. 4.12.

Figures 4.13 and 4.14 shows the final normalized workload distribution (or number of particles per processor) with respect to processor numbers among the 16 and 64 processors for the three different problem sizes using different strategies of implementing SAR (2Δt, 10Δt and 20Δt), respectively. It is clear that the workload distribution is much more uniform with dynamic domain decomposition than that without dynamic domain decomposition. In addition, the workload distribution is found to be most uniform for SAR-2Δt scheme since it monitors the load imbalance more often than others. This does not guarantee better parallel efficiency (referring to Fig. 4.6(d) and Fig. 4.6(e)), since frequent repartition is expensive as compared with the normal DSMC computation.

Time Breakdown of Parallel Implementation

Figure 4.15 illustrates the typical fraction of time spending in DSMC computation and dynamic domain decomposition per simulation time-step as a function of the number of processors by employing SAR scheme at the interval of 2Δt. Note that the DSMC computational time includes the "useful" DSMC computational time, the idle time and the communicational time during particle movement between adjacent processors. It can be seen that, for the small problem, the average fraction of time spending in repartitioning the domain per time-step increases dramatically with the number of processors, which explains the rapid decrease of parallel efficiency at this condition for employing SAR every $2\Delta t$ in Fig. 4.6(d). More or less, similar trend is found for medium problem size. On the contrast, for the large problem, the fraction of time for repartitioning the domain remains approximately the same (~0.04) with increasing number of processors up to 64. Correspondingly, the fraction of time for the DSMC computation varies insignificantly as the number of processors is over 24. This shows the current parallel DSMC method may be highly scalable at least for the large problem size.

Figure 4.16 is the real CPU running time in seconds required to complete one time-step of "useful" DSMC and "repartition" with different problem size and the number of processors. The real time for repartition is derived by the number of repartitions throughout the computation. We can see some information from this figure. First, the time spending on calling once "useful" DSMC process increases as the increasing problem size and decreasing the number of processors. Second, the time of per repartition of each problem size is reduced with the increasing of processor number. In general, the repartition time is proportion to the problem size, especially for the small number of processors. That is because the loading of large problem size is too large and the repartition process has to take a relative long time to deal with. But this phenomenon is disappearing as the increasing of the processor number. Finally, the cost of repartition is very small and can be neglected by comparing with "useful" DSMC.

The relative cost of each step of the parallel DSMC program with different problem size on IBM SP2 is shown in Fig. 4.17. The data with dynamic domain decomposition are also included for comparing. The solid and dash lines represent the data with static and dynamic domain decomposition, respectively. The cost of message passing is included in *MOVE* subroutine, and it cost about 60~70% of the total time. It increases as increasing the number of processors. The trend displayed with different problem size is qualitatively the same.

<u>Degree of Imbalance</u>

Degree of imbalance is a useful indicator for measuring the workload non-uniformity among processors, which is an important parameter for justifying dynamic domain decomposition in the current study. It is interesting to examine the maximal degree of imbalance in the system, I_{max}, which is defined as

$$I_{\max} = \frac{1}{\overline{W}}(W_{\max} - W_{\min}) \tag{4.5}$$

where W_{max} and W_{min} are the maximum and minimum particle numbers across the processor, respectively. $\overline{W}$ is the average particle numbers of each sub-domain. Figure 4.18 shows the variation of maximal imbalance with processor numbers for three different problem sizes. The solid and dash line represent the parallel simulation with static and dynamic domain decomposition, respectively. In general, the maximal imbalance developed by static domain decomposition (0.5-2) is much higher (2-6 times) than that by dynamic domain decomposition (0.1-0.5). In addition, workload imbalance developed very fast among processors as the number of processors increases, if dynamic domain decomposition is not used. Although the maximal degree of imbalance for the small problem size deteriorates with increasing number of processors, it is fairly constant within 0.4 for the medium and large problem sizes, which again demonstrates the effectiveness of the dynamic domain decomposition.

4.2.3 Simulations on IBM-SMP

Figure 4.19 presents the parallel performance by using static and dynamic domain decomposition on IBM-SMP with respect to small, medium and large problem size, respectively. The number of processors is up to 128. The trend are very similar to the simulation on IBM-SP2 and do not repeat here again.

In conclusion, a proposed DSMC code with parallel, dynamic domain decomposition method is implemented here to speed up the computing efficiency and presented by solving a two-dimensional driven cavity flow. The results show the present researches are very well and powerful, and the present method has potential to simulate more realistic physical problems.

4.3 Verifications of the Parallel DSMC

The simulated cases in Chapter 3, that is, a two-dimensional hypersonic flow past a cylinder, a two-dimensional hypersonic flow past a 15°-compression ramp, a three-dimensional supersonic flow past sphere, are again used here to compare the results using statistic and dynamic domain decomposition methods. Results are then compared with experimental data and previous simulation wherever available, while the description of flow physics of the test problems will be as brief as possible since it only serves to verify the applicability and its accuracy of the proposed method. Flow conditions and results for each case are described in the following in turn.

4.3.1 Two-Dimensional Flows

Hypersonic Flow over a Cylinder

Flow and Simulation Conditions

Flow conditions of this case are shown in Section 3.3.1. The adaptive mesh is used here to improve the accuracy of the solutions. The flow domain is divided to 64 sub-domains for the parallel computing and run on IBM-SP2 machine at NCHC in Taiwan.

Domain Decomposition

Figure 4.20 shows the partition history of the dynamic domain decomposition for this simulation. Large variation of sub-domain area in the initial domain decomposition results from the use of a solution-based adaptive mesh, which is obtained from a mesh adaptation module based on a preliminary parallel simulation. Constant time-step scheme is used in this simulation. For the initial domain decomposition, we have assigned the uniform weight of each cell to form the initial domain decomposition. The cell number in each sub-domain is approximately the same initially, although the size of each sub-domain is highly different. This is because the cells in front of the cylinder are smaller than those in the wake regions of the adaptive mesh. Figure 4.20(c) shows that the final decomposition, which adapts to the flow dynamics as simulation continues, is totally different from the initial decomposition. Although the interfaces are not smooth and there are several cells which are isolated from their parent sub-domain. These fragments can lead to more interfaces between sub-domains and increasing the cut edges. But the algorithm is still working for our code. The number of simulated particles is about 1.3 million and the number of sampling is 10,000 in this simulation.

Comparison of Properties Contours

Figures 4.21 and 4.22 illustrate the contours of normalized density and temperatures with static and dynamic domain decomposition methods. From these figures, the simulated results using static or dynamic domain decomposition are almost the same. It proves that the dynamic domain decomposition method can provide accurate results and reduces lots of computational time.

Centerline Properties Distribution

Figure 4.23 illustrates the computed centerline densities and temperatures (rotational and translational) using dynamic domain decomposition, respectively, along with the simulation data without dynamic domain decomposition and previous experimental data [79]. Density increases rapidly along the centerline in front of the cylinder and becomes relatively small in the wake region. Temperature also increases rapidly along the centerline but decreases rapidly after the bow shock. Strong non-equilibrium between rotational and translational temperatures is found after the cylinder due to the highly rarefied conditions in the wake region. Nevertheless, agreement between the current simulation with static/dynamic domain decomposition and experimental data wherever available is excellent, considering the experimental uncertainties. In addition, the running times of static and dynamic domain decomposition are about 6,400 and 2,700 seconds. Thus, simulation using dynamic domain decomposition reduces the running time up to 60% in this case, as compared with that using static domain decomposition.

Hypersonic Flow Over a 15^o-Compression Ramp

Flow and Simulation Conditions

A flow over a 15^o-compression ramp is also proposed as the second benchmark here to identify the validity of dynamic domain decomposition method. The problem and flow conditions are described in Section 3.3.1. The flow domain is also divided to 64 sub-domains and run on IBM-SP2 machine at NCHC.

Domain Decomposition

Figure 4.24 shows the initial and final domain decomposition for this simulation. Similar to previous case, constant time-step scheme is used in this simulation. The initial computational domain is also partitioned by uniform weight. Thus, each sub-domain has an approximately cell number. Some large domains above the flat plate are found due to the rarefied conditions in the region (Fig. 4.24(a)). In addition, some small domains at the end of the ramp plate and shock regions are found, where the density is very high due to the oblique shock originating approximately from the ramp corner. The simulated particles are about two million and the number of sampling is 10,000.

Comparison of Properties Contours

Figures 4.25 and 4.26 illustrate the contours of normalized density and temperatures with static and dynamic domain decomposition methods. Again, the results by using dynamic domain decomposition scheme agree excellent with those with static domain decomposition scheme. Thus, the implemented dynamic domain decomposition method is a very powerful tool for simulations.

Centerline Properties Distribution

Figure 4.27 presents the pressure, shear stress and heat transfer coefficients along the solid wall using static and dynamic domain decomposition methods. Data include computational results without dynamic domain decomposition, previous DSMC results by Robinson [24] and experimental data by Holden [81]. Results show that pressure coefficient increases rapidly near the tip, declines slowly along the flat plate to a lowest value near the ramp corner and finally increases dramatically along the ramp wall. Results generally agree with previous simulation and experimental data, although the current simulated data seems to agree favorably with experimental data along the ramp wall, as compared with those of Robinson's [24]. Final rapid decline of our simulated data should be attributed to the vacuum condition we have imposed at the outflow boundary. In addition, the results of along the solid wall that are obtained with and without dynamic domain decomposition coincides excellently with each other shear stress coefficient and heat transfer coefficient although there is no experimental or simulated results. The running time of static and dynamic domain decomposition are about 14,000 and 7,500 seconds. The simulation time required for dynamic domain decomposition is about 100% shorter than that required for static dynamic domain decomposition, which again justifies the current parallel implementation.

4.3.2 Three-Dimensional Flows

Supersonic Flow over a Sphere

Flow and Simulation Conditions

For a three-dimensional simulation, a supersonic flow over a sphere is proposed here again to recognize the validity of dynamic domain decomposition method. The reasons to choose this as the test problems are, first, there exist experimental data and, second, it is a good test for checking if the simulation can reproduce the flow symmetry. Third, it can prove that the dynamic domain decomposition method can be easily extended to three-dimensional flow. Specific details describing the flow conditions are presented in Section 3.3.2. The flow domain is divided to 8 sub-domains and run on IBM-SP2 machine at NCHC. The number of simulated particles is about 1.7 million at steady state and the number of cells is approximately 164,000 after two levels of mesh refinement. 10,000 sampling are used to sample for obtaining averaged flow properties.

Domain Decomposition

Figure 4.28 illustrates the initial and final domain decomposition for the supersonic flow past a sphere on a reduced (1/16) computational domain surface. The initial domain decomposition (Fig. 4.28(a)) is obtained assigning equal weight to each cell. At the final domain decomposition (Fig. 4.28(b)), the sub-main size in front of the sphere enlarges as compared with the initial sub-domain size, due to the application of variable time-step method and increased density in the stagnation and bow shock region. In this case, the running times of static and dynamic domain decomposition are about 15,000 and 11,100 seconds, respectively. The computational time is saved up to 35% using dynamic domain decomposition, as compared with that using static domain decomposition. We would expect much higher time saving of more processors are used. The number of simulated particles is about 1.7 million and the number of sampling is 10,000.

<u>Comparison of Properties Contours</u>

Figures 4.29 and 4.30 illustrate the contours of the normalized density and temperatures on x-y plane with static and dynamic domain decomposition methods. In these figures, the results by using dynamic domain decomposition show excellent agreement with those by using static domain decomposition.

<u>Centerline Properties Distribution</u>

Normalized density along the stagnation line is presented in Fig.4.31. The previous experimental data by Russell [82] is included for comparisons. All the results using static and dynamic domain decomposition methods agree well with the experimental data [82].

4.4 Concluding Remarks

This chapter has presented a description of the dynamic domain decomposition method for the unstructured DSMC code. The aim of successful dynamically repartition the computational domain is implemented on parallel machines, including IBM series and PC cluster. The parallel performance of the dynamic domain decomposition (DDD) method is consistently and significantly superior to the static domain decomposition (SDD) method. The computational time of the 3-D supersonic sphere flow reduces about 35% by using dynamic domain decomposition method because more balancing load is achieved. Briefly speaking, the DSMC method with dynamic domain decomposition provides an efficient parallel computing and the results are almost the same as those with static domain decomposition. And the algorithm of this method can easily be applied to other particle methods without many modifications.

Chapter 5

Conservative Weighting Scheme

For traditional DSMC code, the weighting of each species is the same (constant weighting). That means each simulated particle represents the same number of real physical particles. It works perfectly when the quantities of components have the same order. But in many applications the species has very small quantity and it is very important. For example, the product of chemical vapor deposition (CVD) is an important factor for film deposition. In order to obtain enough sampling of the trace species, the number of non-trace particles will become very huge by using the constant weighting scheme. This will lead to an enormous computational cost. Thus, a species-dependent weighting scheme is necessary when the flow has trace species, which is introduced in this chapter.

5.1 Conservative Weighting Scheme (CWS)

To overcome the trace problem mentioned in the above, recently Boyd [68] proposed a conservative weighting scheme, which is described briefly in the following. In this method, each species has its weighting. Non-trace and trace species have larger (W_1) and smaller (W_2) weights ($W_2/W_1<1$), respectively. The first stage of the conservative weighting scheme is to split the particle of abundant species (W_1) into a particle with weight W_2 (trace species) and a particle with weight of W_1-W_2 when two particles (trace and abundant species) collide. Then, a collision is then performed using the conventional DSMC procedure for the two particles that have the same weight W_2. The final stage is to merge together the two particles that were split such that the each linear momentum in three physical directions is exactly conserved. The momentum conservative equation is shown as Eq. (5.1);

$$\begin{aligned}P_x &= W_1 m_1 u_1 + W_2 m_2 u_2 = W_1(m_1 u_1 + \phi m_2 u_2), \quad \phi = W_2/W_1 \\ &= W_1[(1-\phi)m_1 u_1 + \phi(m_1 u_1 + m_2 u_2)]\end{aligned} \tag{5.1}$$

u_1, u_2, m_1 and m_2 are the pre-collision velocities and the molecular mass of the collision partners, respectively. Equation (5.1) means the non-trace particle (W_1) is split into two parts imaginatively; one particle has weighting W_2 and the other has weighting W_1-W_2. The first part will has elastic collision with the trace particle, but the second part remains the same situation. Thus, Eq. (5.1) can be rewritten as Eq. (5.2)

$$P_x = W_1[(1-\phi)m_1u_1 + \phi(m_1u_1 + m_2u_2)]$$
$$= W_1[(1-\phi)m_1u_1 + \phi(m_1u_1^{'} + m_2u_2^{'})] \tag{5.2}$$

$u_1^{'}$ and $u_2^{'}$ are the post-collision velocities of the first part of non-trace particle and the trace particle, respectively. Then the two parts of non-trace particle are merged as one particle

$$P_x = W_1[m_1u_1" + \phi m_2u_2^{'}] = W_1m_1u_1^{''} + W_2m_2u_2^{'}, \quad u_1^{''} = (1-\phi)u_1 + \phi u_1^{'} \tag{5.3}$$

$u_1^{''}$ and $u_2^{'}$ are the real final velocities of the non-trace and trace particles, respectively. The momentum is conserved by Eq.(5.1)~Eq.(5.3).

Unfortunately, it does not explicitly conserve total energy. But the energy difference (loss) caused by this split-merge process is found to be proportional to the weight ratio W$_2$/W$_1$ (<1). The energy lost for each collision is calculated by Eq. (5.4),

$$\Delta E = E^{'} - E = W_1\frac{1}{2}m_1[(u_1"-u_1)^2 + (v_1"-v_1)^2 + (w_1"-w_1)^2]$$
$$+W_2\frac{1}{2}m_2[(u_1"-u_1)^2 + (v_1"-v_1)^2 + (w_1"-w_1)^2]$$
$$= W_1\frac{1}{2}m_1\phi(1-\phi)[(u_1-u_1')^2 + (v_1-v_1')^2 + (w_1-w_1')^2] \tag{5.4}$$

Thus, the conservative weighting scheme proposed by Boyd [68] nearly conserves total energy as this weight ratio is much smaller than unity. The split-merge process described in the above can be summarized as Figure 5.1.

It was argued that if the split-merge scheme is employed at each collision, then energy is continuously lost from the system because of energy loss [68]. Boyd also proposed some practical remedy to keep this energy loss to a minimum by adding lost energy to the central-mass energy in a subsequent collision. In general, this energy should only be added to collisions between particles both having the maximum weight used in the simulation to keep this effect a minimum; that is, between two non-trace particles. Thus, energy conservation is essentially maintained for each iterative step of the simulation.

5.2 Verifications of Conservative Weighting Scheme

In the following we are going to perform two simulations to verify the conservative weighting scheme. The first test case is the single-cell simulation using both conventional (constant weighting) and conservative weighting schemes without chemical reaction. By comparing the velocity distribution and relative error, it can be clearly shown that CWS is much superior to the constant weighting scheme if trace species is involved. The second test case is a hypersonic flow over a 3-D cylinder. From this case we can see the CWS can get the same simulated results by using much less particles and computational time.

5.2.1 Maxwell-Boltzmann Distribution of A Single Cell

Two-Component Mixture without Chemical Reaction

The first simulation includes two species: Ar and He with different weight ratios. In this simulation, we would like to know the effectiveness and accuracy of the conservative weighting scheme. We perform the simulation with different weight ratios using both constant weighting scheme and conservative weighting scheme (CWS). The total simulated particles are 10,000 in the cell, and the weight ratios are W_{Ar}/W_{He}=0.1 (10%), 0.05 (5%) and 0.01 (1%), respectively. Note that the number in the parenthesis represents mole fraction of Ar. Thus, Ar is a potential trace species in this case. The temperature in the cell is preset as 1,000 K. Figures 5.2~5.4, with weight ratios of 1/9, 1/19 and 1/99, respectively, show the velocity distribution after 20,000 time-steps and the relative error as a function of simulation time. Note that the relative error is defined as the root mean square of the sum of deviation from the M-B distribution relative to the Maxwell-Boltzmann distributions by dividing the velocity range (-5,000~5,000 m/s) into 50 sections. In these figures, it is clearly that the constant weighting scheme performs better than the CWS for the simulations with weight ratios of 1/9 and 1/19. This is because CWS has energy lost from non-trace gas, and we must add the lost energy to non-trace gas. When the weight ratio is not small enough, the energy loss will be relatively large, and hence it will affect the energy distribution. In turn, the velocity distribution will deviate strongly from the Maxwell-Boltzmann distribution, which it should be in the current test case. But when the weight ratio is less than 0.05, as shown in Figure 5.4, the CWS can perform as accurately as the constant weighting scheme. Also the relative errors of both species by the CWS decrease to a reasonable low value in a few time-steps, while that of trace species by the constant weighting method maintains unacceptably large as simulation continues. This result shows not only the CWS is as accurate as the constant weighting scheme but is also very effective for dealing the case involving trace species with the weight ratio lower than 0.05.

Three-Component Mixture Without Chemical Reaction

This simulation includes three species, Ne, Ar and He, with weight ratio 1:99:9900 in order. The simulation conditions are similar to those of two-species case. The total simulated particles are still 10,000 and the temperature in the cell is kept at 1,000 K. Figure 5.5(a) shows the velocity distribution after 20,000 time-steps. It clearly demonstrates that the constant weighting scheme is incapable of handling the most trace species because very few particles of Ne exist in the cell. Figure 5.5(b) shows the relative errors of both the trace species and the most trace species by the constant weighting scheme are too large and decrease very slowly. Thus, it is clear that the CWS performs much better than the constant weighting scheme if the trace species are involved in DSMC simulation.

5.2.2 Hypersonic Flow over a Quasi-2-D Cylinder

The second test case is a hypersonic flow over a quasi-2-D cylinder. The 1-level adaptive tetrahedral mesh is shown in Fig. 5.6. The flow conditions are listed as follows; the diameter of the cylinder is 1 m, the total number density and mach number of the inflow are 1.29E19 and 20, respectively, the mole fraction of all these cases is 1:99, the temperatures of free-stream, stagnation and surface are 20, 1620 and 291.6 K in order. Resulting Knudsen number based on free-stream condition and the diameter of the cylinder is 0.1. The conservative weighting scheme and the traditional scheme are used in these cases and the corresponding particle number are about 0.24 million and 10 million, respectively. Figure 5.7~5.9 are the contours of the number density for each species with Ar-He, Ar-Ne and O_2-N_2, respectively. From these figures, it is clear that the CWS can use few particle and computational time to obtain the same results by comparing those of traditional method.

5.3 Concluding Remarks

In the current study, the conservative weighting scheme (CWS) is assessed to handle the trace species in the DSMC simulation without chemical reaction using a single-cell and a quasi-2-D hypersonic cylinder flow by assigning lower particle weighting for trace species. The CWS is shown to be effective and accurate in DSMC simulation without chemical reaction for 2-species and 3-species mixtures with concentration ratio (trace to abundant) less than 0.05, which the constant weighting scheme is difficult to use practically. For the quasi-2-D, two-species hypersonic cylinder flow, different compounds and mole fractions are simulated to test the ability of conservative weighting scheme. The simulated particle number with constant weighting scheme reaches to 10 million, which is decreased to 0.24 million by using conservative weighting scheme and the results almost exactly the same. In short, to get correct simulated results, the CWS is necessarily when the flows involve trace particle.

Chapter 6
Chemical Reactions

Chemical reactions are important in rarefied gas flows. For example, hypersonic flows in aerospace/space engineering; film deposition technique in semiconductor and micro-reactor in medical science, etc.. In these applications, the direct simulation of Monte Carlo method is the best simulation method which can couple chemical reaction module directly. We have collaborated with Professor I. D. Boyd's group at the department of aerospace engineering of the University of Michigan in dealing with chemical reactions in DSMC. The chemical reaction function implemented in PDSC are essential the same as those in MONACO [70]. These chemical reactions, involving dissociation, recombination and exchange, are described in the following.

6.1 Chemical Reactions

In the current PDSC, we had incorporated the chemical reaction module from the *MONACO* code by Professor I. D. Boyd at the University of Michigan. It has tree typical chemical reactions, which are dissociation, recombination and exchange reactions. The difference between conventional and chemical codes is the collision partners have to calculate the reaction probability (steric factor) to determine if the chemical reaction occurs or not when two particles are chosen as collision partner. Thus, the most important part of chemical reaction is how to derive the reaction probability. The flow chart of chemistry in PDSC is shown in Fig. 6.1 and the process is described sequentially as follows:

1. Selecting two particles randomly in the current cell. The type of chemical reaction can be easily determined by these two particles.
2. Calculating the total energy and comparing with activation energy of the specific chemical reaction if this collision event has possibility to occur chemical reaction. If not, processing the normal elastic collision as usual.
3. Calculating the reaction probability according to the type of chemistry and using the Acceptant-Rejection method to determine if the chemistry occurs or not.
4. If the reaction probability is too small to process chemical reaction, the collision is processed by the elastic mechanism. But when the reaction

probability is larger than a random number, that means the chemical reaction occurs and we have to assign energies, velocities and positions of product particles. For dissociation reaction, the diatomic product is dissociated as two monatomic atoms which both have the same positions as the product. For recombination reaction, two monatomic atoms are compounded as a diatomic molecule, which position is assigned as the center point of the previous two reactants. The positions of products for exchange reaction do not change as pre-collision. No matter what chemical reaction is, the translational and internal energies are redistributed by the Larsen-Borgnakke model [72] and the post-collision velocity is assigned according to the post-translational energy.

The chemistry process will be finished when all the collision pairs are handled for all cells and then progress to the next step. The detailed derivations and validations are shown in the following section.

6.1.1 Dissociation Reaction

A typical bimolecular reaction may be written as

$$A + B \leftrightarrow C + D \tag{6.1}$$

where A, B, C, D represent separate molecular species, which can be molecules, atoms, ions and photons. The rate equation for the change in number density of species A can be written

$$-\frac{dn_A}{dt} = k_f(T)n_A n_A - k_r(T)n_C n_D \tag{6.2}$$

k_f and k_r are the forward and reverse rate constants as a function of temperature, which can be expressed in the following form (Arrhenius equation)

$$k_f = aT^b \exp(-\frac{E_a}{kT}) = aT^b \exp(-\frac{\theta}{T}) \tag{6.3}$$

where a , b are constants, and $\theta \equiv E_a/k$ is the characteristic dissociation temperature. E_a and k are the activation energy and Boltzmann constant, respectively. These values are usually obtained from experimental data.

From classical collision theory, the forward reaction may be written [85]

$$k_f n_A n_B = n_A \upsilon_{AB} \int_{\frac{E_c}{kT}}^{\infty} f(\frac{E_c}{kT}) P_r(\frac{E_c}{kT}) d(\frac{E_c}{kT}) \tag{6.4}$$

υ_{AB} is the collision rate between one A particle with all B particles in the volume, which may be expressed as

$$\upsilon_{AB} = \frac{n_B \overline{\sigma_T c_r}}{\alpha} \quad \alpha = 1 \ for \ A \neq B, \qquad \alpha = 2 \ for \ A = B \tag{6.5}$$

n_B and α are the number density of B particles and the symmetric factor, respectively. σ_T is the total collision cross section and c_r is the relative velocity of collision pair. Thus, $\overline{\sigma_T c_r}$ is the average volume that one A-B particle pair sweeps per unit time, which can be written as

$$\overline{\sigma_T c_r} = \frac{2\sigma_{ref}}{\sqrt{\pi}}[(2-w)T_{ref}]^w \Gamma(2-w)(\frac{2k}{m_r})T^{1/2-w} \tag{6.6}$$

σ_{ref}, T_{ref} and m_r are the reference cross section, temperature and reduce mass, respectively. By substitution of Eqs. (6.5) and (6.6) into Eq. (6.4), the dissociation probability (steric factor) can be obtained

$$\begin{gathered}
k_f n_A n_B = n_A \upsilon_{AB} \int_{\frac{E_c}{kT}}^{\infty} f(\frac{E_c}{kT}) P_r(\frac{E_c}{kT}) d(\frac{E_c}{kT}) = n_A \frac{n_B \overline{\sigma_T c_r}}{\alpha} \int_{\frac{E_c}{kT}}^{\infty} f(\frac{E_c}{kT}) P_r(\frac{E_c}{kT}) d(\frac{E_c}{kT}) \\
P_r(\frac{E_c}{kT}) = \frac{\alpha a \sqrt{\pi}}{2\sigma_{ref}} \frac{\Gamma(\bar{\zeta}+2-w)}{\Gamma(2-w)\Gamma(\bar{\zeta}+\frac{3}{2}+b)} (\frac{m_r}{2k})^{1/2-w} \times \\
[\frac{m_r}{2(2-w)kT_{ref}}]^w (1-\frac{E_a}{E_c})^{\bar{\zeta}+1-w} (\frac{E_c - E_a}{k})^{b-1/2+w}
\end{gathered} \tag{6.7}$$

This is the so called total collision energy (TCE) model. The equation that *MONACO* uses is the same as the Eq. (6.10) of the Bird's book [11]. Thus, Eq.(6.7) is the dissociation probability for the current DSMC code. The derivation is in Appendix A.

In the DSMC simulation, the reaction probability is obtained by accumulating the number of collisions between reactants and the number of collisions that do react. Then the rate constant can be calculated by using Eq. (6.8)

$$k_f = \frac{\overline{\sigma_T c_r}}{\alpha} \int_{\frac{E_c}{kT}}^{\infty} f(\frac{E_c}{kT}) P_r(\frac{E_c}{kT}) d(\frac{E_c}{kT}) = \frac{\overline{\sigma_T c_r}}{\alpha} \overline{P_r} \tag{6.8}$$

For the theoretical data, the rate constant is evaluated by Eq. (6.4) and the reaction probability is calculated by the following equation:

$$\overline{P_r} = \frac{k_f \alpha}{\overline{\sigma_T c_r}}, \quad \alpha = 1 \ for \ A \neq B, \qquad \alpha = 2 \ for \ A = B \tag{6.9}$$

6.1.2 Recombination Reaction

In the DSMC method, collisions are usually considered as binary. It represents a challenge to simulate three-body recombination reactions. In Vincenti and Kruger [86], it is stated that, for the recombination reaction between two nitrogen atoms and any third body M,

$$N + N + M \rightarrow N_2 + M \tag{6.10}$$

The rate of formation of N_2 is governed by

$$\frac{dn_{N_2}}{dt} = (k_b n_N n_N) n_M \tag{6.11}$$

The parentheses in Eq. (6.11) indicated that, in the new recombination model for the DSMC method, the reaction is first treated as a binary collision between two N atoms. This allows the derivation of a recombination probability in the same way as for a dissociating reaction in the previous section. The correct reaction probability must then be multiplied by the number density of the third body n_M. In the *MONACO* code, the third body is chosen randomly from those that exist in the cell. It can be any species in the cell. Thus, the number density of the third body is the total number density. The derivation of the theoretical reaction probability and rate constants are described in the following [70]:

$$\begin{aligned} &\frac{dn_{N_2}}{dt} = (k_b n_N n_N) n_M = (\overline{P_{r,b}} \cdot Z_{N,N}) \cdot n_M = \overline{P_{r,new}} \cdot Z_{N,N}, \\ &\textit{that is, } \overline{P_{r,new}} = \overline{P_{r,b}} \cdot n_M \end{aligned} \tag{6.12}$$

$\overline{P_{r,b}}$ is the probability of binary collision between two atoms. $\overline{P_{new,b}}$ is the new probability of the recombination reaction. $Z_{N,N}$ is the collision rate between nitrogen atoms. From Eq. (6.12), the rate constant can be obtained as follows:

$$(k_b n_N n_N) n_M = \overline{P_{r,new}} \cdot Z_{N,N} = \overline{P_{r,new}} \cdot \frac{n_N \cdot n_N \cdot \overline{\sigma g}}{\alpha} \tag{6.13}$$

For the *MONACO* code, the form of the binary recombination probability is chosen as [70]

$$P_{r,b} = (\frac{1}{Z_r}) E_c^{\chi}, \; \frac{1}{Z_r} = \frac{\sqrt{\pi}}{2^{1/2-w}} \cdot \frac{a_f / a_c}{k^{b_f - b_c}} \cdot \frac{m_r^{1/2-w}}{\sigma_o} \cdot (\frac{m_r}{2(2-w)kT_o})^w \cdot \frac{\Gamma(\frac{7}{2} + \frac{\zeta_1}{2} - w)}{\Gamma(\frac{7}{2} + \frac{\zeta_1}{2} - w + \chi)} \tag{6.14}$$

The new recombination probability will be multiplied by n_M, that is, $P_{new,b} = P_{r,b} \cdot n_M$.

The derivation of the *MONACO* code and the reference [70] can be found in Appendix B.

For the *MONACO* simulation, the mean recombination probability is obtained in the same way as the dissociation method. Then the rate constant can be determined from Eq. (6.13), that is,

$$k_b = \frac{\overline{P_{r,new}} \cdot \overline{\sigma g}}{\alpha \cdot n_M}, \quad \alpha = 2 \ for \ A = B \tag{6.15}$$

For theoretical results, the rate constant is determined from the Arrhenius equation and the reaction probability can be calculated by the following equation

$$k_b = aT^b \exp(-\frac{\theta}{T}), \quad P_{new,b} = \frac{\alpha \cdot k_b \cdot n_M}{\overline{\sigma g}}, \quad \alpha = 2 \ for \ A = B \tag{6.16}$$

6.1.3 Exchange Reaction

There are several reactions that change the atom in air under high temperature conditions. If a molecule AB reacts with an atom C to form a molecule AC and an atom B, which is an exchange reaction, which formula reaction is

$$AB + C \leftrightarrow AC + B \tag{6.17}$$

The derivations of the probability of exchange reaction are the same as the dissociation reaction in Section 6.1.1, which is total collision energy model (TCE). Thus, we do not repeat here. The only difference of using DSMC between dissociation and exchange reactions is in the setting of the chem.dat file, which is the information of reaction constants.

6.2 Verifications of Chemical Reactions

To verify the chemical reaction function of the DSMC code, a single cell with specific gas, which depends on each case, is simulated. All the walls are diffusive and the temperature is the same as the initial temperature. The range of temperature is from 3,000 to 13,000 K. The value of the initial number density is assigned as $\frac{\rho_d}{\rho} = 10^7$. ρ_d is the characteristic density for dissociation, which is a function of temperature.

In practice, the reactions include dissociation, recombination and exchange reactions. To simplify the chemical reaction, only one reaction (direction) is permitted. Twenty-two cases are simulated and the corresponding constants of rate coefficients are used are shown in Table 6.1. The reaction probability (P_r) and the rate constant (k_f) of each simulation under different temperatures will be calculated and compared with the theoretical data. The definition of the reaction probability (P_r) is the ratio of the collisions that do react to the total collisions between reactants. Thus, the number of collisions between reactants and the number of collisions that do react will be accumulated through the DSMC simulations, and the reaction probability can be easily calculated from these two numbers. These reactions are described and discussed in turn.

6.2.1 A Single Cell with One-Directional Reaction

1. ***Dissociation reaction between nitrogen molecules, ($N_2 + N_2 \rightarrow N + N + N_2$);***

For the 1st simulation, the initial gas is 100% nitrogen molecules (N_2). The dissociation reactions will occur when two nitrogen molecules collide with each other with enough collision energy. The probability of dissociation uses the total collision energy model (TCE). Table 6.2 is the simulation and theoretical results. Figure 6.2 is the average probability and rate constant of simulation and theoretical data as a function of temperature. In Figure 6.2, the data of simulation are in agreement with the theoretical data except the data at 5,000 K. The reason is that very few reactions occur due to lower reaction probability. Only ten reactions occur in this situation, which leads to inaccuracy by insufficient sampling.

2. ***Dissociation reaction between nitrogen molecules, ($N_2 + N \rightarrow N + N + N$);***

The 2nd test case is the dissociation reaction involving the collision between nitrogen molecule (N_2) and atom (N). The initial number densities of both species are the same. Only collision between N_2 and N will be accumulated and processed. Table 6.3 and Figure 6.3 are the comparison of simulation and theoretical results. As mentioned previously, the data of lower temperature is not perfect due to fewer chemical reactions.

3. ***Recombination reaction between nitrogen molecule and atoms ($N + N + N_2 \rightarrow N_2 + N_2$);***

The 3rd test case is the recombination reaction of nitrogen atoms (N), and the third body must be nitrogen molecule (N_2). The initial particle densities of N_2 and N are the same. Comparisons between the simulation and the theoretical results are shown in Table 6.4 and Figure 6.4. The agreement is remarkable. But the data of higher temperature is not very close to the theoretical data. The value is overestimated because a continuous reaction probability, i.e. TCE model, is used. The TCE model is constructed in such a manner that it reproduced the rate constant given by the Arrhenius equation for an equilibrium continuous distribution. This model will has some questions with using discrete vibrational energy model [70].

4. ***Recombination reaction between nitrogen molecule atoms*** **$(N+N+N \rightarrow N_2+N)$*;***

The 4th test case is the recombination reaction with all nitrogen atoms (N). All the particles are nitrogen atoms. The results are shown in Table 6.5 and Figure 6.5. This simulation has the same trend with the third test case, that is, the error increased with the temperature. The reason has been mentioned in previous section.

5. ***Dissociation reaction between oxygen molecules,*** **$(O_2+O_2 \rightarrow O+O+O_2)$*;***

After simulating the dissociation and recombination of nitrogen gas, we repeat all types of analysis for oxygen gas as the cases 5~8. The range of temperatures of these simulations is from 3,000 to 7,000 K. Table 6.6 and Figure 6.6 are the simulation and theoretical results. In Figure 6.6, the data of simulation are in agreement with the theoretical data except the data at 3,000 K, which has the same trend as nitrogen cases.

6. ***Dissociation reaction between oxygen molecule and oxygen atom*** **$(O_2+O \rightarrow O+O+O)$*;***

The 6th test case is the dissociation reaction involving the collision between oxygen molecule (O_2) and atom (O). The initial number densities of both species are the same. Table 6.7 and Figure 6.7 are the comparisons of simulation and theoretical results. The agreement is remarkable.

7. ***Recombination reaction between oxygen molecule and atoms*** **$(O+O+O_2 \rightarrow O_2+O_2)$*;***

The 7th test case is the recombination reaction of oxygen atoms (O), and the third body is oxygen molecule. The initial particle densities of O_2 and O are the same. Comparisons between the simulation simulation and the theoretical results are shown in Table 6.8 and Figure 6.8. The results are not very close to the theoretical results due to low reaction probability, but it should be acceptable.

8. ***Recombination reaction between oxygen molecule atoms*** **$(O+O+O \rightarrow O_2+O)$*;***

The 8th test case is the recombination reaction with all oxygen atoms (O) and the third body particle is oxygen atom. All the particles are O in the beginning. Table 6.9 and Figure 6.9 are the comparisons of simulation and theoretical results. By comparing this reaction with case 7, this reaction has higher reaction probability when the third body is O. And the simulated results agree with theoretical results.

9. *Dissociation of nitric oxide with nitrogen molecule, ($NO + N_2 \rightarrow N + O + N_2$);*

Now, we are interested in the chemical reactions of nitric oxide (NO). Cases 9~13 are the dissociation reaction of nitric oxide (NO) with nitrogen molecule (N_2), oxygen molecule (O_2), nitric oxide (NO), nitrogen atom (N) and oxygen atom (O), respectively. When nitric oxide collides with diatomic particles, which are cases 9~11, the setting of variables of rate constants are the same. The setting of chem..dat of cases 12 and 13 also are the same for monatomic particles. In case 9, the initial number density of NO and N_2 are the same. Table 10 and Figure 6.10 are the average probability and rate constant of simulation and theoretical data as a function of temperature. The data of simulation are in agreement with the theoretical data.

10. *Dissociation of nitric oxide with oxygen molecule, ($NO + O_2 \rightarrow N + O + O_2$);*

The 10th test case is the dissociation reaction involving the collision between nitric oxide and oxygen molecules. Table 6.11 and Figure 6.11 are the comparison of simulation and theoretical results. The results are reasonable.

11. *Dissociation of nitric oxide with nitric oxide molecule, ($NO + NO \rightarrow N + O + NO$);*

The 11th test case is the dissociation reaction between nitric oxide molecules (NO). The initial particles are 100% nitric oxide molecules. Comparisons between the simulation and the theoretical results are shown in Table 6.12 and Figure 6.12. The agreement is remarkable.

12. *Dissociation of nitric oxide with nitrogen atom, ($NO + N \rightarrow N + O + N$);*

The 12th test case is the dissociation reaction of nitric oxide (NO) and nitrogen atom (N). The initial particle number densities of the reactants are the same. Comparisons between the simulation and the theoretical results are shown in Table 6.13 and Figure 6.13. The simulated results are very close to the theoretical results.

13. *Dissociation of nitric oxide with oxygen atom, ($NO + O \rightarrow N + O + O$);*

The 13th test case is the dissociation reaction of nitric oxide (NO) and oxygen atom (O). The initial particle number densities of NO and O are the same. Comparisons between the simulation and the theoretical results are shown in Table 6.14 and Figure 6.14. The simulated results agree with the theoretical results.

14. *Recombination of nitric oxide with nitrogen molecule, ($N + O + N_2 \rightarrow NO + N_2$)*

The following test cases are the reverse reactions of cases 9~13, which are the recombination reaction to form nitric oxide. The third body particles of case 14 to 18 are

N_2, O_2, NO, N and O in order. When the third particles are molecules (cases 14~16), the coefficients of rate constant in chem.dat are the same. For cases 17 and 18, in which the third body particles are monatomic particles, are the same. Case 14 is the recombination reaction with nitrogen molecules (third body particle). Table 6.15 and Figure 6.15 are the comparison of the simulation and theoretical results. The simulated results are scatter and different from the theoretical results due to low reaction probability. Only about 50 real reactions occur in these simulations, which lead to inaccurate results.

15. Recombination of nitric oxide with oxygen molecule, ($N+O+O_2 \rightarrow NO+O_2$);

The 15th test case is the recombination of nitric oxide (NO) and the third body particle are oxygen molecules (O_2). Table 6.16 and Figure 6.16 are the results of simulation and theory. The simulations do not agree perfectly because of the low reaction probability, especially for the case of higher temperature, in which only 13 real reactions occur.

16. Recombination of nitric oxide with nitric oxygen molecule, ($N+O+NO \rightarrow NO+NO$);

The 16th test case is recombination reaction of nitric oxide (NO), and the third body particles are nitric oxides (NO). Comparisons of simulation and theoretical results are shown in Table 6.17 and Figure 6.17. This case has the same situation as case 14 and 15. The simulated results should be improved by running move time-steps to obtain enough sampling.

17. Recombination of nitric oxide with nitrogen atom, ($N+O+N \rightarrow NO+N$);

The 17th test case is the recombination to form nitric oxide (NO) by colliding with the third body particle, which is nitrogen atom (N). Table 6.18 and Figure 6.18 are the comparisons between simulation and theoretical results. From Table 6.18, the reaction probability is higher than the reaction with monatomic third body particle. But the samplings are still not enough to obtain accurate results.

18. Recombination of nitric oxide with oxygen atom, ($N+O+O \rightarrow NO+O$);

The 18th test case is the final simulation about the recombination of nitric oxide (NO), and the third body particle is oxygen atom (O). Table 6.19 and Figure 6.19 are the results of simulation and theory. In conclusion, the reaction probability of recombination is lower than the dissociation. The recombination reaction needs more time-steps to obtain enough sampling and accurate results.

19. Exchange reaction between nitrogen molecule and oxygen atom

($N_2 + O \to NO + N$)

After verifying the dissociation and recombination reaction, the following benchmarks are about the exchange reaction between molecules and atoms. The 19th test case is the exchange reaction (forward) between nitrogen molecule (N_2) and oxygen atom. In this case, nitrogen molecule will dissociate into two nitrogen atoms and exchange one of the nitrogen atoms with the oxygen atom, then forms one nitric oxide molecule and one nitrogen atom. The initial particle number densities of N_2 and O are the same. Only collisions between these two species will be accumulated to calculate the reaction probability. The results are shown in Table 6.20 and Figure 6.20. The simulated results agree perfectly with the theoretical data.

20. Exchange reaction between nitric oxide and nitrogen atom ($NO + N \to N_2 + O$)

The 20th test case is the exchange reaction between nitric oxide (NO) and nitrogen atom (N), which is the reverse reaction of the 19th test case. The initial particle number densities of NO and N are the same. The results are shown in Table 6.21 and Figure 6.21. Although the simulated results have some discrepancies, the simulations are acceptable.

21. Exchange reaction between nitric oxide and oxygen atom ($NO + O \to O_2 + N$)

The 21st test case is also the exchange reaction (forward), which is a reaction between nitric oxide (NO) and oxygen atom (O). The initial particle number densities of NO and O are the same. The results are shown in Table 6.22 and Figure 6.22. These simulated results are also remarkable by comparing to the theoretical data.

22. Exchange reaction between oxygen molecule and nitrogen atom ($O_2 + N \to NO + O$)

The 22nd test case is the exchange reaction between oxygen molecule (O_2) and nitrogen atom (N), which is the reverse reaction of 21st test case. The initial particle number densities of O_2 and N are the same. The results are shown in Table 6.23 and Figure 6.23. The simulated results are in good agreement with theoretical data.

6.2.2 A Single Cell with Pure Gas (Dissociation and Recombination)

After verifying the single direction chemical reaction, the following test case is to simulate the pure species with forward and backward reactions under different number densities and temperatures. First, a new variable called the degree of dissociation and are defined by

$$\alpha \equiv \frac{N^a}{N_A} = \frac{mN^a}{mN_A} = \frac{mass\ of\ dissociated\ A-atoms}{total\ mass\ of\ gas} \tag{6.18}$$

N^a and N_A are the particle number of dissociated A-atoms and the total number of A-atoms present in the mixture, respectively. There are two simulations, in which one is nitrogen gas and the other is oxygen gas. The ranges of temperature of nitrogen and oxygen gas are 5,000~13,000 and 3,000~7,000 K, respectively. Three initial number densities are used for these simulations, which are $\frac{\rho_d}{\rho} = 10^5, 10^6, 10^7$, respectively. The constants of the Arrhenius equation of dissociation are from the user guide of the simulation code. The relation between forward and backward rate constants is described in Eq. (6.19),

$$K_e = \frac{k_f}{k_b} = \frac{a_f T^{b_f} e(-\theta_f / T)}{a_b T^{b_b} e(-\theta_b / T)} \tag{6.19}$$

where a, b and θ are constants for each reaction and the K_e, k_f and k_b are the equilibrium constant, forward and reverse rate constants, respectively. These coefficients are functions of temperature. K_c can be obtained from Vincenti and Kruger [86]. The coefficients of recombination rate constant can be calculated by Eq. (6.19). The theoretical degree of dissociation is

$$\frac{\alpha}{1-\alpha} = \frac{\rho_d}{\rho} e^{-\theta/T} \tag{6.20}$$

ρ_d, ρ and θ are the characteristic density for dissociation, number density and the characteristic temperature of dissociation.

Figure 6.24(a) and 6.24(b) is the degree of dissociation as a function of temperature for three values of the density with pure nitrogen and oxygen gas, respectively. The symbols indicate the simulated results from simulation and the lines are the theoretical results. As can be seen, the simulations are reasonable in Figure 6.24. The data of lowest temperature is not good enough due to too few chemical reactions. A more correct result should be obtained by using more particles and time-steps.

6.2.3 A Single Cell with 5 Species with 34 Reactions

In this section, simulations with dissociation, recombination and exchange reactions (34 chemical reactions without ionizations) of a single cell are presented. The flow conditions are as follows; the initial total number density is 0.01 of standard atmospheric density and the initial mole fractions of nitrogen (N_2) and oxygen molecule

(O_2) are 0.8 and 0.2, respectively. The total number of particles is 100,000. The cell size of each simulation depends on the number density, which the ratio of cell size to mean free path is about 0.2. The order of time-step is 2.E-8. The transient time-step before sampling and the total time-step are 2,900,000 and 3,000,000, respectively. The real time is 0.06 seconds. The temperature range of these simulations is 2,000 to 16,000 K. The original and new fitting input files (chem.dat) are simulated for comparison. For the new fitting rate constants of $O_2 + N \rightarrow NO + O$ reaction is $k_f = 3.91\times10^{-19}\mathrm{T}^{0.65279}\exp(-4.970\times10^{-20}/\mathrm{kT})$. Tables 6.24 are the mole percentage of each species with original and new fitting rate constant. The mole percent is the number of moles of each species as a percentage of the total number of moles, which is also the percentage of number of particles.

Figure 6.25 shows the detailed composition as a function of temperature for each species. Figures 6.25(a) and 6.25(b) are original and new fitting input data, respectively. The symbols in these figures are the steady results, which are obtained from a solution of the rate equations using the same equilibrium constants. Some conclusions are listed as follows;

1. No chemical reaction occurs for the present data at 2,000 K, which is reasonable from the results of the previous single direction chemical reaction in Section 6.2. The dissociation probability of oxygen molecules (O_2) is too low for lower temperature. However, some dissociation of oxygen molecules occurs in the rate equation solution at 2,000 K.
2. The simulated results of nitrogen molecule (N_2) are overestimated when the temperature is higher than 10,000 K. It may be because most of nitrogen molecules are dissociated under higher temperature and leads to too few particles to obtain correct results.
3. The oxygen molecules (O_2) are scattered because the particles are too rare when the temperature is higher. It needs more simulated particles and time-steps to predict the results.
4. The trends of nitric oxide (NO), nitrogen atom (N) and oxygen atom (O) are in very good agreement with the rate equation analysis.
5. If all the recombinations are neglected in the simulation, all the nitrogen molecules (N_2) and oxygen molecules (O_2) will be dissociated as nitrogen atoms (crossed circle) and oxygen atoms (crossed square) when temperature is

16,000 K.

6.3 Concluding Remarks

This chapter has presented a chemical reaction module for the DSMC code. The total collision energy (TCE) model is used to calculate the reaction probabilities of dissociation and exchange. An extended TCE model, which is the three-body collision model, is applied to obtain the recombination probability. The function is successful verified by several benchmarks and most of those results agree with theoretical data. In general, the probabilities of dissociation increase with increasing temperatures and the recombination probabilities decrease with increasing temperatures from the single 2-D cell test cases. Those cases show the chemical reactions appear and need to be considered when the temperature is very high. There are some suggestions to the users of using chemical reaction module, which are listed in the following;

1. It needs a large total number of time-steps and a long transient time before sampling.
2. In general, the recombination reaction has lower probability to react. Thus, the recombination simulations need more time-steps to obtain enough sampling and accurate results.

Chapter 7
Applications and Examples

The proposed DSMC method combining mesh adaptation, variable time-step scheme, dynamic domain decomposition, conservative weighting scheme and chemical reaction has been verified successfully by the test problems in Chapters 3-6, respectively. Thus, to demonstrate the powerful capability of the current simulation tool, we will apply it to compute three rather challenging flows, which are an underexpanded twin-jet flow, a 3-D Apollo re-entry vehicle at 100 km altitude and a 3-D sphere shape re-entry vehicle at an altitude of 90 km. The results will then compare with previous simulated or experimental studies available in the literature. Note that either only preliminary results or problem definition will be described in the following.

7.1 Near-Continuum Underexpanded Twin-jet Interaction

<u>Flow and Simulation Conditions</u>

For a truly three-dimensional flow, two parallel, near-continuum, under-expanded jets issuing from sonic orifices into a near-vacuum environment is selected as the first application since the experimental data is available [87]. Sketch of the twin-jet interaction is shown in Fig. 7.1(a). It is expected that a secondary jet form between the two primary parallel jets under certain flow conditions. Corresponding flow conditions represent a challenging problem since it involves flow regimes from near-continuum at the inlet to near free-molecular flow at the outlet, where the DSMC method may be the only available tool for analyzing this problem. Related flow conditions are listed as follows: the test gas is nitrogen gas; stagnation pressure P_o=870 Pa; stagnation temperature T_o=285 K; background pressure P_b=3.7 Pa; resulting pressure ratio P_o/P_b=235. Background pressure effect is either neglected or included in the simulation, in which previous simulation by Dagum and Zhu [88] has ignored and vacuum boundary is used instead for simplicity. The corresponding Knudsen number Kn_{th} (=λ_{throat}/D; D is the throat diameter, which equals 3 mm) is 0.00385.

Computational domain is reduced to 1/4 of the original physical domain by taking advantage of the geometrical symmetry of the flow. Height (H), width (W) and length (L) of the simulation domain are taken long enough as 10D, 10 D and 20 D, respectively, as shown in Fig. 7.1(b), where the positive z-direction is out of the paper. Not that the distance between the centers of two primary jet centerline is 3 times the diameter of the orifice. Internal energy is relaxed through the Borgnakke-Larsen model [72], using a fixed rotational collision number of 5. The variable time-step method and an *h-refinement* mesh are used to reduce the computational time further and to increase the accuracy of the solution, respectively. An *h-refinement* three-dimensional mesh with mesh quality control is used in this simulation (32 processors on IBM-SP2 system at NCHC) to increase the accuracy of the solution. The resulting simulated particles are about 10 millions at steady state and the number of cells is approximately 0.66 million after two levels of mesh refinement (Fig. 7.2). Figure 7.2(a) shows the surface mesh along x-y and y-z planes, while Fig. 7.2(b) illustrates the exploded view of the surface mesh in the same planes near the sonic orifice, where the mesh is refined. Corresponding local cell Knudsen number distribution is shown in Fig. 7.3, where most of the flow field satisfies the general cell-size requirement except the locations very near the sonic orifice ($0.5<Kn_{cc}<1$). Besides, 20,000 time-steps are used to sample the particles for obtaining macroscopic properties.

<u>Dynamic Domain Decomposition</u>

Figure 7.4 illustrates the initial and final domain decomposition on the surfaces of x-y, y-z and z-x planes for the parallel twin-jet interaction using vacuum outflow boundary. The initial domain decomposition (Fig. 7.4(a)) is obtained by assigning equal weight to each cell, which is the same as the three-dimensional sphere flow. Final domain decomposition changes dramatically as compared with the initial domain decomposition. In this case, the computational time is saved up to 100% using dynamic domain decomposition, as compared with that using static domain decomposition. Similar trend is found for the case using pressure outflow boundary.

<u>Properties Contour</u>

Figure 7.5 is the particle distribution by using constant time-step and variable time-step schemes. As the figure illustrates, the particle distribution by using variable time-step scheme is less than those by constant variable time-step scheme. And the particle distribution is more uniform. Figure 7.6 presents the normalized density contours (with respect to the density at the sonic orifice) on x-y and y-z planes using vacuum outflow boundary. Note that y-z plane is the symmetric plane between the two primary jets. It is clear that a secondary jet is formed and centered along y-axis between the two primary jets. In addition, the density expends similar to a single jet on the side, where there is no jet interaction. Similar evidence for the formation of a secondary jet

between the two jets exists also for temperature (translational and rotational) contours in Fig. 7.7, which is label in degree of Kelvin.

Centerline Properties Distributions

Figure 7.8 illustrates the density and rotational temperature distribution along the symmetric centerline (y-axis), respectively, using different outflow boundary conditions (vacuum and pressure). Experimental data measured using laser induced fluorescence by Soga [87] and the DSMC simulation data by Dagum and Zhu [88] using vacuum outflow boundary condition are also included for comparison. It is clear that current simulation data using vacuum outflow boundary condition agree favorably with experimental data [87] and previous DSMC simulation data [88] using the same outflow boundary condition. It is, however, that the centerline density increases again for y/d>10 when pressure outflow boundary condition is used. Similar trend is also found for rotational temperature distribution in Fig. 7.8(b). Note that the background pressure in the vacuum chamber is maintained at 3.7 Pa, where no measurement location is clearly specified in the paper [87]. The only possible way to fully duplicate the experimental conditions is to conduct chamber-scale simulation, which is not possible due to the unclear experimental conditions provided in the paper [87].

7.2 Hypersonic Flow around a Three-Dimensional Apollo Re-entry Vehicle

Flow and Simulation Conditions

The second application is a three dimensional Apollo case, which is at 100 km altitude and simulated with 34 chemical reactions [89]. Half of the whole Apollo geometry is simulated to save the computational time. Related flow conditions are listed as follows: the altitude of this case is 100 km and the angle of attack is 25^o; the initial number densities of inflow gases are determined by the MSIS-E-90 Atmosphere model (http://nssdc.gsfc.nasa.gov/space/model/models/msis.html), which are nitrogen, oxygen and oxygen atoms with 8.467E18, 2.025E18 and 3.995E17, respectively; the velocity and temperature are 9,592.64 m/s and 190.6 K, respectively; the wall temperature is fixed and is predicted by the Stephan-Boltzmann Law, which is 1,378 K; the corresponding Knudsen number *Kn* (based on the radius of aft compartment) is 0.033. The cell number and particle number of this simulation are about 750,000 and 15,000,000, respectively. The mesh is shown as Fig. 7.9(a). Figure 7.9(b) is the particle locations on the slide near the symmetric plane. Each dot means a simulated particle. There have more particles gathering in front of the object because the bow shock. And fewer particles existed behind the object.

Properties Contour

Figure 7.10 is number density contour of each species, which is a slide of X-Z plane that can easily understand the flow field of Apollo case. Figure 7.10(a)~7.10(e) represent the number densities of N2, O2, NO, N and O, respectively. In these figures, a clear bow shock is formed in front of the object and the weak region is at under part behind the object. The chemical reactions are occurred at those regions to produce No and N because the temperature is heated by the bow shock. The temperature and velocity contours are shown as Fig. 7.11. Figure 7.11(a)~7.11(c) are the contours of translation, rotation and vibration, respectively. The maximum translational temperature is located on the bow shock near the upper tip of the object. The value reaches to 33,000 K which leads to chemical reactions occurred. These three temperature contours show strong degree of thermal non-equilibrium at the shock and wake regions. Figure 7.11(d)~7.11(f) illustrate the velocity contours in three coordinate directions near X-Z plane. In Fig. 7.11(d), the U-velocity along X-direction is decreased by the bow shock and then increased by rarefaction. The V-velocity along Y-direction in Fig. 7.11(e) is almost zero because it is a symmetric case. The W-velocity along Z-direction in Fig. 7.11(f) is interesting. Particles flowing through the lower part of the Apollo are faster than those through the upper part. And the separation exits behind the object.

7.3 A Three-Dimensional Re-entry Sphere Flow

Flow and Simulation Conditions

The third application is a three dimensional sphere case, which is at 90 km altitude and simulated with 19 chemical reactions [89]. According to the verifications of a single cell, the order of recombination probability is much less than the dissociation and exchange reactions. Thus, recombination is negligible in this simulation. Only one in sixteenth physical domain is simulated because its axis-symmetric geometry. Related flow conditions are listed as follows [69] the altitude of this case is 90 km and the free-stream density is 3.43E-6 kg/m^3; the initial mole fraction of oxygen molecule, nitrogen molecule and oxygen atoms are 0.209, 0.788 and 0.004, respectively; the free-stream velocity and temperature are 7,500 m/s and 188 K, respectively; the wall temperature is 350 K with full diffuse surface; the corresponding Knudsen number *Kn* and Reynolds number *Re* (based on the diameter of the sphere, which is 1.6 meter) are 0.01 and 3,243, respectively. The two-level adaptive cell number and particle number of this simulation are about 670,000 and 3,000,000, respectively. The adaptive mesh refinement can easily capture the bow shock in the front of the sphere because it is a hypersonic flow, which is shown as Fig. 7.12.

<u>Properties Contour</u>

Figure 7.13(a)~(c) illustrate flow field contours of this simulation, which including normalized density, normalized overall temperature and Mach number on the X-Z plane, respectively. The normalized properties are based on the free-stream condition. As we can see, there is a very strong bow shock stands in the front of the sphere and the wake region exists behind the object. The normalized density and overall temperature are increased along the X-axis first because the bow shock effect, which the maximum value are about 120 and 60, respectively. Figure 7.14 represents the mole fractions for five species, which are N2, O2, NO, N and O, along the stagnation line. Nitrogen and oxygen molecules stay constant before the bow shock and then decreasing near the shock region because the temperature goes very high, which can process the dissociation reaction. Thus, the mole fractions of nitric oxide, nitrogen atom and oxygen atom are increased. Figure 7.15 shows the surface coefficients, including pressure, skin-friction and heat transfer coefficients, along the symmetric line of the surface of the sphere. This figure is plotted as a function of the circumferential angle (θ) measured clockwise from the stagnation point and the simulation data of Dogra et al. [69] is also plotted into this figure for comparison. The pressure (Fig. 7.15(a)) and heat transfer (Fig. 7.15(c)) coefficient decreased with increasing circumferential angle and the maximum value are at stagnation point θ=0^{o}. The simulated result agrees very well with the reference [69]. The skin-friction coefficient increased then decreasing along the sphere surface and the maximum value is near θ=40^{o}. As we can see the results of skin-friction and heat transfer coefficients are pretty scattering because the particle sampling is not enough. More particle number and longer time-steps can overcome this problem and obtain more accurate simulated results. Although the results are not exact the same, the results are acceptable by comparing with reference data.

7.4 Concluding Remarks

This chapter presents several three-dimensional applications, which are a near-continuum parallel twin-jet, the Apollo re-entry vehicle flow and a hypersonic re-entry sphere flow. All these cases are interesting and important but it is very complicated and the computation is time-consuming. By using the present PDSC, it can efficiently simulate and obtain accurate results by using reasonable computing time, which demonstrates its capability.

Chapter 8
Conclusions

8.1 Summary

The direct simulation of Monte Carlo (DSMC) method is used to simulate gas flows under rarefied gas environment in these four decades. Mesh resolution, huge computational cost and flow involving specific problems are three main drawbacks by using the DSMC method. To overcome these disadvantages, a general-purpose parallel DSMC code (PDSC) is presented and verified by comparing with experiment and references. The current PDSC has following features;

1. It is a three-dimensional DSMC code with unstructured tetrahedral mesh, which is easier and flexible to deal with the flow with complex and irregular geometry. A cell-by-cell particle tracking technique can easily and correctly track particle movements.
2. A general unstructured adaptive mesh refinement with variable time-step scheme is developed to save computational time and to obtain accurate results. The particle number of a 3-D hypersonic sphere flow by using constant time-step scheme is 1.7 million, which can be reduce to 0.34 million particles by applying variable time-step scheme and the transient time is decreased to only 25%. The computational efficiency can be speeded up to 10 times and more accurate result can be obtained if simulation uses suitable mesh resolution and time-step.
3. Parallelization of a 3-D DSMC method is developed to save the computing time. The parallel DSMC code with dynamic domain decomposition is also implemented to speed up the computational efficiency. The dynamic domain decomposition function can alleviate the unbalancing loading between each processor. This method can save 30-100% for the driven cavity flow by using static domain decomposition method.
4. Conservative weighting scheme (CWS) for flows with trace species is incorporated in PDSC to obtain reasonable number of simulated particles. A quasi 2-D hypersonic cylinder flow is used to verify this function. The total particle number by using constant weighting scheme is up to 10 million, which can be reduced to 0.24 million if the conservative weighting scheme is

activated.

5. Chemical reaction is developed for hypersonic reactive flows. It is tested by several cases, which includes comparisons the reaction probability (Pr) and rate constant (k_f) of each single reaction, degree of dissociation of pure species and the mole fraction of air (5 species) for a 2-D single cell. This function can correctly simulate dissociation, exchange and recombination reactions and the results agree with theoretical data.

Finally, several applications, which are a 3-D parallel underexpanded twin-jets, a hypersonic Apollo re-entry vehicle and a hypersonic re-entry sphere flow, are simulated to demonstrate potential and ability of the PDSC.

8.2 Recommended Future Studies

There is still has something to make the PDSC complete. The diagram of recommended future studies is shown in Fig. 2.4 and described in the following:

1. To develop hybrid mesh code in PDSC. This can be a hybrid code with structured/unstructured and tetrahedral/hexahedral mesh system, which can save the cell number and reduce the time of tracking particle;
2. To couple with other numerical solves which can extend its capability of simulate complex flows. Combining with computational fluid dynamics (CFD) solver can solve the flow has continuum flow region. Combining with particle-in-cell (PIC) method can simulate flows with plasma. Incorporating with particle flux method (PFM) can simulate inviscid flow.

References

1. Lee, Y. K. and Lee, J. W., "Direct simulation of compression characteristics for a simple drag pump model", Vacuum, 47, pp. 807-809, 1996.
2. Lee, Y. K. and Lee, J. W., "Direct simulation of pumping characteristics for a model diffusion pump", Vacuum, 47, pp. 297-306, 1996.
3. Chang, Y. W., *et al.*, "Pumping performance analyses of a turbo booster pump by direct simulation Monte Carlo method", Vacuum, 60, pp. 291-298, 2001.
4. Plimpton, S. and Bartel, T., "Parallel particle simulation of low-density fluid flows," U.S. Department of Energy Report No. DE94-007858, 1993.
5. Yang, X. and Yang, J. M., "Micromachined membrane particle filters", Sensors and Actuators A, 73(1-2), pp. 184-191, 1999.
6. Piekos, E. S. and Breuer, K. S., "Numerical modeling of micromechanical devices using the direct simulation Monte Carlo method", Transaction for ASME Journal of Fluids Engineers, 118(3), pp. 464-469, 1996.
7. Nance, R. P., *et al.*, "Role of boundary conditions in Monte Carlo simulation of microelectromechanical systems", AIAA Journal, 12(3), pp. 447-449, 1998.
8. Tseng, K.-C., "Analysis of Micro-scale Gas Flows with Pressure Boundaries Using The Direct Simulation Monte Carlo Method", Mechanical Engineering, National Chiao-Tung University, Taiwan, Master Thesis, 2000.
9. Schaff, S. and Chambre, P., Fundamentals of Gas Dynamics, Princeton University Press, Princeton, NJ, 1958, Chapter H.
10. Bird, G. A., Molecular Gas Dynamics, Clarendon Press, Oxford, UK, 1976.
11. Bird, G. A., Molecular Gas Dynamics and the Direct Simulation of Gas Flows, Oxford University Press, New York, 1994.
12. Nanbu, K., "Theoretical Basis on the Direct Monte Carlo Method," Rarefied Gas Dynamics, 1, Boffi, V. and Cercignani, C. (editor), Teubner, Stuttgart, 1986.
13. Wagner, W., "A convergence proof for Bird's direct simulation Monte Carlo method for the Boltzmann equation", Journal Stat Physics, 66(3/4), pp. 1011-1044, 1992.
14. Alexander, F. J., *et al.*, "Direct Simulation Monte Carlo for Thin-film Bearings", Physics of Fluids A, 6, pp. 3854-3860, 1994.
15. Stefanov, S. and Cercignani, C., "Monte Carlo simulation of the Taylor-Couette flow of a rarefied gas", Journal of Fluid Mechanics, 256, pp. 199-213, 1993.

16. Golshtein, E. and Elperin, T., "Convective instabilities in rarefied by direct simulation Monte Carlo method", Journal of Thermophysics and Heat Transfer, 10, pp. 250-256, 1996.
17. Beskok, A. and Karniadakis, G.. E. "A Model for Flows in Channels, Pipes and Ducts at Micro and Nano Scales", Microscale Thermophysical Engineering, 3(1), pp. 43-77, 1999.
18. LeBeau, G. J. and Lumpkin III, F. E., "Application highlights of the DSMC Analysis Code (DAC) software for simulating rarefied flows", Computer Methods in Applied Mechanics and Engineering, 191(6-7), pp. 595-609, 2001
19. Boyd, I. D., "Vectorization of a Monte Carlo Simulation Scheme for Nonequilibrium Gas Dynamics", Journal of Computational Physics, 96, pp. 411-427, 1991.
20. Ivanov, M. S., *et al.*, "Statistical simulation of reactive rarefied flows: numerical approach and applications", AIAA Paper, 98–2669, 1998
21. Merkle, C. L., "New Possibilities and Applications of Monte Carlo Methods", 13th International Symposium, Rarefied Gas Dynamics, pp. 333-348, Belotserkovsk, II (editor), 1985.
22. Shimada, T. and Abe, T., "Applicability of the Direct Simulation Monte Carlo Method in a Body-fitted Coordinate System", Progress in Astronautics and Aeronautics, Rarefied Gas Dynamics, pp. 258-270, Muntz, *et al* (editor), 1989.
23. Olynick, D. P., *et al.*, "Grid Generation and Application for the Direct Simulation Monte Carlo Method to the Full Shuttle Geometry", AIAA Paper, 90-1692, 1990.
24. Robinson, C. D., Particle Simulations on Parallel Computers with Dynamic Load Balancing, Imperial College of Science, Technology and Medicine, UK, Ph.D. Thesis, 1998.
25. Robinson, C. D. and Harvey, J. K., "A Parallel DSMC Implementation on Unstructured Meshes with Adaptive Domain Decomposition," Proceeding of 20th International Symposium on Rarefied Gas Dynamics, pp. 227-232, Shen, C. (editor), Peking University Press, 1996.
26. Robinson, C. D. and Harvey, J. K., "Adaptive Domain Decomposition for Unstructured Meshes Applied to the Direct Simulation Monte Carlo Method", Parallel Computational Fluid Dynamics: Algorithms and Results using Advanced Computers, pp. 469-476, 1997.
27. Boyd, I. D., *et al.*, "Particle simulation of helium microthruster flows", Journal of

Spacecraft and Rockets, 31, pp. 271-281, 1994.

28. Boyd, I. D., *et al.*, "Experimental and numerical investigations of low-density nozzle plume flows of nitrogen", AIAA Journal, 30, pp. 2453-2461, 1992.
29. Kannenberg, K. C., "Computational method for the Direct Simulation Monte Carlo technique with application to plume impingement", Cornell University, Ithaca, New York, Ph.D. Thesis, 1998.
30. Wilmoth, R. G., *et al.*, "DSMC Grid Methodologies for Computing Low-Density, Hypersonic Flow About Reusable Launch Vehicles", AIAA Paper, 96-1812, 1996.
31. Wu, J.-S. and Lian, Y.-Y., "Parallel Three-Dimensional Direct Simulation Monte Carlo Method and Its Applications", Computers & Fluids, 32(8), pp. 1133-1160, 2003.
32. Powell K. G., *et al.*, "Adaptive mesh algorithms for computational fluid dynamics", Algorithmic Trends in Computational Fluid Dynamics, pp. 303-337, Springer Verlag Co, New York, 1992.
33. Rausch, R. D., *et al.*, "Spatial Adaptation Procedures on Unstructured Meshes for Accurate Unsteady Aerodynamics Flow Computation", AIAA Paper, 91-1106, 1991.
34. Connell, S. D. and Holms, D. G.., "Three-Dimensional Unstructured Adaptive Multigrid Scheme for the Euler Equations", AIAA Journal, 32, pp. 1626-1632, 1994.
35. Kallinderis, Y. and Vijayan, P., "Adaptive Refinement-Coarsening Scheme for Three-Dimensional Unstructured Meshes", AIAA Journal, 31, pp. 1440-1447, 1993.
36. Wang, L. and Harvey, J. K., "The Application of Adaptive Unstructured Grid Technique to the Computation of Rarefied Hypersonic Flows Using the DSMC Method", 19th International Symposium, Rarefied Gas Dynamics, pp. 843-849, Harvey, J. and Lord, G. (editor), 1994.
37. Oh, C. K., *et al.*, "Massive parallelization of DSMC combined with the monotonic Lagrangian grid", AIAA Journal, 34, pp. 1363-1370, 1996.
38. Garcia, A., *et al.*, "Adaptive Mesh and Algorithm Refinement using Direct Simulation Monte Carlo", Journal of Computational Physics, 154, pp. 134-155, 1999.
39. Kuo, C.-H., "The Direct Simulation Monte Carlo Method Using Unstructured Adaptive Mesh and Its Application", Mechanical Engineering, National

Chiao-Tung University, Taiwan, Master Thesis, 2000.

40. Wu, F.-Y., "The Three-Dimensional Direct Simulation Monte Carlo Method Using Unstructured Adaptive Mesh and Its Applications", Mechanical Engineering, National Chiao-Tung University, Taiwan, Master Thesis, 2002.
41. Lohern, R. and Parikh, P., "Generation of Three-dimensional Unstructured Grids by Advancing Front Method", AIAA paper, 88-0515, 1988.
42. Baker, T., "Unstructured meshes and surface fidelity for complex shapes," In AIAA 10th Computational Fluid Dynamics Conference, 1991.
43. Kannenberg, K. and Boyd, I. D., "Strategies for Efficient Particle Resolution in the Direct Simulation Monte Carlo Method", Journal of Computational Physics, 157, pp. 727-745, 2000.
44. Markelov, G. N. and Ivanov, M. S., "Kinetic Analysis of Hypersonic Laminar Separated Flows for Hollow Cylinder Flare Configurations", AIAA paper, 2000-2223, 2000.
45. Olynick, D. P., *et al.*, "Grid Generation and Adaptation for the Direct Simulation Monte Carlo Method", Journal of Thermophysics Thermophysics and Heat Transfer, 3(4), pp. 368-373, 1989.
46. Teshima, K. and Usami, M., "DSMC Calculation of Supersonic Expansion at a Very Large Pressure Ratio", 22nd Rarefied Gas Dynamics Symposium, Australia, 2000.
47. Wehage, R. A. and Haug, E. J., "Generalized Coordinate Partitioning for Dimension Reduction in Analysis of Constrained Dynamic Systems", ASME Journal of Mechanical Design, 104, pp. 247-255, 1982.
48. Simon, H., "Partitioning of Unstructured Problems for Parallel Processing", Computing Systems in Engineering, 2(2/3), pp. 135-148, 1991.
49. Diniz, P., *et al.*, "Parallel Algorithms for Dynamic Partitioning Unstructured Grids", Proceedings of the 7th SIAM Conference, Parallel Processing for Scientific Computing, In Bailey, D.H., *et al.* (editor), SIAM, 1995.
50. Vanderstraeten, D., *et al.*, "A retrofit Based Methodology for the Fast Generation and Optimization of Large-Scale Mesh Partitions: Beyond the Minimum Interface Size Criterion", Computer Methods in Applied Mechanics and Engineering, 133, pp. 133 25-45, 1996.
51. Barnard, S. T. and Simon, H. D. "A Fast Multilevel Implementation of Recursive Spectral Bisection for Partitioning Unstructured Problems", Concurrence Practice

Experience, 6(2), pp. 101-117, 1994.

52. Karypis, G.. and Kumar, V., "Metis: Unstructured Graph Partitioning and Sparse Matrix Ordering, Version 2.0 User Manual", Minneapolis MN55455, Computer Science Department, University of Minnesota, U.S.A, 1995.

53. Walshaw, C., *et al.*, "Partitioning and Mapping of Unstructured Meshes to Parallel Machine Topologies", Proc. Irregular Parallel Algorithms for Irregularly Structured Problems, 980, pp. 121-126, Ferreira, A. and Rolim, J. of LNCS Springer (editor), 1995.

54. Walshaw, C., "The jostle user manual: Version 2.1, School of Computation & Mathematical Sciences", University of Greenwich, London, SE10 9LS, UK, 1999.

55. Bartel, T. J. and Plimpton, S. J., "DSMC Simulation of Rarefied Gas Dynamics on a Large Hypercube Supercomputer", In 27th AIAA Thermophysics Conference, AIAA Paper, 92-2860, 1992.

56. Walshaw, C., "Parallel Jostle Library Interface: Version 1.2.1, School of Computation & Mathematical Sciences", University of Greenwich, London, SE10 9LS, UK, 2000.

57. Karypis, G., *et al.*, "ParMeTis*, Parallel Graph Partitioning and Sparse Matrix Ordering Library: Version 2.0", University of Minnesota, Department of Computer Science/Army HPC Research Center, Minneapolis, MN 55455.

58. Furlani, T. R. and Lordi, J. A., "Implementation of the Direct Simulation Monte Carlo Method for an Exhaust Plume Flowfield in a Parallel Computing Environment", AIAA Paper, 88-2736, 1988.

59. Matsumoto, Y. and Tokumasu, T., "Parallel Computing of Diatomic Molecular Rarefied Gas Flows", Parallel Computing, 23, pp. 1249-1260, 1997.

60. Nance, R. P., *et al.*, "Parallel Solution of Three-dimensional Flow Over a Finite Flat Plate", AIAA Paper, 94-0219, 1994.

61. Ota, M. and Tanaka, T., "On Speedup of Parallel Processing Using Domain Decomposition Technique for Direct Simulation Monte Carlo Method", The Japan Society of Mechanical Engineering (B), 57(540), pp. 2696-2700, 1991.

62. Ota, M., *et al.*, "Parallel Processings for Direct Simulation Monte Carlo Method", The Japan Society of Mechanical Engineering (B), 61(582), pp. 496-502, 1995.

63. Dietrich, S. and Boyd, I. D., "Scalar and Parallel Optimized Implementation of the Direct Simulation Monte Carlo Method", Journal of Computational Physics, 126, pp. 328-342, 1996.

64. Ivanov, M., *et al.*, “Parallel DSMC strategies for 3D computations”, Proceeding of Parallel CFD'96, Schiano, P., *et al.* (editor), pp. 485-492, North Holland, Amsterdam, 1997.
65. Taylor, S., *et al.*, “The Concurrent Graph: Basic Technology for Irregular Problems”, IEEE Parallel and Distributed Technology, 4(15), pp. 15-25, 1996.
66. Nicol, D. M. and Saltz, J. H., “Dynamic Remapping of Parallel Computations with Varying Resource Demands”, IEEE Transactions on Computers, 37(9), pp. 1073-1087, 1988.
67. LeBeau, G. J., “A Parallel Implementation of the Direct Simulation Monte Carlo Method”, Compute Methods in Applied Mechanics and Engineering, 174, pp. 319-337, 1999.
68. Boyd, I. D., “Conservative Species Weighting Scheme for the Direct Simulation Monte Carlo Method”, Journal of Thermophysics and Heat Transfer, 10(4), pp. 579-585, 1996.
69. Dogra, V. K., *et al.*, “Aerothermodynamics of a 1.6-Meter-Diameter Sphere in Hypersonic Rarefied Flow”, AIAA Journal, 30(7), 1993.
70. Boyd, I. D., “Analysis of vibration-dissociation-recombination processes behind strong shock waves of nitrogen”, Physics of Fluids A, 4(1), pp. 178-185, 1992.
71. Gimelshein, S. F., *et al.*, “On the use of chemical reaction rates with discrete internal energies in the direct simulation Monte Carlo method”, Physics of Fluids, 16(7), pp. 2442-2451, 2004.
72. Borgnakke, C. and Larsen, P. S., “Statistical Collision Model for Monte Carlo Simulation of Polyatomic Gas Mixture”, Journal of Computational Physics, 18, pp. 405-420, 1975.
73. Wu, J.-S., “MuST Visual Preprocessor, Version 1.0”, Mulitiscale Science and Technology Laboratory, Department of Mechanical Engineering, National Chiao-Tung University, Hsinchu, Taiwan, 2005.
74. Lian, Y.-Y., “Parallel Three-Dimensional Direct Simulation Monte Carlo Method and Its Applications”, Mechanical Engineering, National Chiao-Tung University, Taiwan, Master Thesis, 2001.
75. Wu, J.-S. and Hsu Y.-L., “Derivation of Variable Soft Sphere Model Parameters in Direct-Simulation Monte Carlo Method Using Quantum Chemistry Computation”, Japanese Journal of Applied Physics, 42, pp. 7574-7575, 2003.
76. Wu, J.-S., *et al.*, “Pressure Boundary Treatment In Micromechanical Devices

Using Direct Simulation Monte Carlo Method", JSME International Journal, Series B, 44(3), pp.439-450, 2001.

77. Hsiao, W.-C., "Particle Simulation of a Silicon Deposition in LPCVD", Mechanical Engineering, National Chiao-Tung University, Taiwan, Master Thesis, 2001.
78. Koura, K. and Takahira, M., "Monte Carlo Simulation of Hypersonic Rarefied Nitrogen Flow Around a Circular Cylinder", 19th International Symposium on Rarefied Gas Dynamics, pp. 1236-1242, 1994.
79. Bütefisch, K., "Investigation of Hypersonic Non-equilibrium Rarefied Gas Flow Around a Circular Cylinder by the Electron Beam Technique", Rarefied Gas Dynamics II, pp. 1739-1748, Academic Press, New York, 1969.
80. Parker, J. G., "Rotational and vibrational relaxation in diatomic gases", Physics of Fluids, 2, pp. 449-462, 1959.
81. Holden, M. S. and Moselle, J. R., "Theoretical and Experimental Studies of the Shock Wave-Boundary Layer Interaction on Compression Surfaces in Hypersonic Flows", Technical Report 70-0002, ARL.
82. Russell, D. A., "Density Disturbance ahead of a Sphere in Rarefied Supersonic Flow", The Physics of Fluids, 11(8), pp. 1679-1685, 1968.
83. Hypermesh Version 2.0, Altair Computing, Inc., Maplelawn, USA,1757.
84. Yang, T.-J., "The Parallel Implementation of The Direct Simulation Monte Carlo Method Using Unstructured Mesh and Its Applications", Mechanical Engineering, National Chiao-Tung University, Taiwan, Master Thesis, 2000.
85. Boyd, I. D. and Stark, J. P. W., "Direct Simulation of Chemical Reactions", Journal of Thermophysiscs and Heat Transfer, 4(3), pp. 391-393, 1990.
86. Vincenti, W. G. and Kruger, C. H., Introduction to Physical Gas Dynamics, Wiley, New York, 1965.
87. Soga, T., *et al.*, "Experimental Study of Interaction of Underexpanded Free Jets", 14th International Symposium on Rarefied Gas Dynamics, 1, pp. 485-493, 1984.
88. Dagum, L. and Zhu, S. K., "Direct Simulation Monte Carlo Simulation of the Interaction Between Rarefied Free Jets", Journal of Spacecraft and Rockets, 31(6), pp. 960-964, 1994.
89. Dorothy, B. Lee and Winston, D. G., "The aerothermodynamic environment of the Apollo command module during superorbital entry", NASA TND-6792, 1072.

Appendix A
Derivation of the Probability of Dissociation/Exchange

How to derive the reaction probability of the particle method is the most important issue to process the chemical reaction. For the dissociation and exchange reactions, the total collision energy model (TCE) is used in MONACO. To make sure the MONACO uses the TCE model as mentions in Bird's book [11], simple derivation in the following shows the dissociation probability of MONACO is the same as the Eq. (6.10) in Bird's book.

<u>From \PHYS\col_model.c</u> directory

ETA[iclass]=refcxs*2.0/sqrt(PI)*pow((2.0*GASCONST*Trefvhs)/reducedm[iclass],omega)*pow(2.0*GASCONST/reducedm[iclass],(0.5-omega))

$$=\frac{2\sigma}{\sqrt{\pi}\varepsilon}(\frac{2RT_{ref}}{m_r})^{omega}(\frac{2R}{m_r})^{1/2-omega}=\frac{2\sigma}{\sqrt{\pi}\varepsilon}(\frac{2RT_{ref}}{m_r})^{\omega_{12}-1/2}(\frac{2R}{m_r})^{1-\omega_{12}}\quad (omega=\omega_{12}-1/2)$$

if ispec=jspec$\Rightarrow \varepsilon = 2$, if ispec!jspec$\Rightarrow \varepsilon = 1$,

<u>From \PHYS\chem.c</u> directory

ZETArot[iclass] = species[ispec].DOFrot + species[jspec].DOFrot=$\zeta_{r,1}+\zeta_{r,2}$;

ZETAvib[iclass] = species[ispec].DOFvib + species[jspec].DOFvib=$\zeta_{v,1}+\zeta_{v,2}$;

ZETAc[iclass] = DOFrel + ZETArot[iclass] +ZETAvib[iclass]

$= 2(2-omega)+\zeta_{r,1}+\zeta_{r,2}\zeta_{v,1}+\zeta_{v,2} = 2(5/2-\omega_{12})+\zeta_{r,1}+\zeta_{r,2}+\zeta_{v,1}+\zeta_{v,2}$

REAphi2[iclass][ireac] =

ETAc[iclass]/2.0-1.0= $(3/2-\omega_{12})+\overline{\zeta}\quad (\overline{\zeta}=\frac{\zeta_{r,1}+\zeta_{r,2}\zeta_{v,1}+\zeta_{v,2}}{2})$

REAphi1[iclass][ireac] =REAb[iclass][ireac]-0.5+omega+REAphi2[iclass][ireac]-REAphi3[iclass][ireac]=

$b-1/2+\omega_{12}-1/2+(3/2-\omega_{12})+\overline{\zeta}-REAphi3=b+\overline{\zeta}+1/2-REAphi3$

r0 = REAphi2[iclass][ireac]+1.0= $(5/2-\omega_{12})+\overline{\zeta}$

r1 = REAphi1[iclass][ireac]+1.0= $b+\overline{\zeta}+3/2-REAphi3$

r2 = ZETAvib[iclass]/2.0= $(\zeta_{v,1}+\zeta_{v,2})/2$

r3 = r2 + REAphi3[iclass][ireac]= $(\zeta_{v,1}+\zeta_{v,2})/2+REAphi3$

r4 = REAphi1[iclass][ireac]-REAphi2[iclass][ireac]+REAphi3[iclass][ireac]=$b+\omega_{12}-1$

For Dissociation Reaction

REAbeta[iclass][ireac] = mc_gamma(r0)/mc_gamma(r1) *mc_gamma(r2)/mc_gamma(r3) *REAa[iclass][ireac]/ETA[iclass]*pow(GASCONST,-r4)

=

$$\frac{\Gamma(5/2-\omega_{12}+\bar{\zeta})}{\Gamma(b+\bar{\zeta}+3/2-REAphi3)}\cdot\frac{\Gamma((\zeta_{v,1}+\zeta_{v,2})/2)}{\Gamma((\zeta_{v,1}+\zeta_{v,2})/2+REAphi3)}\cdot\frac{a}{\frac{2\sigma}{\sqrt{\pi\varepsilon}}(\frac{2RT_{ref}}{m_r})^{\omega_{12}-1/2}(\frac{2R}{m_r})^{1-\omega_{12}}}\cdot R^{b+\omega_{12}-1}$$

P[ireac] = REAbeta[iclass][ireac]*pow(Ediff,REAphi1[iclass][ireac]) *pow(Ecoll,-REAphi2[iclass][ireac])

=

$$\frac{\Gamma(5/2-\omega_{12}+\bar{\zeta})}{\Gamma(b+\bar{\zeta}+3/2-REAphi3)}\cdot\frac{\Gamma((\zeta_{v,1}+\zeta_{v,2})/2)}{\Gamma((\zeta_{v,1}+\zeta_{v,2})/2+REAphi3)}\cdot\frac{a}{\frac{2\sigma}{\sqrt{\pi\varepsilon}}(\frac{2RT_{ref}}{m_r})^{\omega_{12}-1/2}(\frac{2R}{m_r})^{1-\omega_{12}}}\cdot R^{b+\omega_{12}-1}\times$$

$$(E_{coll}-E_a)^{b+\bar{\zeta}+1/2-REAphi3}\cdot E_{coll}^{-3/2+\omega_{12}-\bar{\zeta}}$$

=

$$\frac{\Gamma(5/2-\omega_{12}+\bar{\zeta})}{\Gamma(b+\bar{\zeta}+3/2-REAphi3)}\cdot\frac{\Gamma((\zeta_{v,1}+\zeta_{v,2})/2)}{\Gamma((\zeta_{v,1}+\zeta_{v,2})/2+REAphi3)}\cdot\frac{a\sqrt{\pi\varepsilon}\,m_r^{1/2}T^{1/2-\omega_{12}}}{2\sqrt{2}\sigma}\cdot R^{1/2-b-\omega_{12}}\cdot$$

$$(E_{coll}-E_a)^{b+\bar{\zeta}+1/2-REAphi3}\cdot E_{coll}^{-3/2+\omega_{12}-\bar{\zeta}}$$

For the *MONACO* code, $m_r=m\times N_o$, $R=k\times N_o$, $E_{coll}=E_c\times N_o$, $E_a=E_a\times N_o$

By comparing with Eq. (6.10) of the Bird's book.

$$P_r(\frac{E_c}{kT})=\frac{\pi^{1/2}\varepsilon\Lambda T_{ref}^{\eta}}{2\sigma_{ref}(kT_{ref})^{\eta-1+\omega_{AB}}}\frac{\Gamma(\bar{\zeta}+5/2-\omega_{AB})}{\Gamma(\bar{\zeta}+\eta+3/2)}(\frac{m_r}{2kT_{ref}})^{1/2}\frac{(E_c-E_a)^{\eta+\bar{\zeta}+1/2}}{E_c^{\bar{\zeta}+3/2-\omega_{AB}}}$$

$$=\frac{\Lambda\sqrt{\pi}\varepsilon m_r^{1/2}T_{ref}^{1/2-\omega_{AB}}}{2\sqrt{2}\sigma_{ref}}\cdot k^{1/2-\eta-\omega_{AB}}\cdot\frac{\Gamma(\bar{\zeta}+5/2-\omega_{AB})}{\Gamma(\bar{\zeta}+\eta+3/2)}\cdot\frac{(E_c-E_a)^{\eta+\bar{\zeta}+1/2}}{E_c^{\bar{\zeta}+3/2-\omega_{AB}}}$$

So, the probability of dissociation uses the Eq. (6.10) of the Bird's book. The probability of exchange also uses the same equation expect the constants of the Arrhenis equation.

Appendix B

Derivation of the Probability of Recombination

A three-body model for recombination reaction is proposed by Professor Boyd. The third body is used to provide the energy to process recombination reaction. The detail of this method is described in Section 6.1.2. The following paragraph is the equation of reaction probability.

From \PHYS\col_model.c directory

ETA[iclass]

=refcxs*2.0/sqrt(PI)*pow((2.0*GASCONST*Trefvhs)/reducedm[iclass],omega)

*pow(2.0*GASCONST/reducedm[iclass],(0.5-omega))

$$\frac{2\sigma}{\sqrt{\pi}\varepsilon}\cdot\left(\frac{2RT_{ref}}{m_r}\right)^{omega}\cdot\left(\frac{2R}{m_r}\right)^{1/2-omega}=\frac{2^{1+omega+1/2-omega}\sigma}{\sqrt{\pi m_r}\varepsilon}\cdot R^{1/2}\cdot T^{omega}$$

=

$$=\frac{2\sqrt{2}\sigma}{\sqrt{\pi m_r}\ \varepsilon}\cdot R^{1/2}\cdot T^{omega}\qquad (omega=\omega_{12}-1/2)$$

if ispec=jspec$\Rightarrow\varepsilon=2$, if ispec!jspec$\Rightarrow\varepsilon=1$,

From \PHYS\chem.c directory

REAphi1[iclass][ireac]=REAb[iclass][ireac]-0.5+omega= $b-1/2+omega=\chi$

r0 = 7.0/2.0+species[ireac].DOFrot/2.0-omega= $7/2+\zeta_1/2-omega$

r1 = r0 + REAphi1[iclass][ireac]= $7/2+\zeta_1/2-omega+\chi$

REAbeta[iclass][ireac]= Wp*mc_gamma(r0)/mc_gamma(r1)*

REAa[iclass][ireac]/ETA[iclass] *

pow(GASCONST,-REAphi1[iclass][ireac])

$$W_p\cdot\frac{\Gamma(\frac{7}{2}+\frac{\zeta_1}{2}-omega)}{\Gamma(\frac{7}{2}+\frac{\zeta_1}{2}-omega+\chi)}\cdot\frac{a}{\frac{2\sqrt{2}\sigma}{\sqrt{\pi m_r}\ \varepsilon}\cdot R^{1/2}\cdot T^{omega}}\cdot R^{-\chi}$$

=

$$=W_p\cdot\frac{\Gamma(\frac{7}{2}+\frac{\zeta_1}{2}-omega)}{\Gamma(\frac{7}{2}+\frac{\zeta_1}{2}-omega+\chi)}\cdot\frac{a\sqrt{\pi m_r}\ \varepsilon}{2\sqrt{2}\sigma\cdot T^{omega}}\cdot R^{-\chi-1/2}$$

Precom=*nobj*volinv * REAbeta[iclass][kspec] *pow((Ecoll+E3),REAphi1[iclass][kspec])

$$*\,nobj*volinv*W_p\cdot\frac{\Gamma(\frac{7}{2}+\frac{\zeta_1}{2}-omega)}{\Gamma(\frac{7}{2}+\frac{\zeta_1}{2}-omega+\chi)}\cdot\frac{a\sqrt{\pi m_r}\,\varepsilon}{2\sqrt{2}\,\sigma\cdot T^{omega}}\cdot R^{-\chi-1/2}\cdot E_c^{\chi}$$

$$=*nobj*volinv*W_p\cdot\frac{\Gamma(\frac{7}{2}+\frac{\zeta_1}{2}-omega)}{\Gamma(\frac{7}{2}+\frac{\zeta_1}{2}-omega+\chi)}\cdot\frac{a\sqrt{\pi m_r}\,\varepsilon\cdot N_o^{1/2}}{2\sqrt{2}\,\sigma\cdot T^{omega}}\cdot(kN_o)^{-\chi-1/2}\cdot E_c^{\chi}\cdot N_o^{\chi}$$

=

$$=*nobj*volinv*W_p\cdot\frac{\Gamma(\frac{7}{2}+\frac{\zeta_1}{2}-omega)}{\Gamma(\frac{7}{2}+\frac{\zeta_1}{2}-omega+\chi)}\cdot\frac{a\sqrt{\pi m_r}\,\varepsilon}{2\sqrt{2}\,\sigma\cdot T^{omega}}\cdot k^{-\chi-1/2}\cdot E_c^{\chi}$$

$$=*nobj*volinv*W_p\cdot\frac{\Gamma(\frac{7}{2}+\frac{\zeta_1}{2}-omega)}{\Gamma(\frac{7}{2}+\frac{\zeta_1}{2}-omega+\chi)}\cdot\frac{a\sqrt{\pi m_r}\,\varepsilon}{2\sqrt{2}\,\sigma\cdot T^{omega}}\cdot\frac{1}{k^{b+omega}}\cdot E_c^{\chi}$$

if ispec=jspec$\Rightarrow\varepsilon=2$, if ispec!jspec$\Rightarrow\varepsilon=1$,

By comparing with Eq. (13) (Recombination probability of binary collision) of the paper of Boyd, Phys.Fluids A 4(1), 1992, pp.178-185.

$$P_r(E_c)=(\frac{1}{Z_r})\cdot E_c^{\chi}=\frac{\sqrt{\pi}}{2^{1/2-w}}\cdot\frac{a_f/a_c}{k^{b_f-b_c}}\cdot\frac{m_r^{1/2-w}}{\sigma_o}\cdot(\frac{m_r}{2(2-w)kT_o})^{w}\cdot\frac{\Gamma(\frac{7}{2}+\frac{\zeta_1}{2}-omega)}{\Gamma(\frac{7}{2}+\frac{\zeta_1}{2}-omega+\chi)}\cdot E_c^{\chi}$$

$$=\frac{\sqrt{\pi m_r}}{\sqrt{2}\sigma_o}\cdot\frac{a_b}{k^{b_b+w}}\cdot(\frac{1}{(2-w)T_o})^{w}\cdot\frac{\Gamma(\frac{7}{2}+\frac{\zeta_1}{2}-omega)}{\Gamma(\frac{7}{2}+\frac{\zeta_1}{2}-omega+\chi)}\cdot E_c^{\chi}$$

Table 1. 1 Comparison of some well-known DSMC codes.
[a] Dynamic Domain Decomposition
[b] Variable Time-Step Scheme
[c] Quantum Vibration Model
[d] Graphic User Interface

Simulator	**Coordinate System**	**Grid System**	**Parallel Computing**	**DDD** [a]	**VTS** [b]	**Chemistry**	**QVM** [c]	**GUI** [d]
Visual DSMC Program	2-D/Axis./3-D	Structured Sub-cells Adaptive	No	No	Yes	Yes	Yes	Yes
DAC	2-D/Axis./3-D	Unstructured 2-level embedded Cartesian	Yes	Yes	-	Yes	-	-
MONACO	2-D/Axis./3-D	Unstructured Sub-cell	Yes	Manual	Yes	Yes	Yes	No
SMILE	2-D/Axis./3-D	Rectangular Adaptive	Yes	Yes	Yes	Yes	Yes	Yes
PDSC	2-D/3-D	Unstructured Adaptive	Yes	Yes	Yes	Yes	Yes	Yes

Table 3. 1 The complete listing of physical and VHS parameters of a hypersonic flow over a cylinder.

N_2 gas	Kn_∞=0.025	n_∞=5.1775E19 (#/m^3)	U_∞= 1823.149 (m/s)
Ma_∞=20	ω=0.74	m_{ref}= 4.65E-26 (Kg)	d_{ref}=4.17E-10 (m)
T_{ref}=273 (K)	T_w=291.6 (K)	T_∞=20 (K)	T_o=1620 (K)
Zr_∞=21	T^*=79.8		

Table 3. 2 The cell numbers of adaptive mesh with or without cell quality control.

Level	0	1		2		3		4	
cell quality control	-	No	Yes	No	Yes	No	Yes	No	Yes
cell no.	7,025	13,916	13,916	33,737	33,773	67,060	67,021	75,305	75,099
$(Kn_c)_{\min}$	0.066	0.104	0.104	0.176	0.171	0.300	0.289	0.314	0.321

Table 3. 3 The complete listing of physical and VHS parameters of a hypersonic flow over a 15^o-compression ramp.

N_2 gas	$Kn=\lambda_\infty/X_c$=2E-4	Re_L=1.04E5	ρ_∞=5.221E-4 (kg/m^3)
p_∞=12.79 (Pa)	Ma_∞=14.36	U_∞= 2652.1 (m/s)	ω=0.75
m_{ref}= 4.65E-26 (Kg)	d_{ref}=4.17E-10 (m)	T_{ref}=273 (K)	T_w=294.4 (K)
T_∞=84.83 (K)	Zr=5		

Table 3. 4 The cell numbers of adaptive mesh with or without cell quality control.

Level	0	1		2	
cell quality control	-	No	Yes	No	Yes
cell no.	15,063	30,219	30,219	83,398	83,776

Table 3. 5 The complete listing of physical and VHS parameters of a hypersonic flow over a sphere.

N_2 gas	$Kn=\lambda_\infty/D$=1.035E-1	Re_L=30	n_∞=9.77e20 (#/m^3)
p_∞=12.79 (Pa)	Ma_∞=4.2	U_∞= 697.022 (m/s)	ω=0.74
m_{ref}= 4.65E-26(Kg)	d_{ref}=4.17E-10 (m)	T_{ref}=273 (K)	T_w=300 (K)
T_o=300 (K)	T_∞=66.25 (K)	Zr=5	

Table 3. 6 The cell numbers of adaptive mesh with or without cell quality control.

Level	0	1		2	
cell quality control	-	No	Yes	No	Yes
cell no.	5,353	22,510	22,510	151,732	164,276

Table 4. 1 Comparisons of the parallel machines at NCHC.

System Hardware Configuration	IBM-SP2	IBM-SMP
CPU	1 per node P2SC –160 -MHz (64 cpu)	4 per node Power3-II 375-MHz (128 cpu)
Memory	256 -MB (per node)	4 -GB (per node)
L1 Cache (per CPU)	128-KB	32KB/64KB

Table 4. 2 Number of cells and particles for three different problem sizes for the driven cavity flow

Problem size	Small	Medium	Large
Cell numbers	11,250	45,000	180,000
Particle numbers	225,000	900,000	3,600,000

Table 6. 1 The constants of the rate coefficients in chem.dat file.

* Rate constant $k = aT^b \exp(-\frac{\theta}{T}) = aT^b \exp(-\frac{E_a}{kT})$

Reactions	a *	b *	E_a *
1. $N_2 + N_2 \rightarrow N + N + N_2$	6.170E-09	-1.6	1.561E-18
2. $N_2 + N \rightarrow N + N + N$	1.850E-08	-1.6	1.561E-18
3. $N + N + N_2 \rightarrow N_2 + N_2$	5.691E-40	-1.6	0
4. $N + N + N \rightarrow N_2 + N$	1.706E-39	-1.6	0
5. $O_2 + O_2 \rightarrow O + O + O_2$	4.580E-11	-1.0	8.197E-19
6. $O_2 + O \rightarrow O + O + O$	1.375E-10	-1.0	8.197E-19
7. $O + O + O_2 \rightarrow O_2 + O_2$	6.305E-44	-0.5	0.0
8. $O + O + O \rightarrow O_2 + O$	1.905E-43	-0.5	0.0
9. $NO + N_2 \rightarrow N + O + N_2$	3.830E-13	-0.5	1.043E-18
10. $NO + O_2 \rightarrow N + O + O_2$	3.830E-13	-0.5	1.043E-18
11. $NO + NO \rightarrow N + O + NO$	3.830E-13	-0.5	1.043E-18
12. $NO + N \rightarrow N + O + N$	7.660E-13	-0.5	1.043E-18
13. $NO + O \rightarrow N + O + O$	7.660E-13	-0.5	1.043E-18
14. $N + O + N_2 \rightarrow NO + N_2$	1.583E-43	-0.5	0.0
15. $N + O + O_2 \rightarrow NO + O_2$	1.583E-43	-0.5	0.0
16. $N + O + NO \rightarrow NO + NO$	1.583E-43	-0.5	0.0
17. $N + O + N \rightarrow NO + N$	3.180E-43	-0.5	0.0
18. $N + O + O \rightarrow NO + O$	3.180E-43	-0.5	0.0
19. $N_2 + O \rightarrow NO + N$	5.300E-17	0.10	5.177E-19
20. $NO + N \rightarrow N_2 + O$	2.020E-17	0.10	0.0
21. $NO + O \rightarrow O_2 + N$	3.600E-22	1.29	2.719E-19
22. $O_2 + N \rightarrow NO + O$	5.200E-22	1.29	4.970E-20

Table 6. 2 Comparison of the reaction probability and rate constant of $N_2 + N_2 \rightarrow N + N + N_2$ reaction

Temp. (^{o}K)	**Total collisions between reactants**	**Number of real reaction**	P_r **(Simulation)**	P_r **(Theoretical)**	k_f **(Simulation)**	k_f **(Theoretical)**
5000	2.008310E+09	1.000000E+01	4.979312E-09	0.290278E-08	1.910565E-24	0.111380E-23
7000	1.005983E+09	1.030000E+02	1.023875E-06	0.999210E-06	4.273378E-22	0.417043E-21
9000	9.993514E+08	2.472600E+04	2.475205E-05	0.227658E-04	1.1000727E-20	0.101180E-19
11000	9.992670E+08	1.687040E+05	1.688278E-04	0.154345E-03	7.889357E-20	0.721257E-19
13000	9.813145E+08	5.963470E+05	6.077022E-04	0.551246E-03	2.960917E-19	0.268585E-18

Table 6. 3 Comparison of the reaction probability and rate constant of $N_2 + N \rightarrow N + N + N$ reaction

Temp. (^{o}K)	**Total collisions between reactants**	**Number of real reaction**	P_r **(Simulation)**	P_r **(Theoretical)**	k_f **(Simulation)**	k_f **(Theoretical)**
5000	6.722381E+08	2.000000E+00	2.975137E-09	0.409370E-08	2.427084E-24	0.333959E-23
7000	6.790187E+08	8.070000E+02	1.188480E-06	0.140915E-05	1.054633E-21	0.125045E-20
9000	5.994706E+08	1.897000E+04	3.164459E-05	0.321058E-04	2.990167E-20	0.303375E-19
11000	5.729034E+08	1.262960E+05	2.204490E-04	0.217667E-03	2.190240E-19	0.216260E-18
13000	5.446063E+08	4.427480E+05	8.129689E-04	0.777403E-03	8.421626E-19	0.805319E-18

Table 6. 4 Comparison of the reaction probability and rate constant of $N + N + N_2 \rightarrow N_2 + N_2$ reaction

Temp. (^oK)	**Total collisions between reactants**	**Number of real reaction**	P_r **(Simulation)**	P_r **(Theoretical)**	k_f **(Simulation)**	k_f **(Theoretical)**
5000	2.653486E+08	6.000000E+01	2.261177E-07	0.235230E-06	6.602036E-46	0.686809E-45
7000	2.659365E+08	2.900000E+01	1.090486E-07	0.116441E-06	3.754451E-46	0.400896E-45
9000	2.625208E+08	2.000000E+01	7.618443e-08	0.678132E-07	3.012677E-46	0.268164E-45
11000	2.573439E+08	1.200000E+01	4.663021E-08	0.437044E-07	2.075406E-46	0.194518E-45
13000	5.382297 E+08	2.600000E+01	4.830651e-08	0.301891E-07	2.382503E-46	0.148895E-45

Table 6. 5 Comparison of the reaction probability and rate constant of $N + N + N \rightarrow N_2 + N$ reaction

Temp. (^oK)	**Total collisions between reactants**	**Number of real reaction**	P_r **(Simulation)**	P_r **(Theoretical)**	k_f **(Simulation)**	k_f **(Theoretical)**
3000	1.058814E+09	1.606000E+03	1.516791E-06	0.141031E-05	2.214313E-45	0.205886E-44
4000	1.064403E+09	7.730000E+02	7.262284E-07	0.698113E-06	1.250171E-45	0.120177E-44
5000	1.050878E+09	4.980000E+02	4.738893E-07	0.406569E-06	9.369865E-46	0.803879E-45
6000	1.031560E+09	3.370000E+02	3.266960E-07	0.262026E-06	7.2702517E-46	0.583110E-45
7000	1.011896E+09	2.590000E+02	2.559552E-07	0.180997E-06	6.311923E-46	0.446344E-45

Table 6. 6 Comparison of the reaction probability and rate constant of $O_2 + O_2 \rightarrow O + O + O_2$ reaction

Temp. (^oK)	**Total collisions between reactants**	**Number of real reaction**	P_r **(Simulation)**	P_r **(Theoretical)**	k_f **(Simulation)**	k_f **(Theoretical)**
3000	909944997	224	2.461687E-07	0.127783E-06	7.407696E-23	0.384524E-22
4000	812138012	14664	1.805605E-05	0.125894E-04	5.838560E-21	0.407089E-20
5000	902565292	211049	2.338324E-04	0.185648E-03	7.994987E-20	0.634750E-19
6000	876586710	1116620	1.273827E-03	0.107052E-02	4.558466E-19	0.383093E-18
7000	846974877	3550774	4.192302E-03	0.363173E-02	1.559184E-18	0.135070E-17

Table 6. 7 Comparison of the reaction probability and rate constant of $O_2 + O \rightarrow O + O + O$ reaction

Temp. (^{o}K)	**Total collisions between reactants**	**Number of real reaction**	P_r **(Simulation)**	P_r **(Theoretical)**	k_f **(Simulation)**	k_f **(Theoretical)**
3000	433257607	97	2.238853E-07	0.201857E-06	1.280387E-22	0.115441E-21
4000	437929163	9191	2.098741E-05	0.198873E-04	1.289762E-20	0.122216E-19
5000	429199928	139810	3.257456E-04	0.293266E-03	2.116688E-19	0.190564E-18
6000	417020964	790388	1.895320E-03	0.169109E-02	1.289010E-18	0.115011E-17
7000	402801642	2608729	6.476461E-03	0.573701E-02	4.577705E-18	0.405505E-17

Table 6. 8 Comparison of the reaction probability and rate constant of $O + O + O_2 \rightarrow O_2 + O_2$ reaction

Temp. (^oK)	**Total collisions between reactants**	**Number of real reaction**	P_r **(Simulation)**	P_r **(Theoretical)**	k_f **(Simulation)**	k_f **(Theoretical)**
3000	186447756	148	7.937880E-07	0.643972E-06	1.419123E-045	0.115128E-44
4000	188588480	120	6.363061E-07	0.488072E-06	1.299858E-045	0.997041E-45
5000	185131710	91	4.915419E-07	0.383242E-06	1.143789E-045	0.891781E-45
6000	179556576	67	3.731414E-07	0.309784E-06	9.805790E-046	0.814081E-45
7000	173254226	51	2.943651E-07	0.256480E-06	8.650219E-046	0.753692E-45

Table 6. 9 Comparison of the reaction probability and rate constant of $O + O + O \rightarrow O_2 + O$ reaction

Temp. (^{o}K)	**Total collisions between reactants**	**Number of real reaction**	P_r **(Simulation)**	P_r **(Theoretical)**	k_f **(Simulation)**	k_f **(Theoretical)**
3000	746437311	3657	4.899273E-06	0.450826E-05	3.781211E-45	0.347944E-44
4000	755242580	2791	3.695501E-06	0.341685E-05	3.259020E-45	0.301328E-44
5000	741177555	2148	2.898091E-06	0.268297E-05	2.911261E-45	0.269516E-44
6000	718982844	1703	2.368624E-06	0.216870E-05	2.687134E-45	0.246033E-44
7000	694873458	1335	1.921213E-06	0.179554E-05	2.437256E-45	0.227783E-44

Table 6. 10 Comparison of the reaction probability and rate constant of $NO + N_2 \rightarrow N + O + N_2$ reaction

Temp. (^{o}K)	**Total collisions between reactants**	**Number of real reaction**	P_r **(Simulation)**	P_r **(Theoretical)**	k_f **(Simulation)**	k_f **(Theoretical)**
5000	249626105	464	1.858780E-06	0.194162E-05	1.412561E-21	0.147551E-20
7000	250173393	28670	1.146005E-04	0.113297E-03	9.473233E-20	0.936544E-19
9000	247166816	271610	1.098893E-03	0.103368E-02	9.672819E-19	0.909877E-18
11000	242704757	1077996	4.441594E-03	0.409384E-02	4.110775E-18	0.378892E-17
13000	238343701	2719260	1.140899E-02	0.103942E-01	1.100956E-17	0.100303E-16

Table 6. 11 Comparison of the reaction probability and rate constant of $NO + O_2 \rightarrow N + O + O_2$ reaction

Temp. (^{o}K)	**Total collisions between reactants**	**Number of real reaction**	P_r **(Simulation)**	P_r **(Theoretical)**	k_f **(Simulation)**	k_f **(Theoretical)**
5000	235424229	628	2.667525E-06	0.205645E-05	1.913965E-21	0.147551E-20
7000	236615349	38244	1.616294E-04	0.119997E-03	1.261472E-19	0.936543E-19
9000	233134814	343420	1.473053E-03	0.109481E-02	1.224226E-18	0.909877E-18
11000	228967220	1315563	5.745639E-03	0.433596E-02	5.020758E-18	0.378892E-17
13000	224779133	3221065	1.432991E-02	0.110090E-01	1.305607E-17	0.100303E-16

Table 6. 12 Comparison of the reaction probability and rate constant of $NO + NO \rightarrow N + O + NO$ reaction

Temp. (^{o}K)	**Total collisions between reactants**	**Number of real reaction**	P_r **(Simulation)**	P_r **(Theoretical)**	k_f **(Simulation)**	k_f **(Theoretical)**
5000	493274083	1872	3.795050E-06	0.389306E-05	1.427100E-021	0.146396E-20
7000	494547920	115934	2.344242E-04	0.227166E-03	9.588958E-020	0.929208E-19
9000	488493032	1070455	2.191341E-03	0.207259E-02	9.544769E-019	0.902750E-18
11000	479800926	4259445	8.877525E-03	0.820839E-02	4.065693E-018	0.375925E-17
13000	470268366	10667725	2.268434E-02	0.208411E-01	1.083194E-017	0.995177E-17

Table 6. 13 Comparison of the reaction probability and rate constant of $NO + N \rightarrow N + O + N$ reaction

Temp. (^{o}K)	**Total collisions between reactants**	**Number of real reaction**	P_r **(Simulation)**	P_r **(Theoretical)**	k_f **(Simulation)**	k_f **(Theoretical)**
5000	266760622	980	3.673706E-06	0.363018E-05	2.986411E-21	0.295103E-20
7000	270420558	59722	2.208486E-04	0.211827E-03	1.952863E-19	0.187309E-18
9000	264623436	548760	2.073739E-03	0.193263E-02	1.952618E-18	0.181975E-17
11000	259673553	2150813	8.282757E-03	0.765412E-02	8.200219E-18	0.757785E-17
13000	254694166	5391097	2.116694E-02	0.194338E-01	2.184979E-17	0.200607E-16

Table 6. 14 Comparison of the reaction probability and rate constant of $NO + O \rightarrow N + O + O$ reaction

Temp. (^{o}K)	**Total collisions between reactants**	**Number of real reaction**	P_r **(Simulation)**	P_r **(Theoretical)**	k_f **(Simulation)**	k_f **(Theoretical)**
5000	223358315	978	4.378614E-06	0.433327E-05	2.981911E-021	0.295103E-20
7000	224668576	58772	2.615942E-04	0.252853E-03	1.937843E-019	0.187309E-18
9000	221564005	552839	2.495166E-03	0.230694E-02	1.968229E-018	0.181975E-17
11000	217709070	2156316	9.904576E-03	0.913655E-02	8.214855E-018	0.757785E-17
13000	213956988	5394345	2.521229E-02	0.231977E-01	2.180288E-017	0.200607E-16

Table 6. 15 Comparison of the reaction probability and rate constant of $N + O + N_2 \rightarrow NO + N_2$ reaction

Temp. (^oK)	**Total collisions between reactants**	**Number of real reaction**	P_r **(Simulation)**	P_r **(Theoretical)**	k_f **(Simulation)**	k_f **(Theoretical)**
5000	99122344	40	4.035417E-07	0.304911E-06	2.963733E-045	0.223936E-44
7000	99329520	24	2.416200E-07	0.218537E-06	2.092509E-045	0.189261E-44
9000	98280965	17	1.729735E-07	0.167800E-06	1.720584E-045	0.166912E-44
11000	96350646	10	1.037876E-07	0.134867E-06	1.161857E-045	0.150978E-44
13000	94522664	14	1.481126E-07	0.111954E-06	1.837347E-045	0.138879E-44

Table 6. 16 Comparison of the reaction probability and rate constant of $N + O + O_2 \rightarrow NO + O_2$ reaction

Temp. (^oK)	**Total collisions between reactants**	**Number of real reaction**	P_r **(Simulation)**	P_r **(Theoretical)**	k_f **(Simulation)**	k_f **(Theoretical)**
5000	99223776	32	3.225034E-07	0.304911E-06	2.368562E-045	0.223936E-44
7000	99226239	25	2.519495E-07	0.218537E-06	2.181966E-045	0.189261E-44
9000	98226610	18	1.832497E-07	0.167800E-06	1.822802E-045	0.166912E-44
11000	96373596	15	1.556443E-07	0.134867E-06	1.742370E-045	0.150978E-44
13000	94694018	13	1.372843E-07	0.111954E-06	1.703021E-045	0.138879E-44

Table 6. 17 Comparison of the reaction probability and rate constant of $N + O + NO \rightarrow NO + NO$ reaction

Temp. (^{o}K)	**Total collisions between reactants**	**Number of real reaction**	P_r **(Simulation)**	P_r **(Theoretical)**	k_f **(Simulation)**	k_f **(Theoretical)**
5000	99073342	37	3.734607E-07	0.304911E-06	2.742809E-045	0.223936E-44
7000	99355778	21	2.113616E-07	0.218537E-06	1.830461E-045	0.189261E-44
9000	98045441	24	2.447844E-07	0.167800E-06	2.434893E-045	0.166912E-44
11000	96465977	17	1.762279E-07	0.134867E-06	1.972795E-045	0.150978E-44
13000	94639816	15	1.584957E-07	0.111954E-06	1.966150E-045	0.138879E-44

Table 6. 18 Comparison of the reaction probability and rate constant of $N + O + N \rightarrow NO + N$ reaction

Temp. (^{o}K)	**Total collisions between reactants**	**Number of real reaction**	P_r **(Simulation)**	P_r **(Theoretical)**	k_f **(Simulation)**	k_f **(Theoretical)**
5000	222368524	225	1.011834E-06	0.914497E-06	2.476983E-045	0.223870E-44
7000	223682461	170	7.600059E-07	0.655444E-06	2.193882E-045	0.189205E-44
9000	220237391	124	5.630288E-07	0.503272E-06	1.866755E-045	0.166863E-44
11000	216670853	122	5.630661E-07	0.404463E-06	2.101188E-045	0.150933E-44
13000	212511245	74	3.482169E-07	0.335746E-06	1.439954E-045	0.138838E-44

Table 6. 19 Comparison of the reaction probability and rate constant of $N + O + O \rightarrow NO + O$ reaction

Temp. (^{o}K)	**Total collisions between reactants**	**Number of real reaction**	P_r **(Simulation)**	P_r **(Theoretical)**	k_f **(Simulation)**	k_f **(Theoretical)**
5000	222535385	201	9.032271E-07	0.914497E-06	2.211112E-045	0.223870E-44
7000	223472402	185	8.278427E-07	0.655444E-06	2.389704E-045	0.189205E-44
9000	220318540	123	5.582826E-07	0.503272E-06	1.851018E-045	0.166863E-44
11000	216464094	117	5.405054E-07	0.404463E-06	2.016999E-045	0.150933E-44
13000	212461834	65	3.059373E-07	0.335746E-06	1.265119E-045	0.138838E-44

Table 6. 20 Comparison of the reaction probability and rate constant of $N_2 + O \rightarrow NO + N$ reaction

Temp. (^{o}K)	**Total collisions between reactants**	**Number of real reaction**	P_r **(Simulation)**	P_r **(Theoretical)**	k_f **(Simulation)**	k_f **(Theoretical)**
5000	448696187	43293	9.648622E-05	0.100183E-03	6.597438E-20	0.685024E-19
7000	450814795	374796	8.313747E-04	0.812581E-03	6.183565E-19	0.604378E-18
9000	444335334	1205511	2.713066E-03	0.257461E-02	2.148765E-18	0.203911E-17
11000	436669803	2480978	5.681588E-03	0.533062E-02	4.731359E-18	0.443909E-17
13000	428279387	4039183	9.431187E-03	0.878537E-02	8.188798E-18	0.762805E-17

Table 6. 21 Comparison of the reaction probability and rate constant of $NO + N \rightarrow N_2 + O$ reaction

Temp. (^{o}K)	**Total collisions between reactants**	**Number of real reaction**	P_r **(Simulation)**	P_r **(Theoretical)**	k_f **(Simulation)**	k_f **(Theoretical)**
5000	532736242	33141438	6.220984E-02	0.582376E-01	5.057131E-17	0.473422E-16
7000	534468471	31479363	5.889845E-02	0.553712E-01	5.208119E-17	0.489622E-16
9000	528640226	29895721	5.655211E-02	0.533227E-01	5.324907E-17	0.502083E-16
11000	519066616	28442223	5.479494E-02	0.517416E-01	5.424891E-17	0.512260E-16
13000	509389746	27193371	5.338421E-02	0.504612E-01	5.510638E-17	0.520890E-16

Table 6. 22 Comparison of the reaction probability and rate constant of $NO + O \rightarrow O_2 + N$ reaction

Temp. (^oK)	**Total collisions between reactants**	**Number of real reaction**	P_r **(Simulation)**	P_r **(Theoretical)**	k_f **(Simulation)**	k_f **(Theoretical)**
5000	446187472	264810	5.934950E-04	0.607356E-03	4.041802E-19	0.413620E-18
7000	448940904	1222346	2.722732E-03	0.265693E-02	2.016951E-18	0.196820E-17
9000	442632876	2987210	6.748731E-03	0.644966E-02	5.323514E-18	0.508760E-17
11000	435020516	5452565	1.253404E-02	0.118315E-01	1.039573E-17	0.981302E-17
13000	426429436	8435851	1.978252E-02	0.185425E-01	1.710737E-17	0.160350E-16

Table 6. 23 Comparison of the reaction probability and rate constant of $O_2 + N \rightarrow NO + O$ reaction

Temp. (^{o}K)	**Total collisions between reactants**	**Number of real reaction**	P_r **(Simulation)**	P_r **(Theoretical)**	k_f **(Simulation)**	k_f **(Theoretical)**
5000	446187472	264810	5.934950E-04	0.607356E-03	4.041802E-19	0.413620E-18
7000	448940904	1222346	2.722732E-03	0.265693E-02	2.016951E-18	0.196820E-17
9000	442632876	2987210	6.748731E-03	0.644966E-02	5.323514E-18	0.508760E-17
11000	435020516	5452565	1.253404E-02	0.118315E-01	1.039573E-17	0.981302E-17
13000	426429436	8435851	1.978252E-02	0.185425E-01	1.710737E-17	0.160350E-16

Table 6. 24 Mole percentage of each species by using original chem.dat.

Temp. (°K)	**N_2 (Original)**	**N_2 (Fitting)**	**O_2 (Original)**	**O_2 (Fitting)**	**NO (Original)**	**NO (Fitting)**	**N (Original)**	**N (Fitting)**	**O (Original)**	**O (Fitting)**
2000	80	80	20	20	0	0	0	0	0	0
3000	78.73293	79.30586	19.04047	19.49159	0.5480151	0.2640975	0.0005548	5.9726E-5	1.678009	0.9383817
4000	58.51967	58.68676	0.2123696	0.2220255	0.4692463	0.938484	1.096412	0.5560535	39.70229	39.59669
6000	25.41407	36.97584	0.0039259	0.0063654	0.1099764	0.2671703	38.56771	25.1045	35.90432	37.64613
8000	0.24792	0.2827485	0.0001338	3.5631E-5	0.0020865	0.0023249	70.51454	70.89766	29.23532	28.81723
10000	0.0164276	0.0224934	2.6668E-5	4.4456E-6	0.0005067	0.0005867	71.36262	71.67037	28.62042	28.30656
12000	0.0039995	0.0045668	9.9989E-6	6.6664E-6	0.0002100	0.0001900	71.63901	71.78274	28.35678	28.2125
14000	0.0016150	0.0016265	5.0041E-6	6.6657E-6	9.4986E-5	5.9992E-5	71.94937	72.16114	28.04892	27.83717
16000	0.0006602	0.0006750	0	1.6665E-6	2.0007E-5	4.5000E-5	72.29124	72.49075	27.70809	27.50853

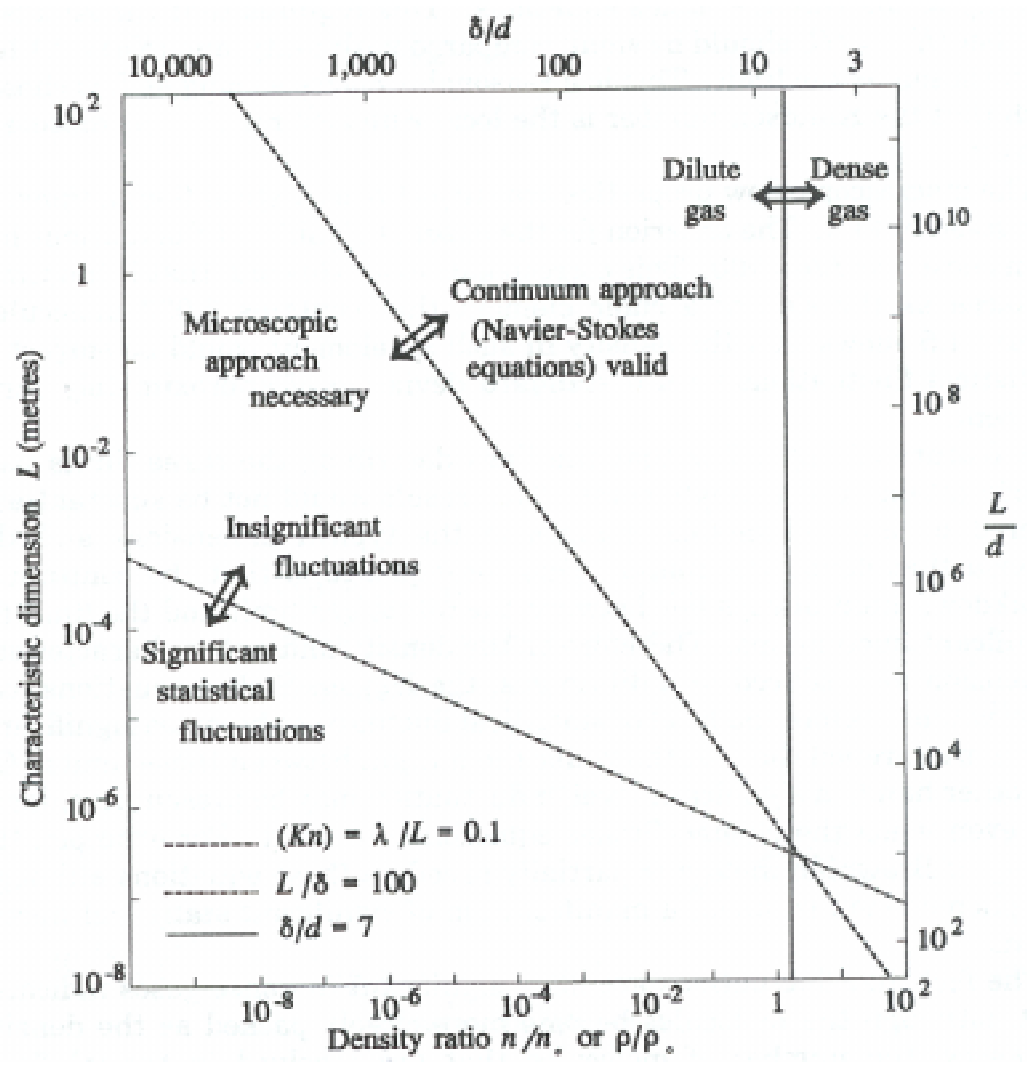


Figure 1. 1 Effective limits of major approximations in the DSMC method [10, 11].

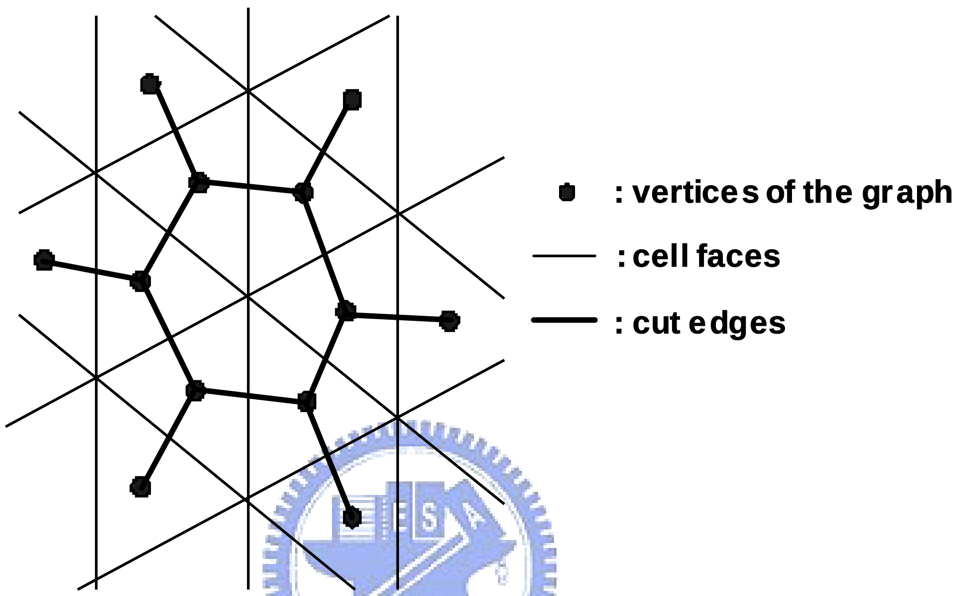


Figure 1. 2 Sketch of graph and mesh.

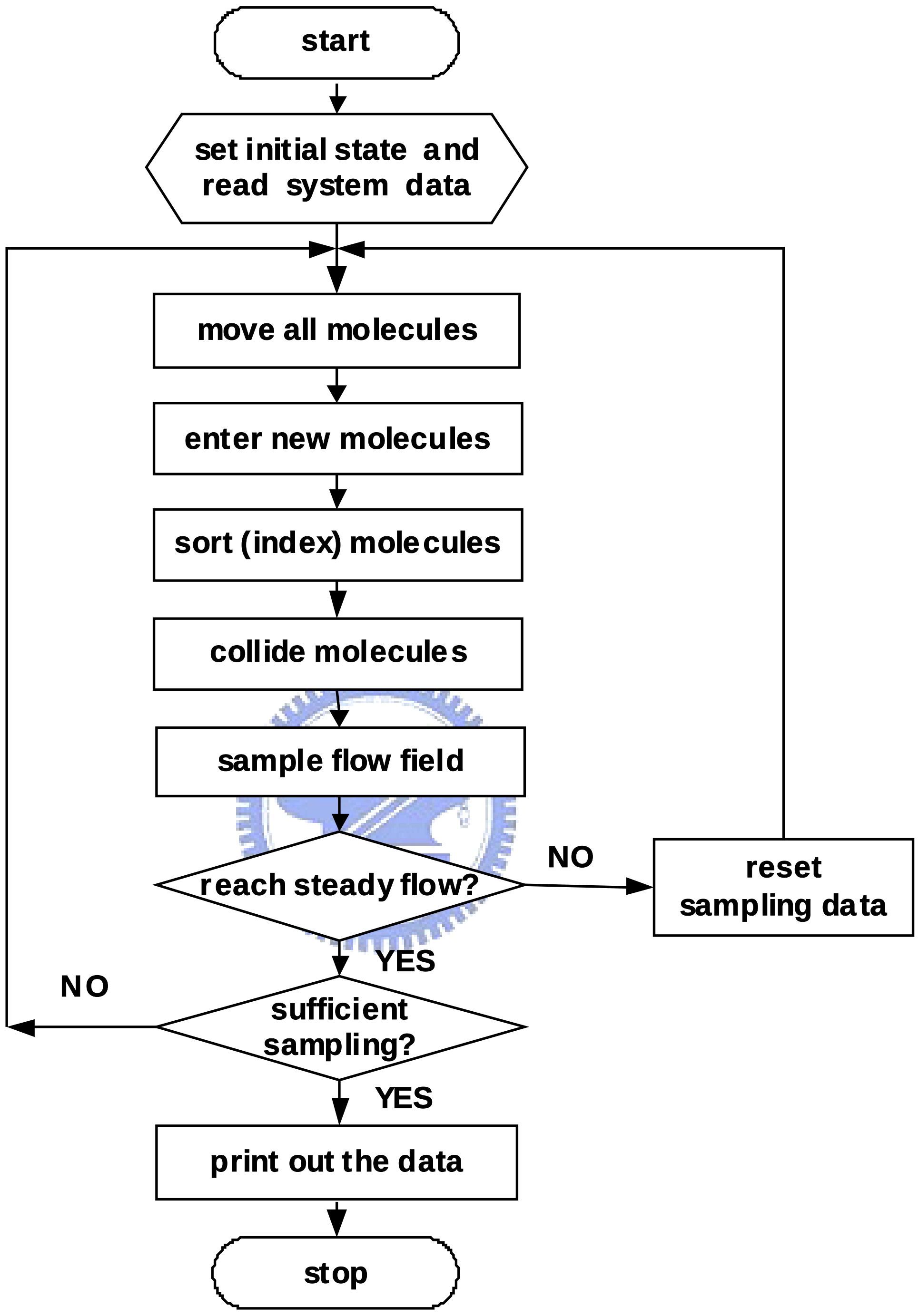


Figure 2. 1 The flowchart of the standard DSMC method.

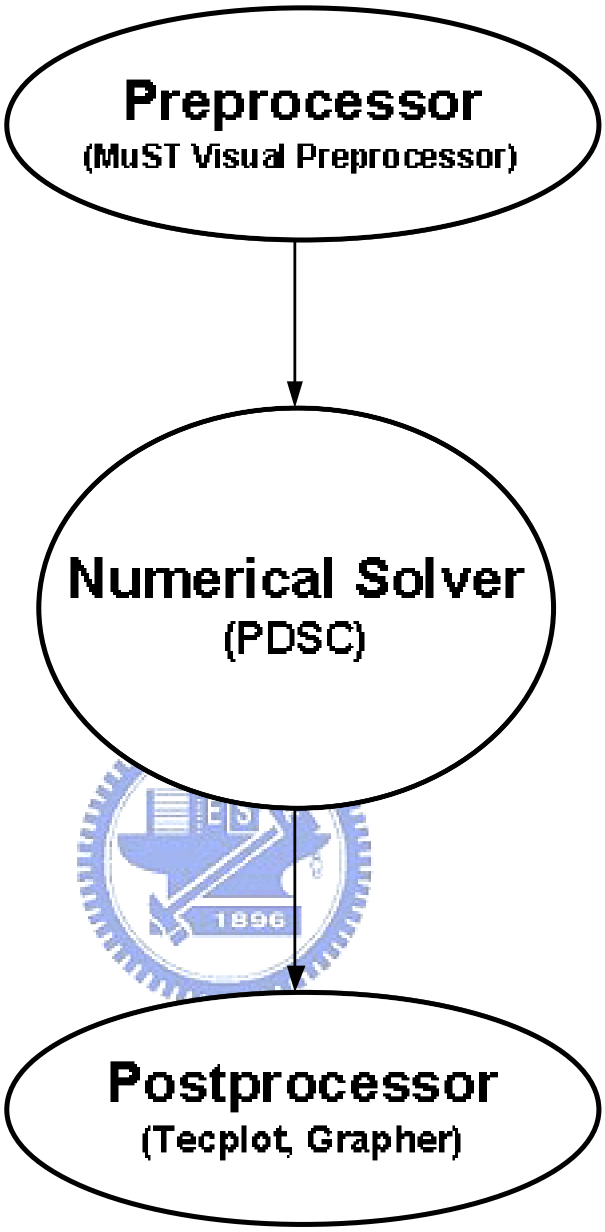
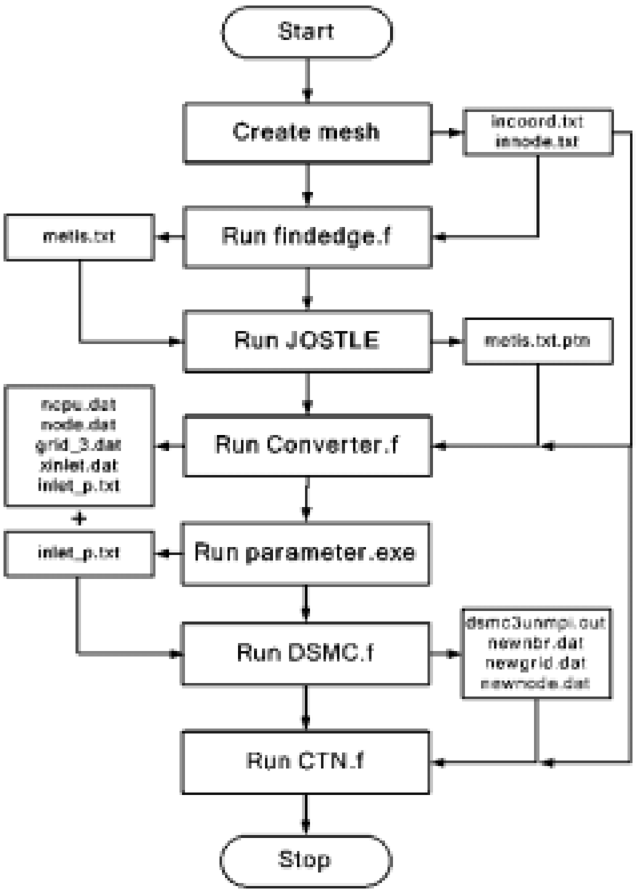


Figure 2. 2 Schematic diagram of the PDSC.



(a)

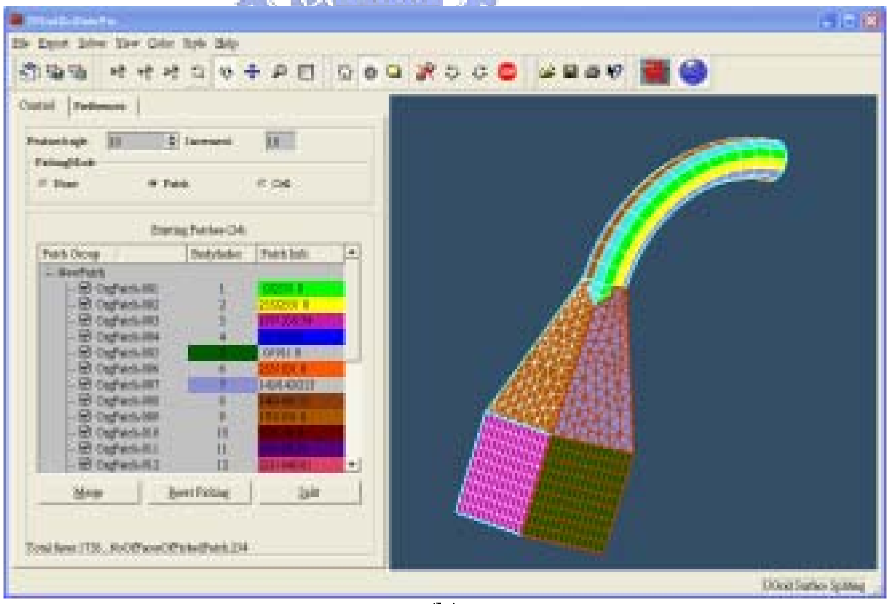
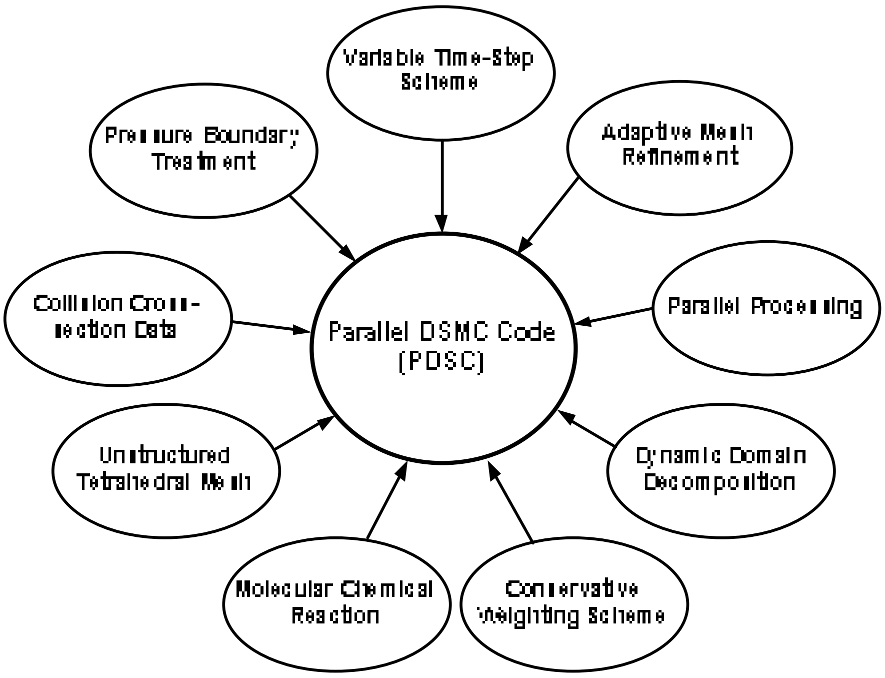
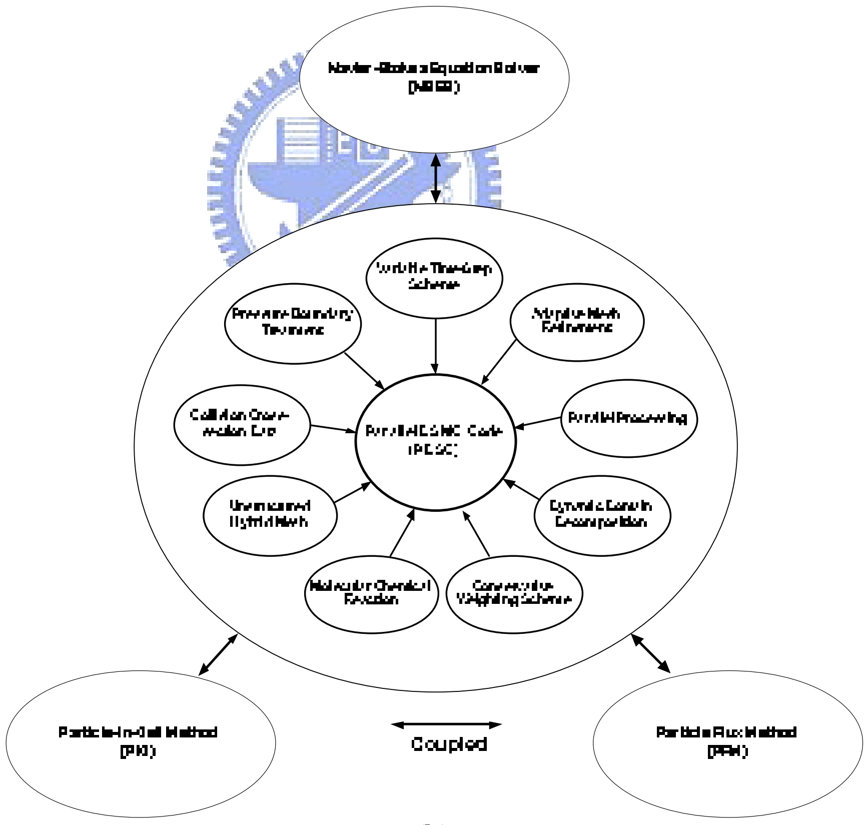

(b)

Figure 2. 3 MuST Visual Preprocessor for numerical simulations [73].


Variable Time-Step Scheme
Pressure Boundary Treatment
Adaptive Mesh Refinement
Collision Cross-section Data
Parallel DSMC Code (PDSC)
Parallel Processing
Unstructured Tetrahedral Mesh
Dynamic Domain Decomposition
Molecular Chemical Reaction
Conservative Weighting Scheme

(a)


Navier-Stokes Equation Solver (NSES)
Variable Time-Step Scheme
Pressure Boundary Treatment
Adaptive Mesh Refinement
Collision Cross-section Data
Parallel DSMC Code (PDSC)
Parallel Processing
Unstructured Hybrid Mesh
Dynamic Domain Decomposition
Molecular Chemical Reaction
Conservative Weighting Scheme
Particle-In-Cell Method (PIC)
Coupled
Particle Flux Method (PFM)

(b)

Figure 2. 4 Important features of the PDSC (a) future PDSC; (b) planned study of PDSC.

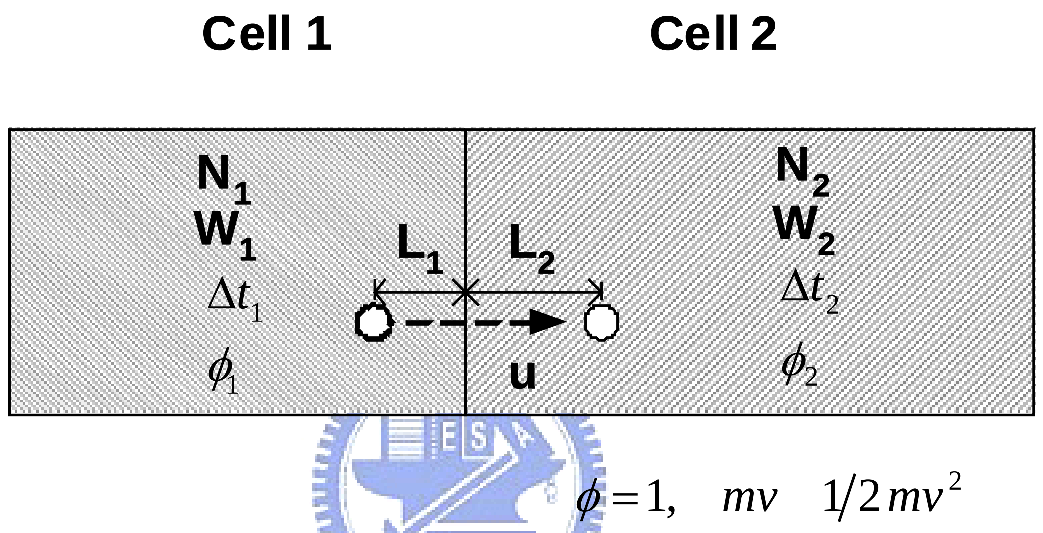


Figure 3. 1 Sketch of the concept of variable-time-step scheme.

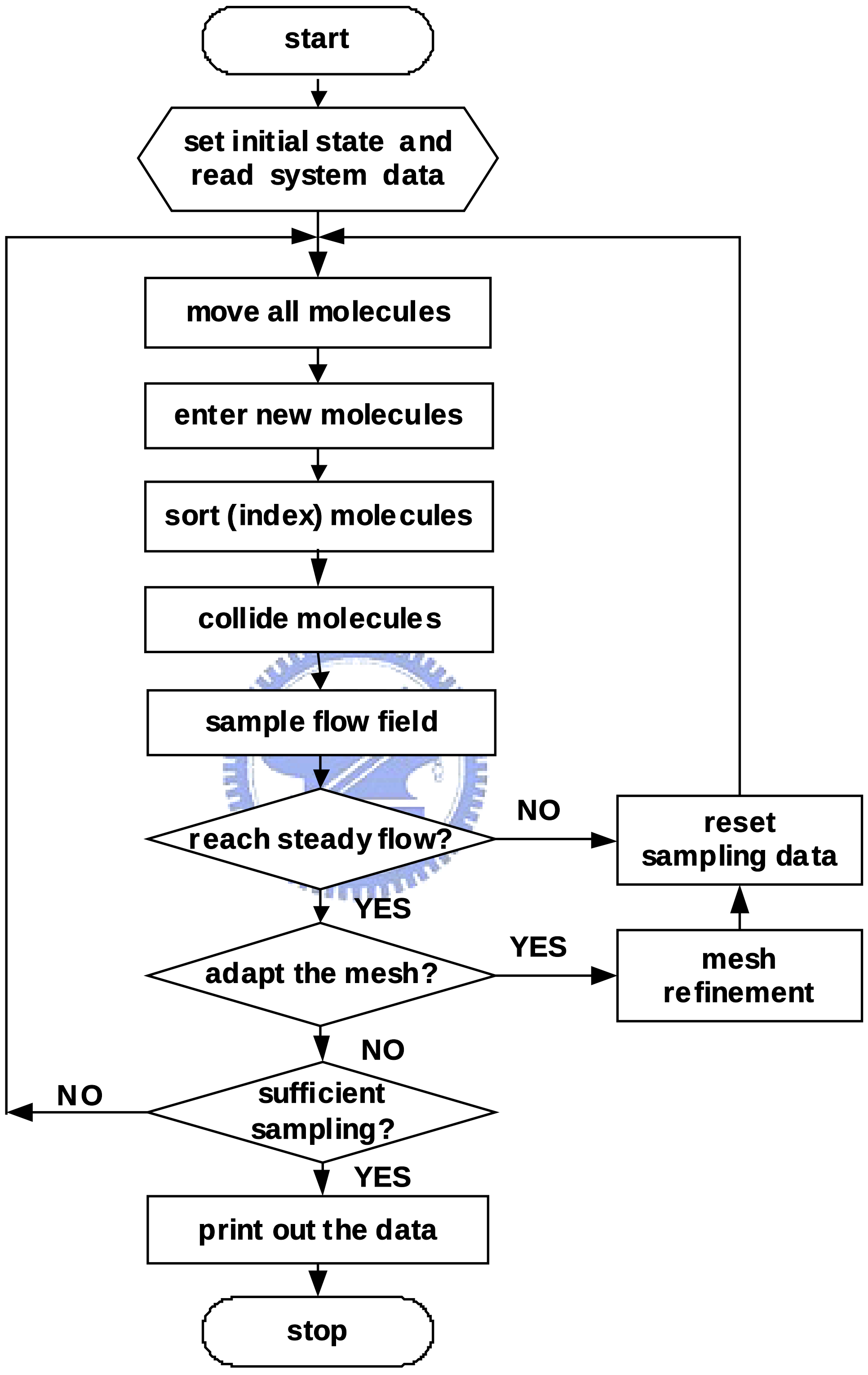


Figure 3. 2 The flowchart of the DSMC method with mesh adaptation.

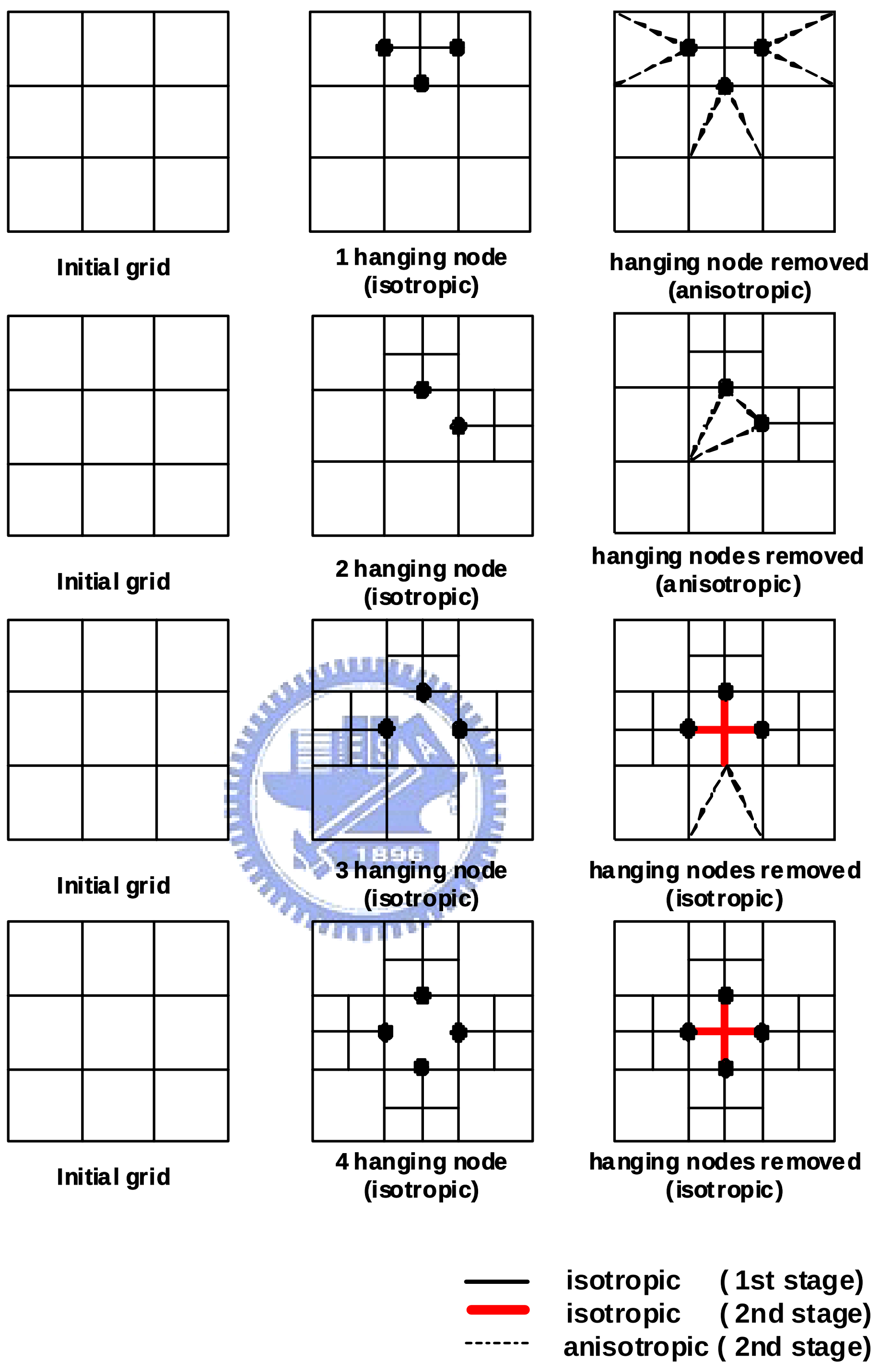


Figure 3. 3 Mesh refinement scheme for unstructured quadrilateral mesh [39].

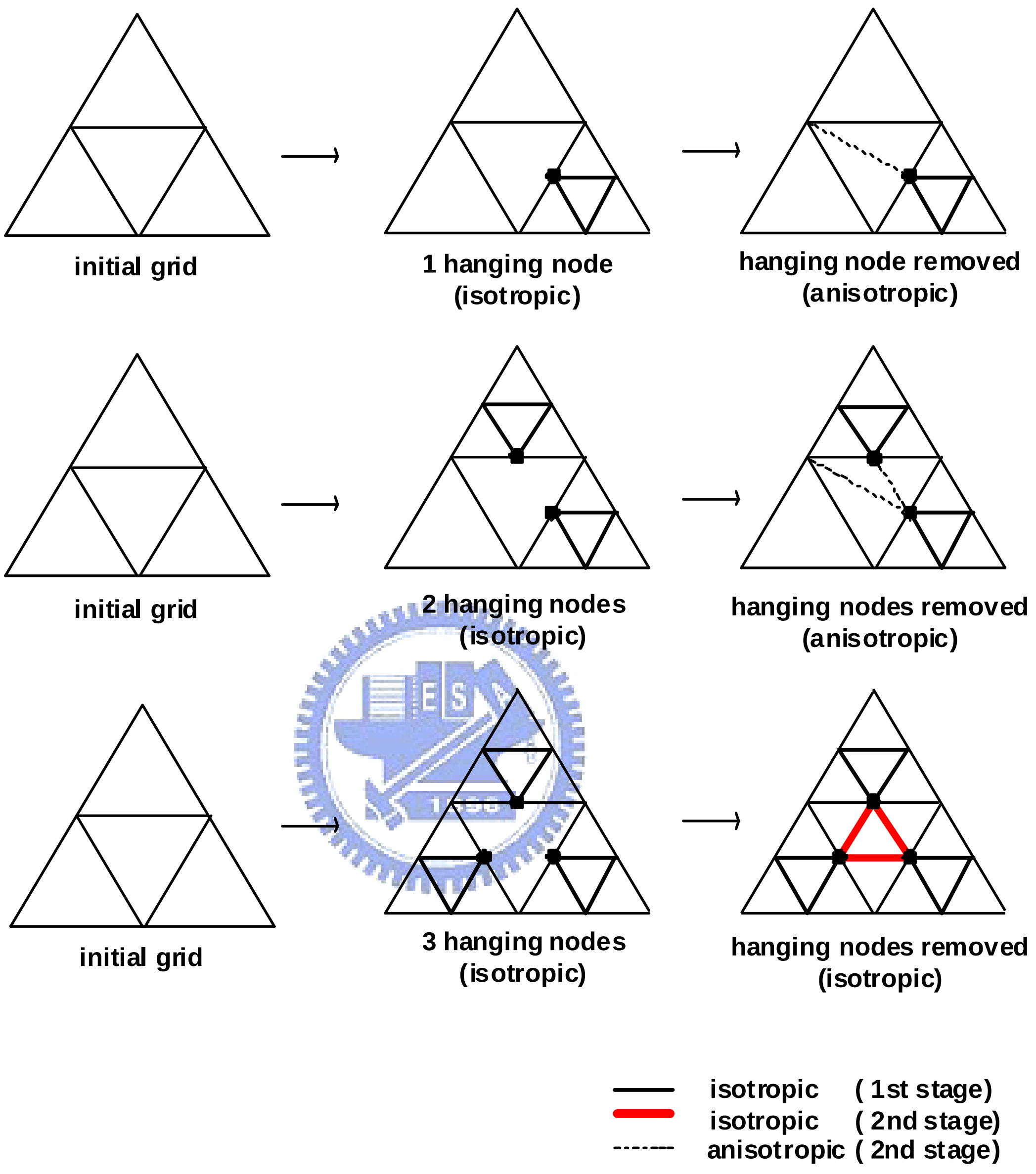


Figure 3. 4 Mesh refinement scheme for unstructured triangular mesh [39].

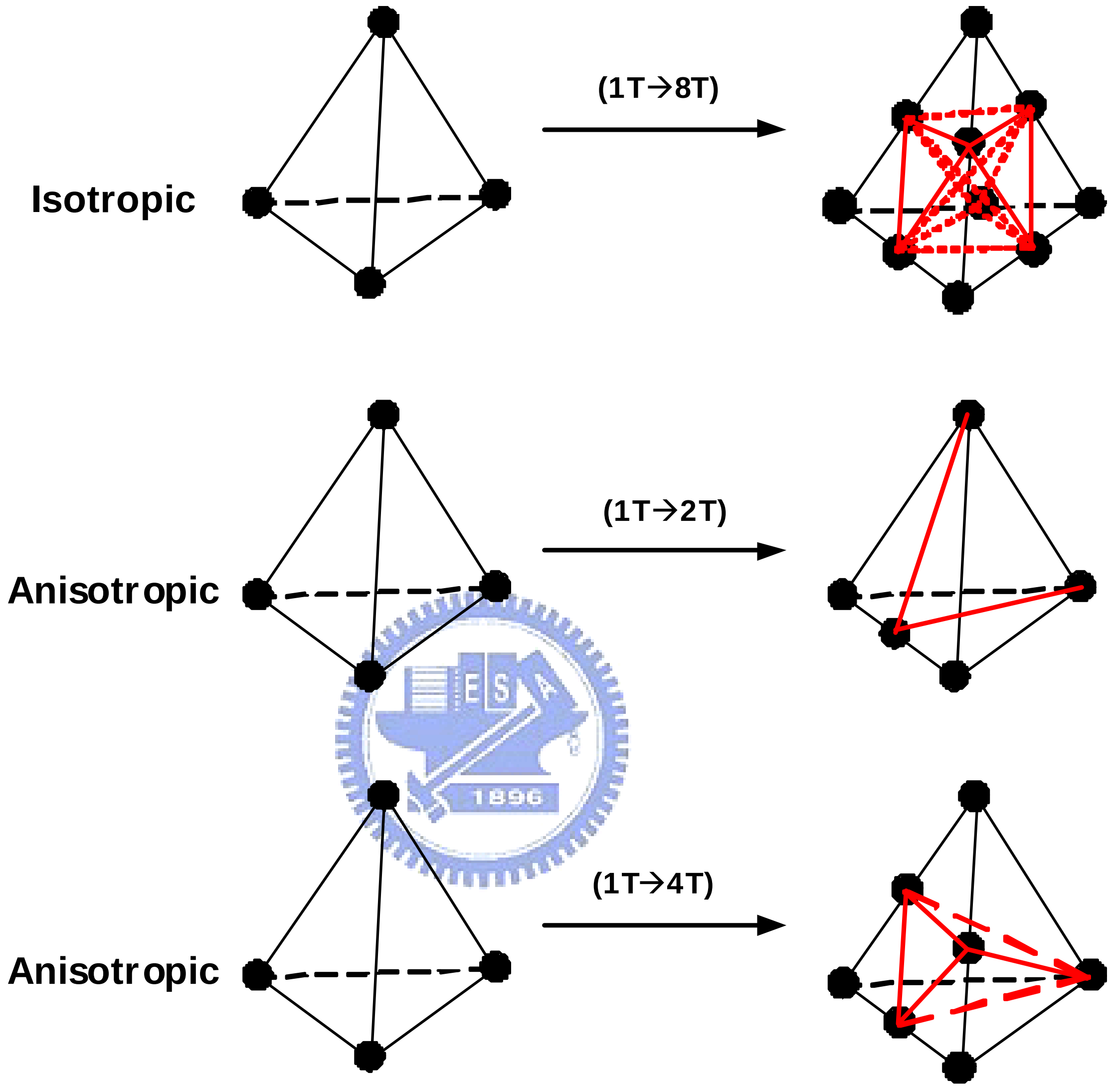


Figure 3. 5 The diagram of mesh adaptation of tetrahedral cell [40].

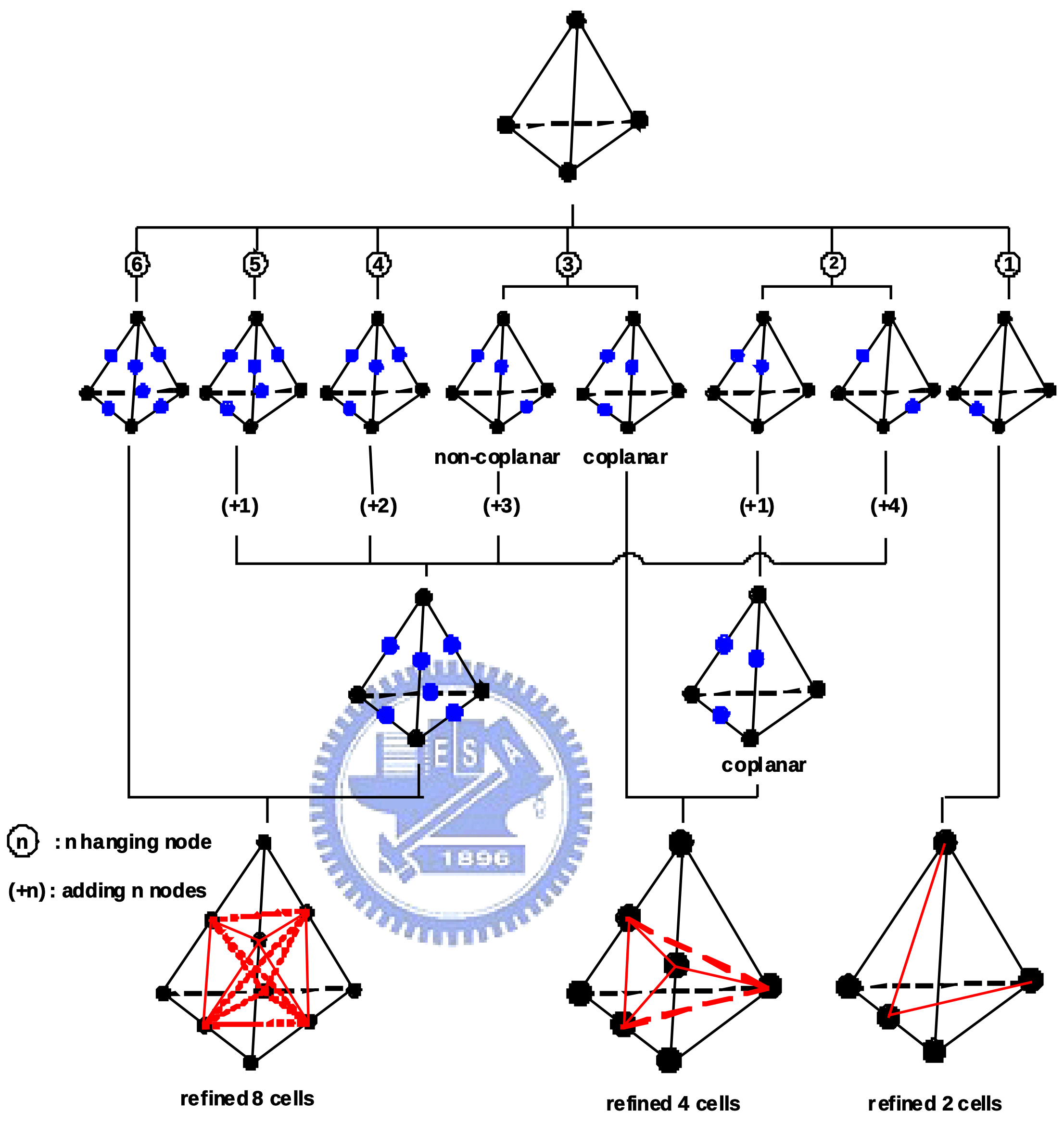


Figure 3. 6 The tree diagram of removing hanging nodes of tetrahedral mesh adaptation [40].

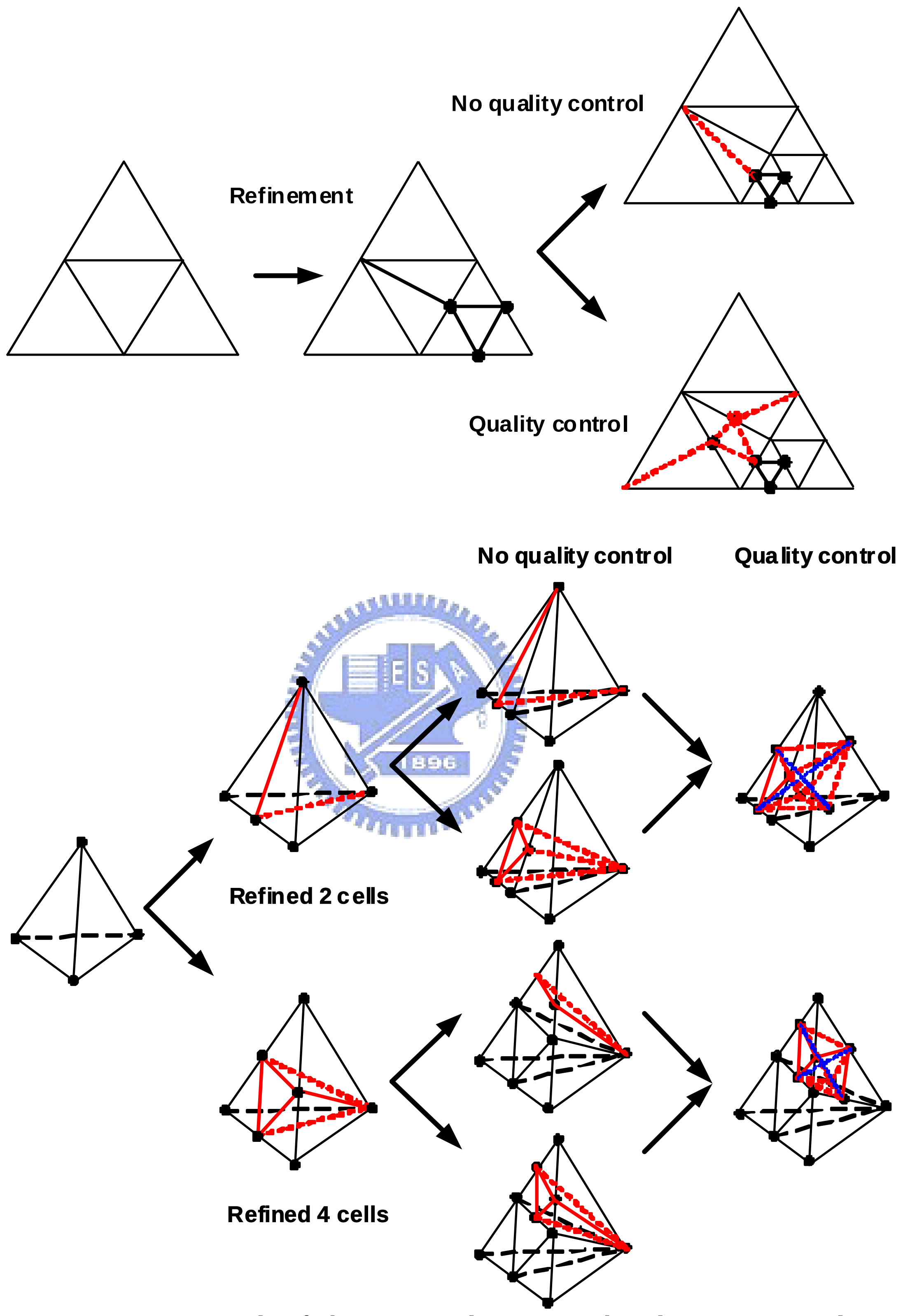


Figure 3. 7 Removal of hanging nodes in mesh adaptation procedures for two-dimensional triangular and three-dimensional tetrahedral meshes [40].

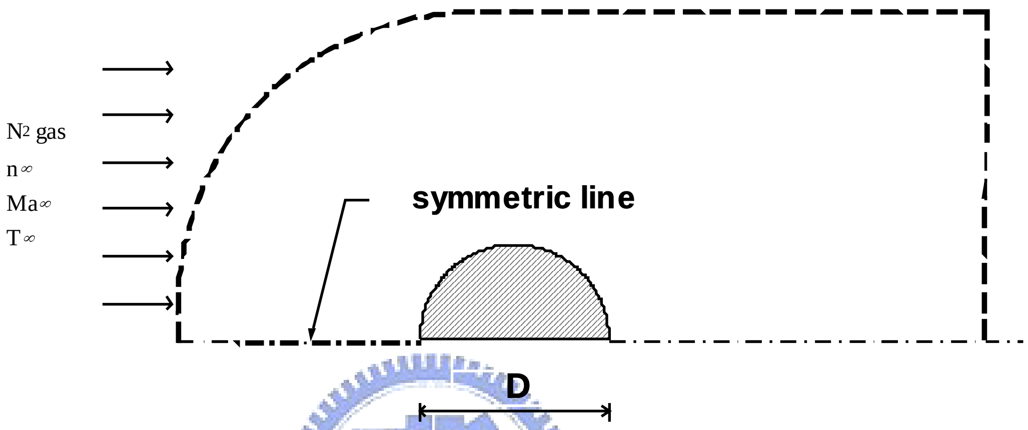
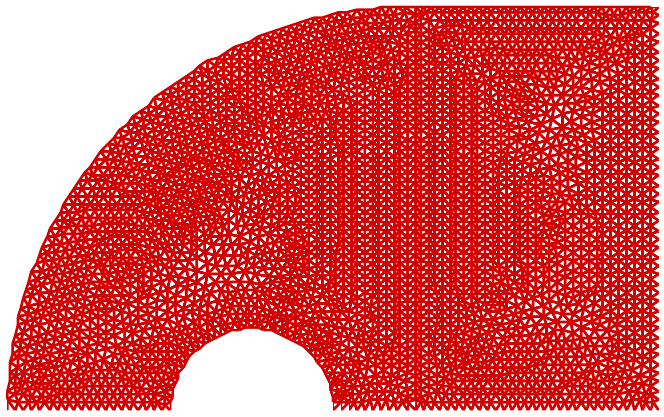
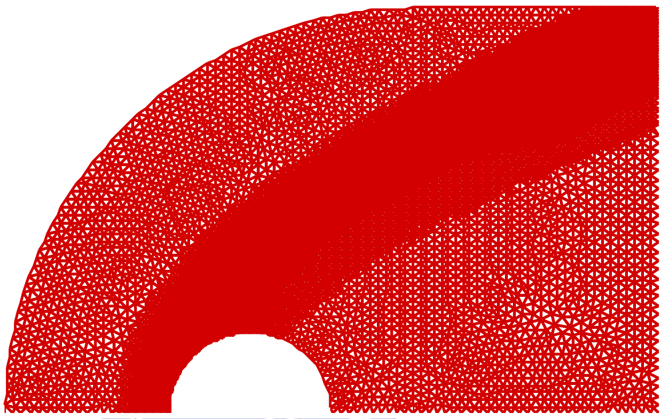


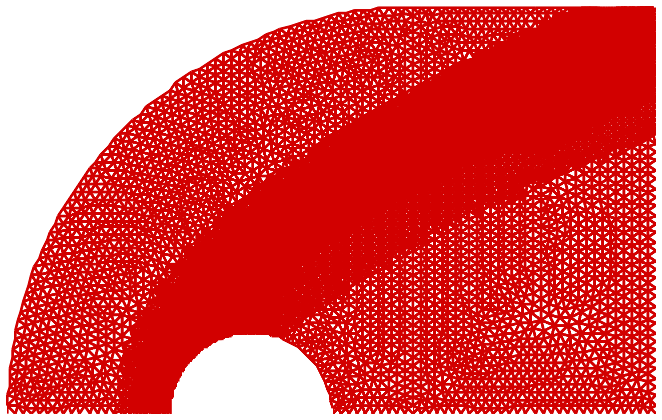
Figure 3. 8 Sketch of the computational domain of a nitrogen hypersonic flow over a cylinder (N_2 gas, $Kn_\infty=\lambda_\infty/D$=0.025, M_∞=20, T_∞=20 K, n_∞=5.1775E19 particles/m^3, D= 1m).



(a)



(b)



(c)

Figure 3. 9 Evolution of 2-D, unstructured triangular cells for a hypersonic cylinder flow with cell quality control (a) initial (7,025); (b) level-2 (33,773); (c) level-4 (75,099).

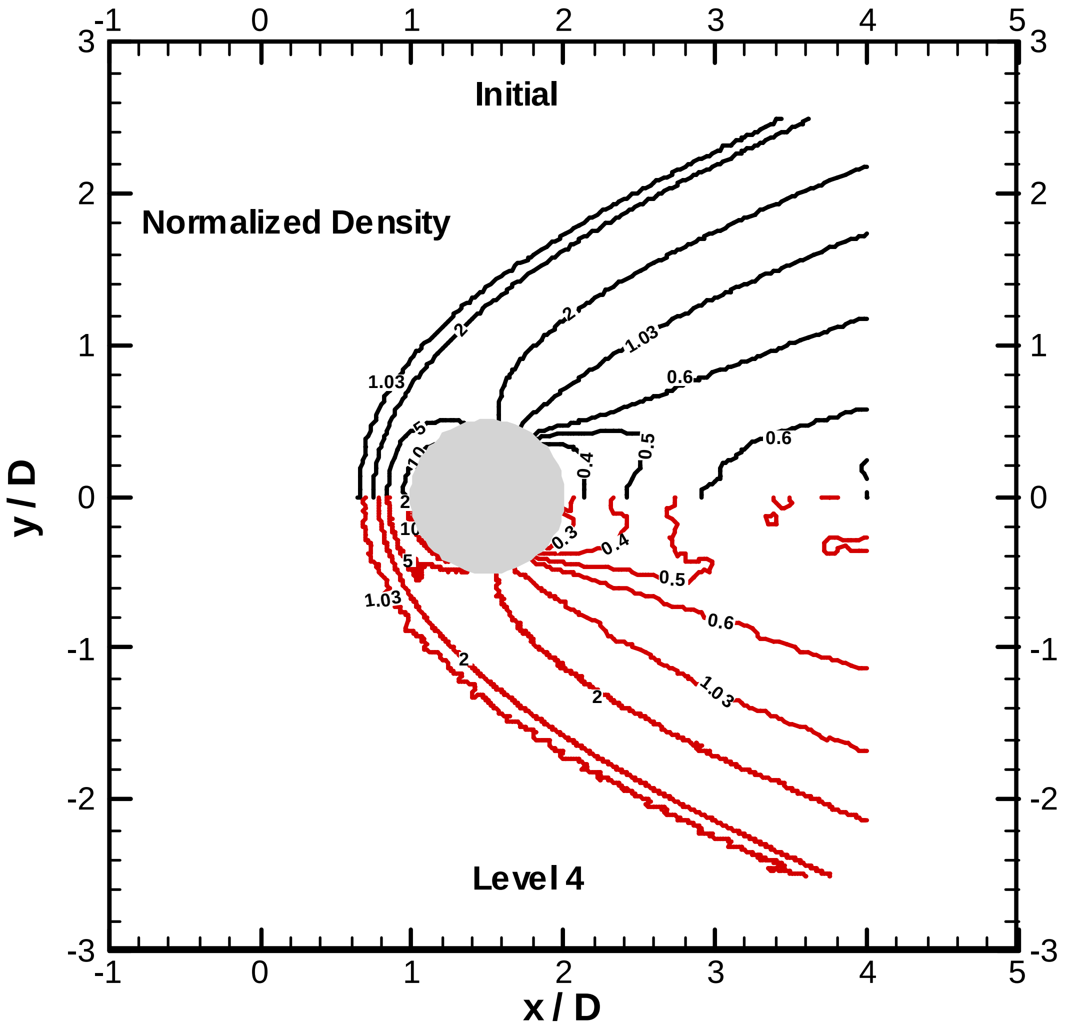
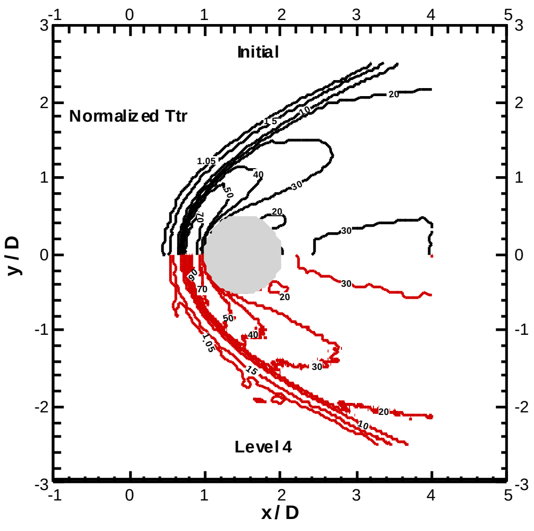
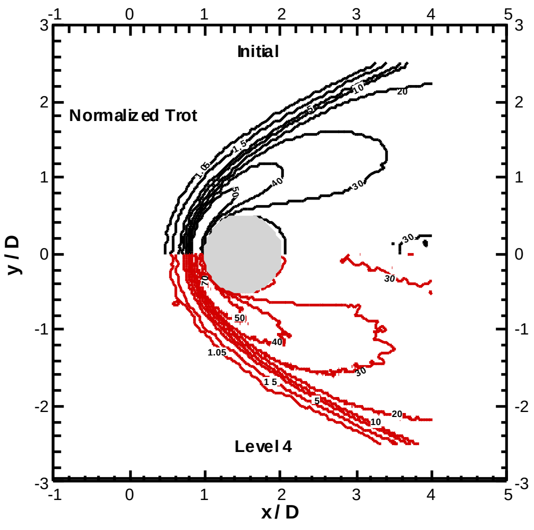


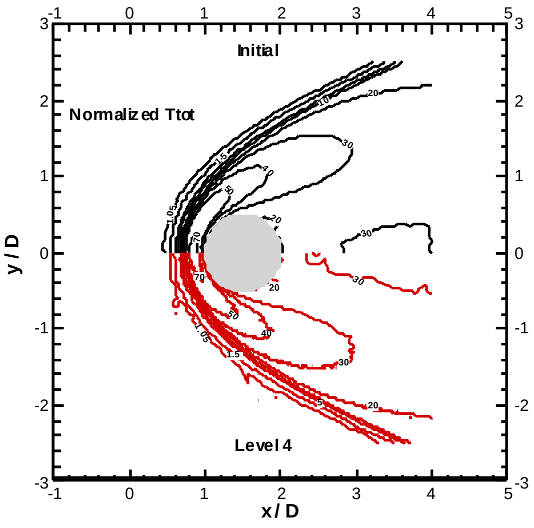
Figure 3. 10 Normalized density contour of a hypersonic cylinder flow with different meshes.



(a)

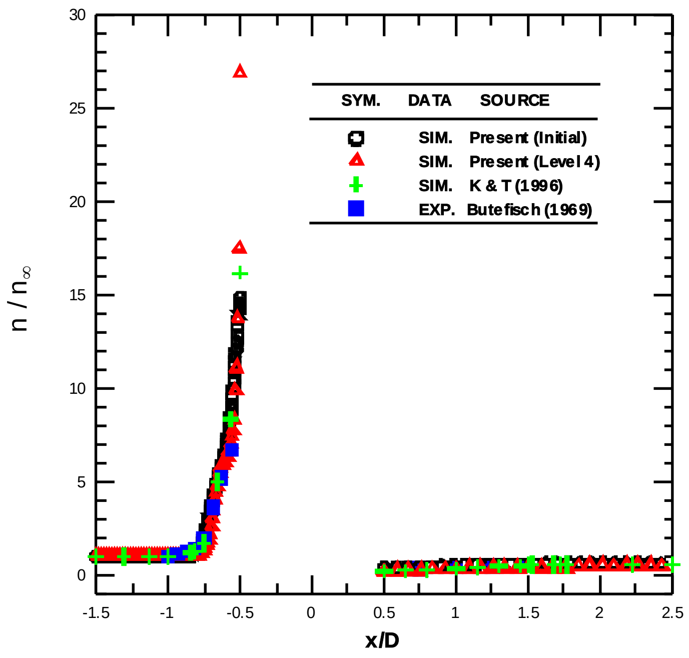


(b)

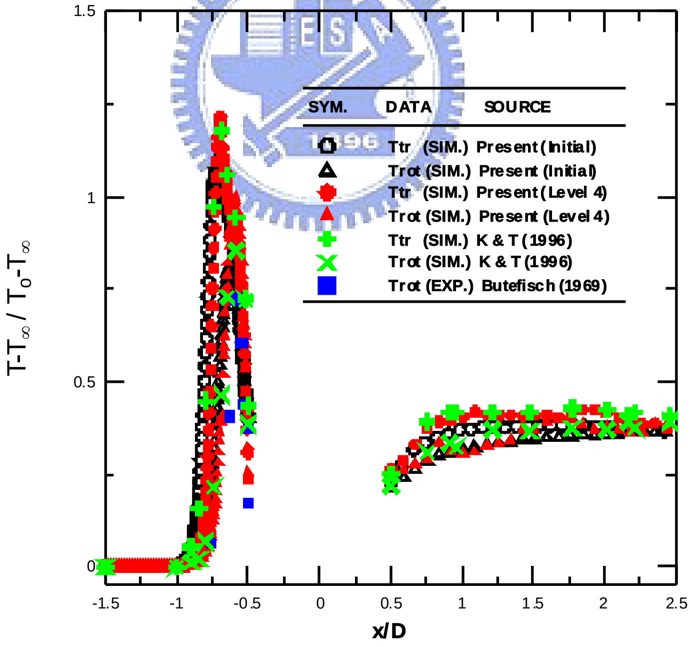


(c)

Figure 3. 11 Normalized temperature contours of a hypersonic cylinder flow with different meshes (a) translational; (b) rotational; (b) total.

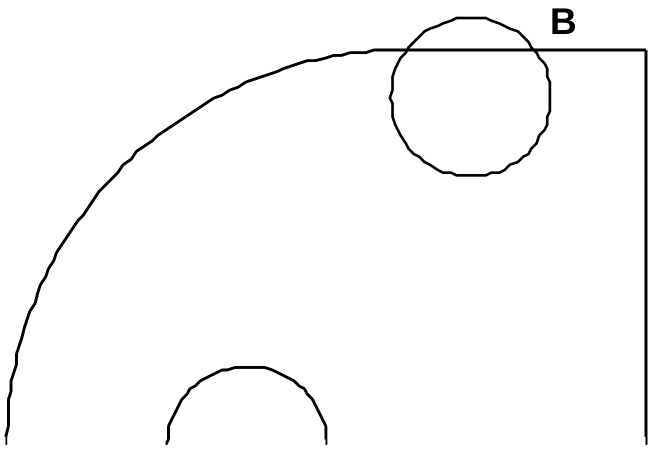


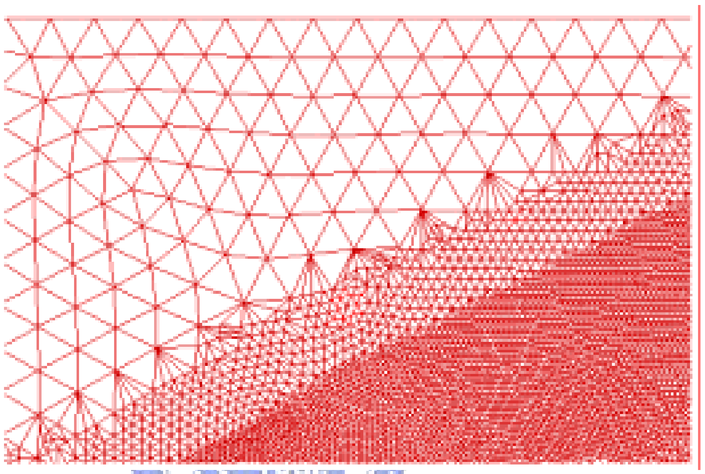
(a)

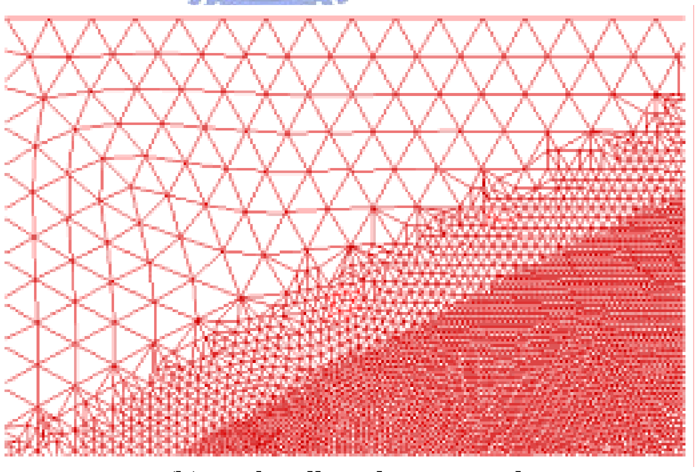


(b)

Figure 3. 12 Normalized number density and temperatures along the stagnation line with different meshes.

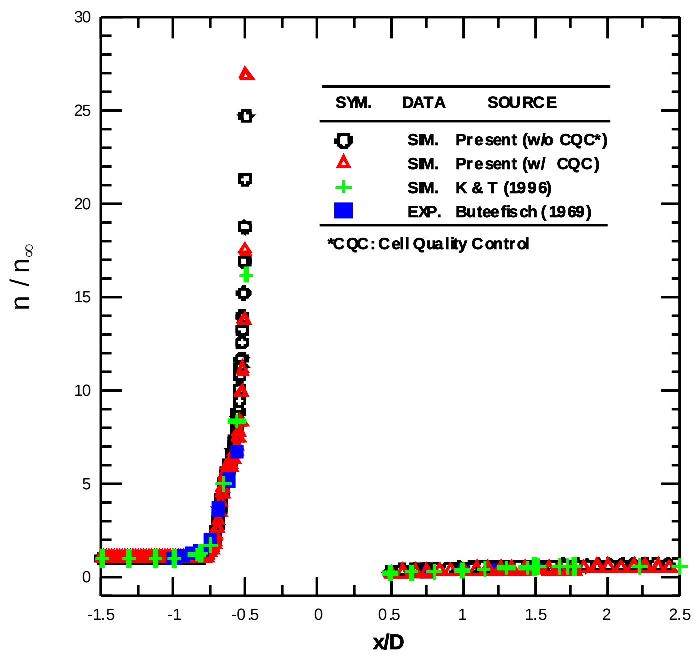



(a) without cell quality control

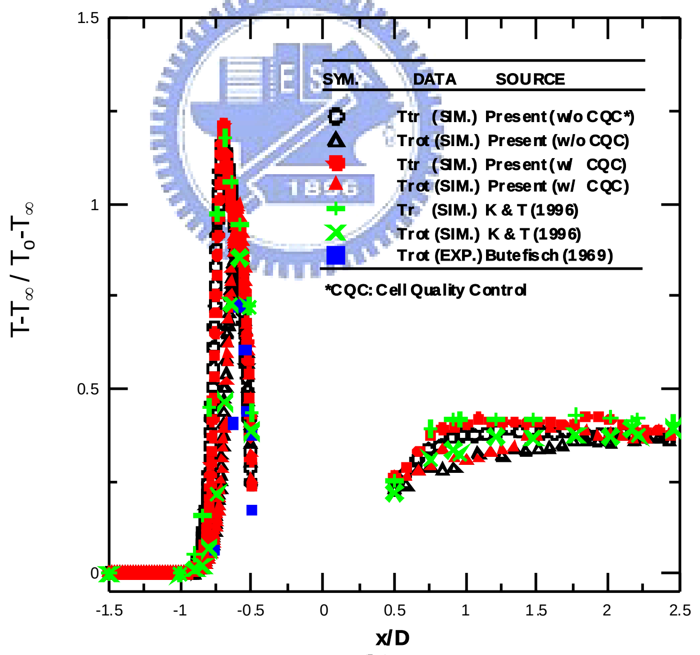

(b) with cell quality control

(c)

Figure 3. 13 Comparison of the adaptive mesh with cell quality control for a hypersonic cylinder flow.



(a)



(b)

Figure 3. 14 Normalized number density and temperatures along the stagnation line with cell quality control.

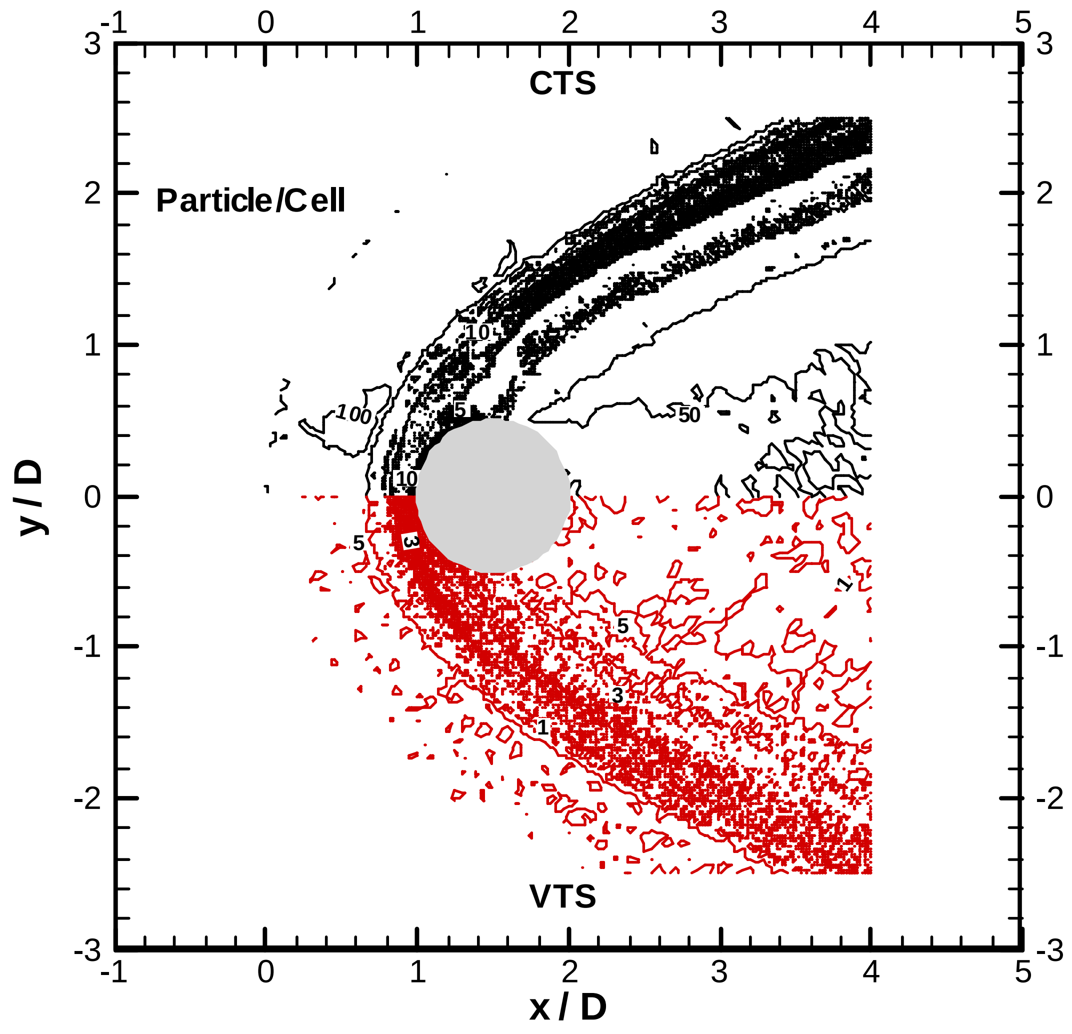


Figure 3. 15 Particle distribution of a hypersonic cylinder flow using variable time-step (VTS) and constant time-step (CTS) schemes.

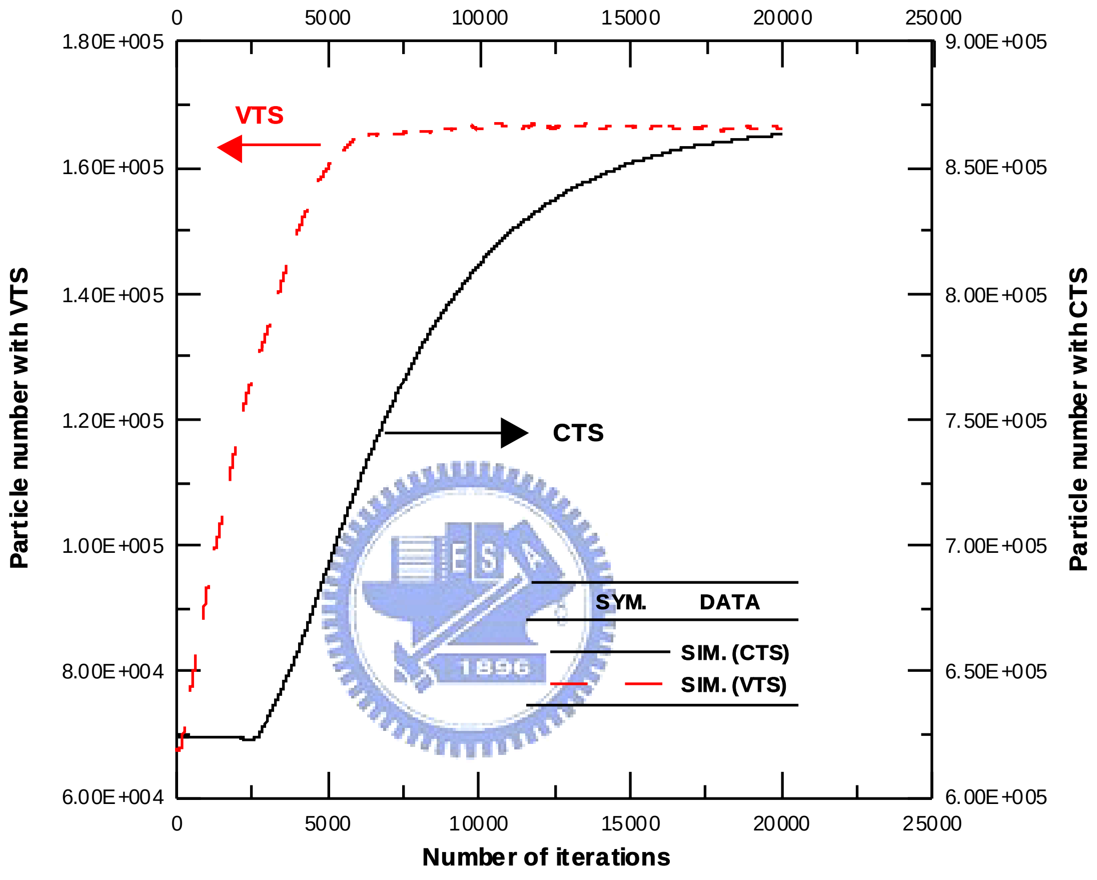


Figure 3. 16 Comparison of particle count of a hypersonic cylinder flow using variable time-step (VTS) and constant time-step (CTS) schemes.

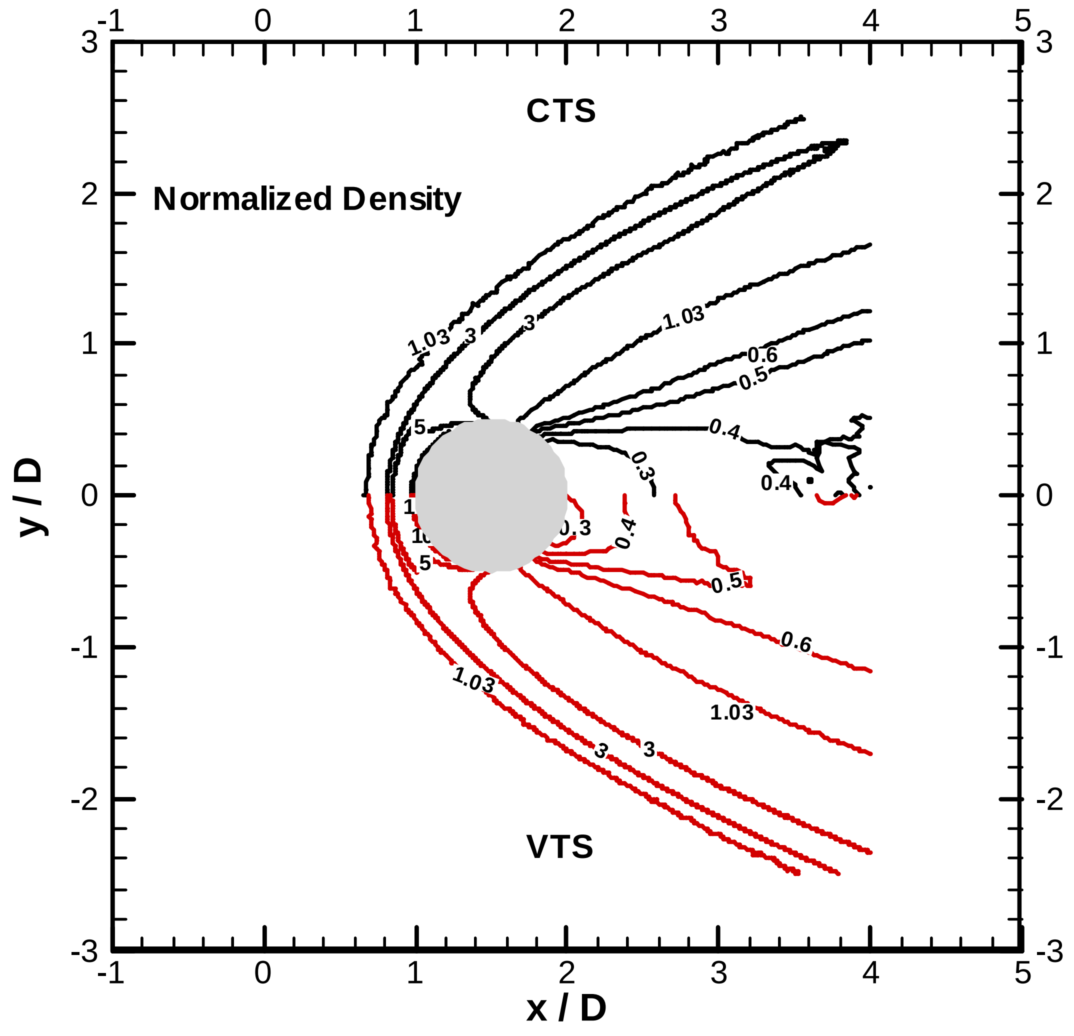
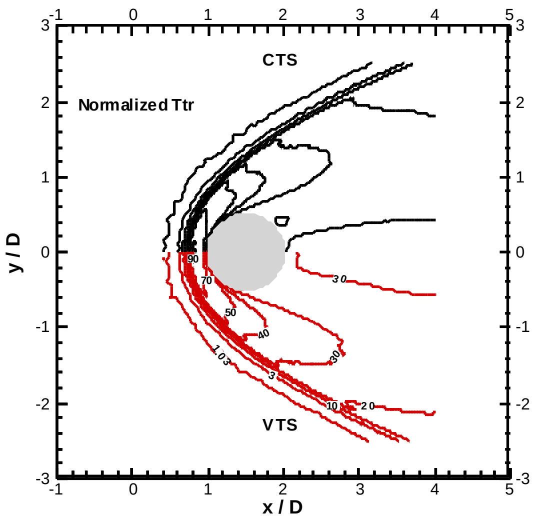
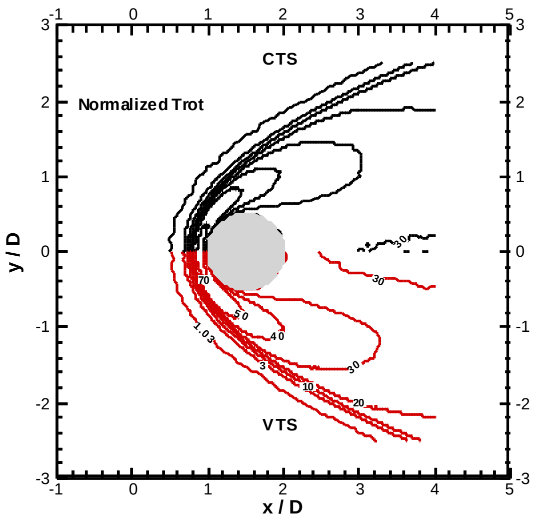


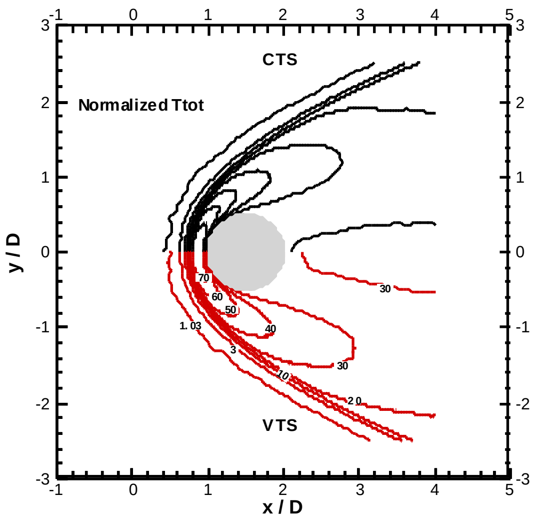
Figure 3. 17 Comparison of normalized density contour of a hypersonic cylinder flow using variable time-step (VTS) and constant time-step (CTS) schemes (x-y plane).



(a)

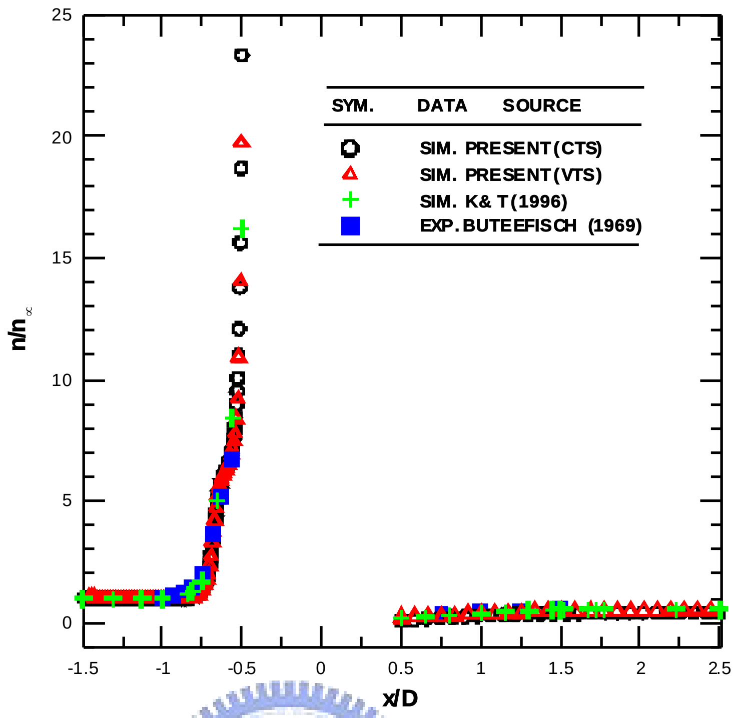


(b)

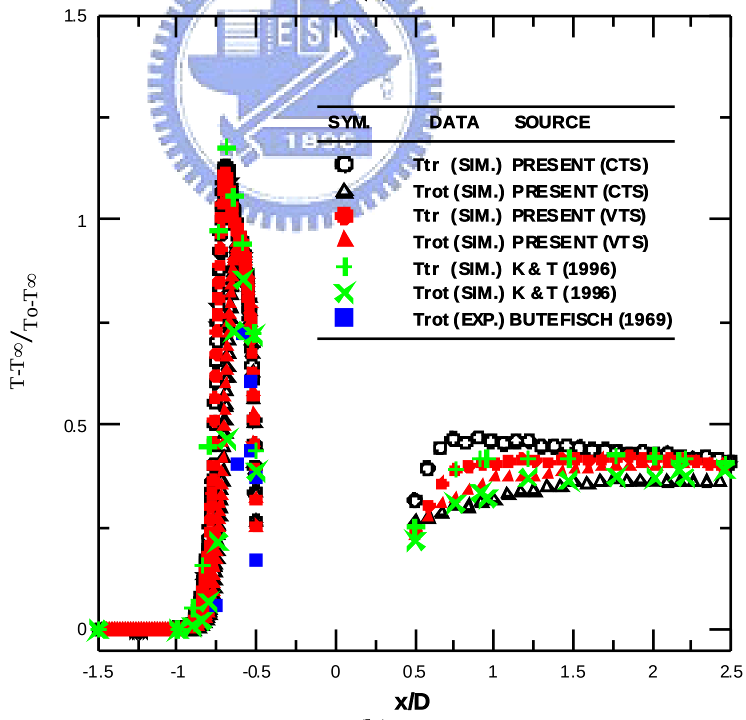


(c)

Figure 3. 18 Comparison of normalized temperature contours of a hypersonic cylinder flow using variable time-step (VTS) and constant time-step (CTS) schemes (x-y plane).



(a)



(b)

Figure 3. 19 Comparison of normalized density and temperatures of a hypersonic cylinder flow along the stagnation line using variable time-step (VTS) and constant time-step (CTS) schemes.

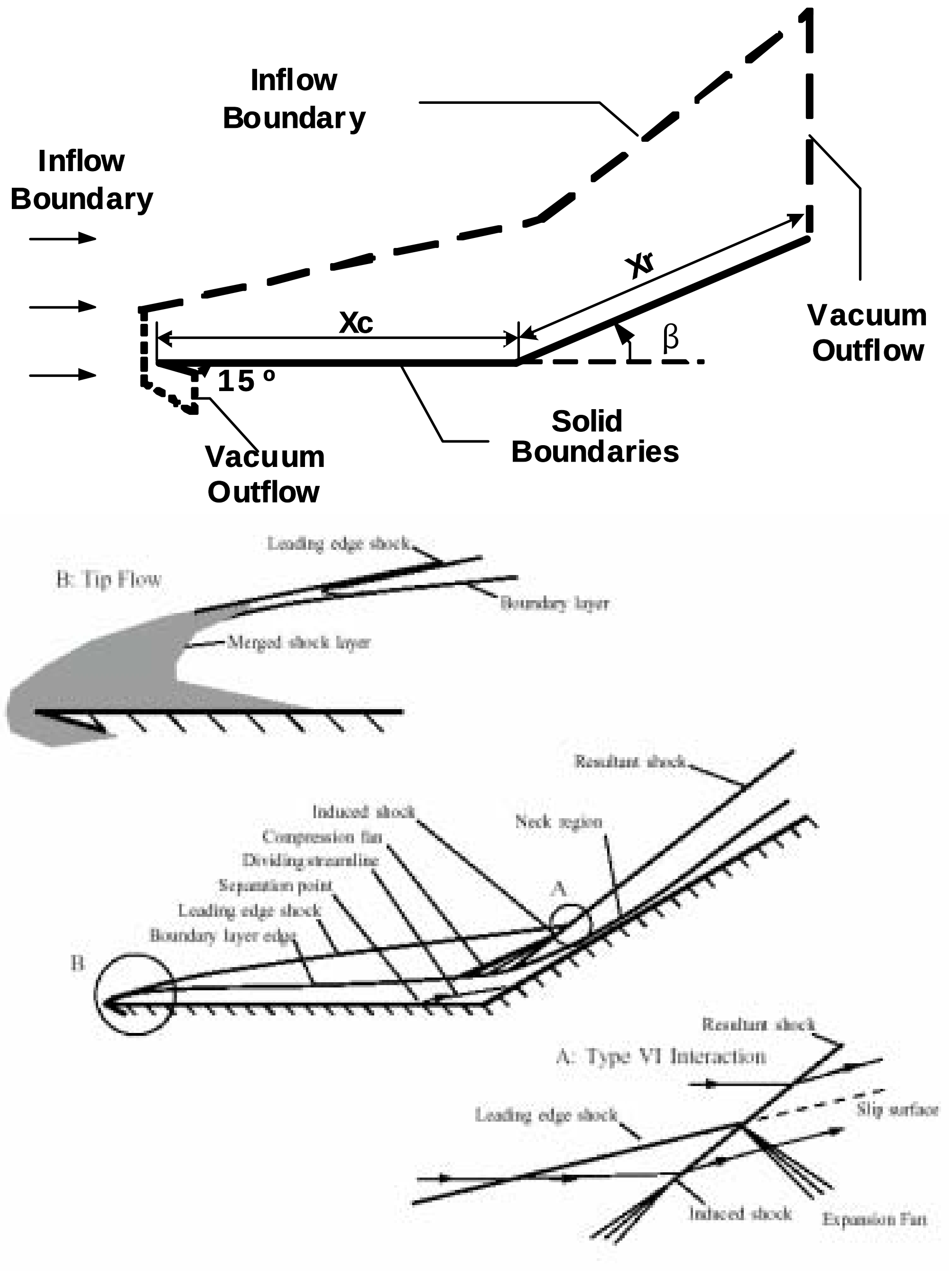


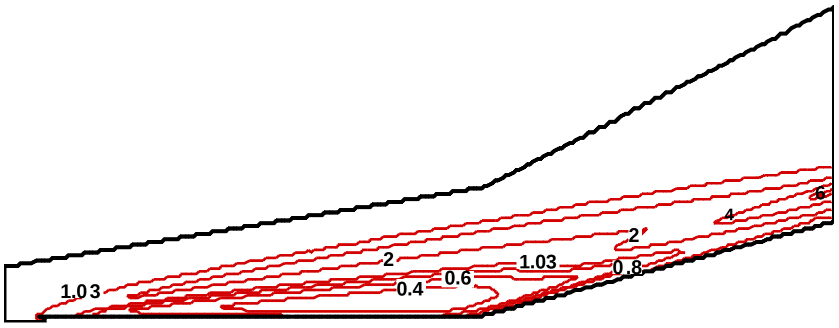
Figure 3. 20 Schematic diagram of flow features in a typical hypersonic flow over a hypersonic 15^{o}-compression ramp (N_2 gas, $Kn_\infty=\lambda_\infty/D$=2.E-4, M_∞=14.36, ρ_∞=5.221E-4 kg/m^3, T_∞=84.83 K, *Xc*=43.891 cm, *Xr*=36.86 cm).

(a)

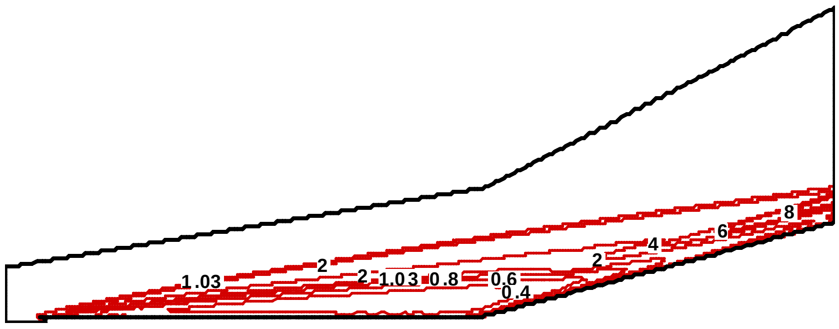
(b)

(c)

Figure 3. 21 Evolution of 2-D, unstructured triangular cells for a hypersonic flow over a hypersonic 15^{o}-compression ramp with adaptive refinement (a) initial (15,063); (b) level-1 (30,219); (b) level-2 (83,776).

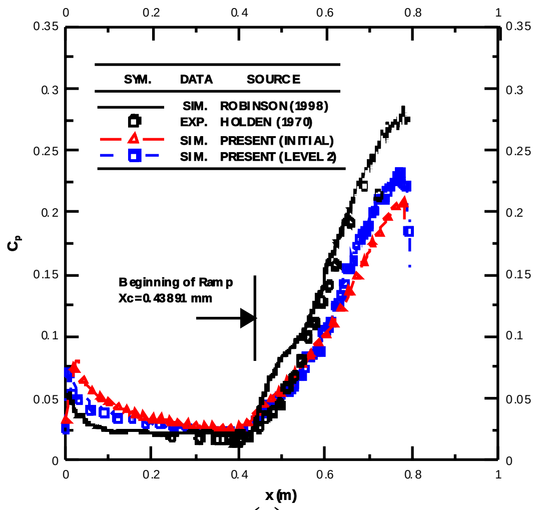


(a)

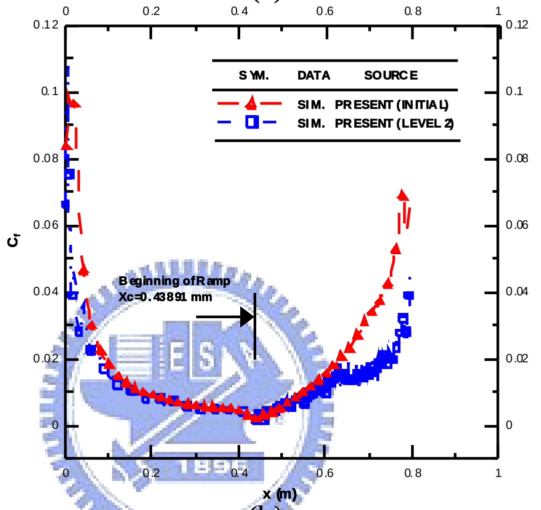


(b)

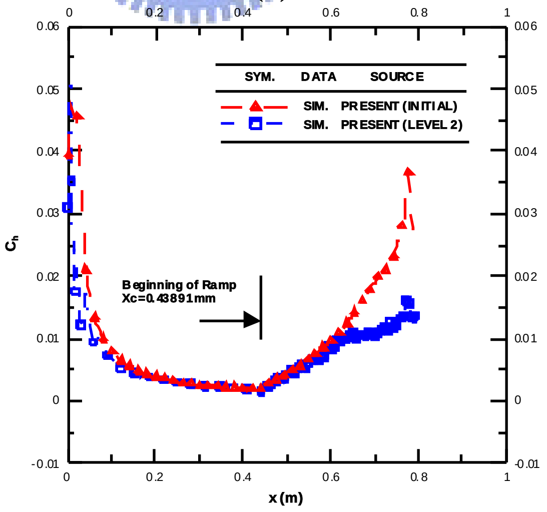
Figure 3. 22 Normalized density contour over a hypersonic 15^{o}-compression ramp with different meshes (a) initial (15,063); (b) level 2 (83,776).



(a)

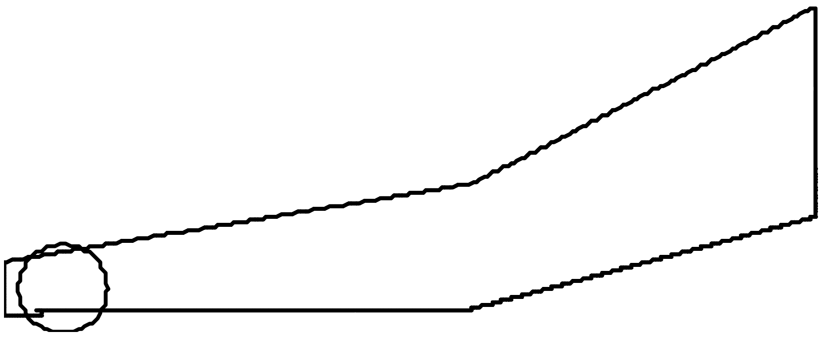


(b)

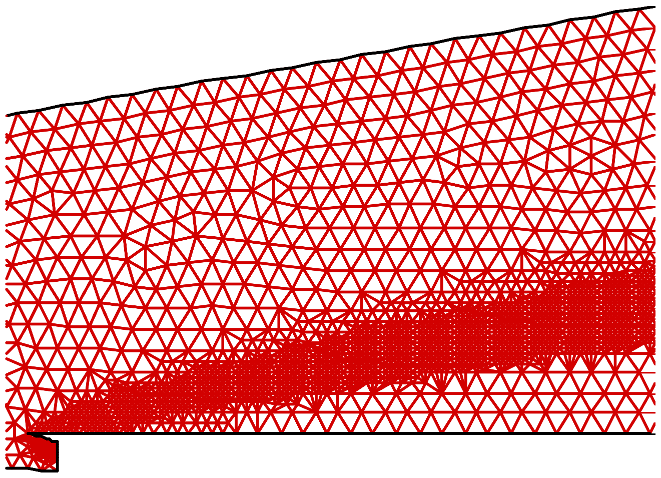


(c)

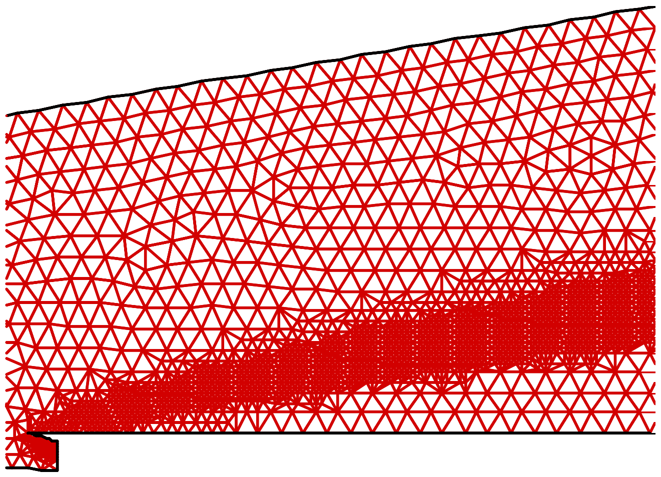
Figure 3. 23 Pressure, shear and heat transfer coefficient distributions along the solid wall for a hypersonic 15^{o}-compression ramp.



(a)

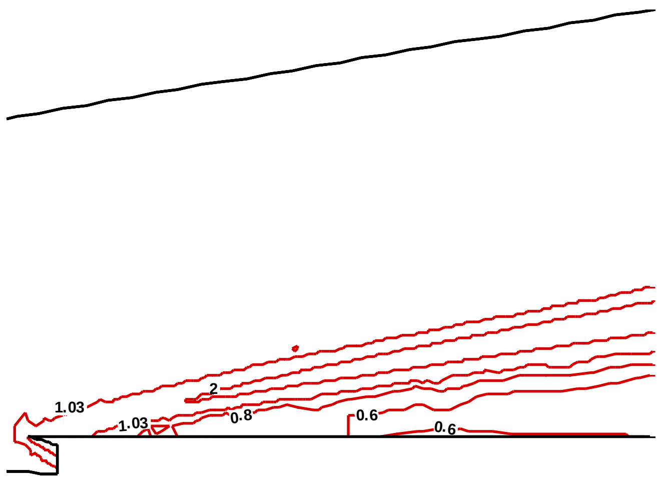


(b)

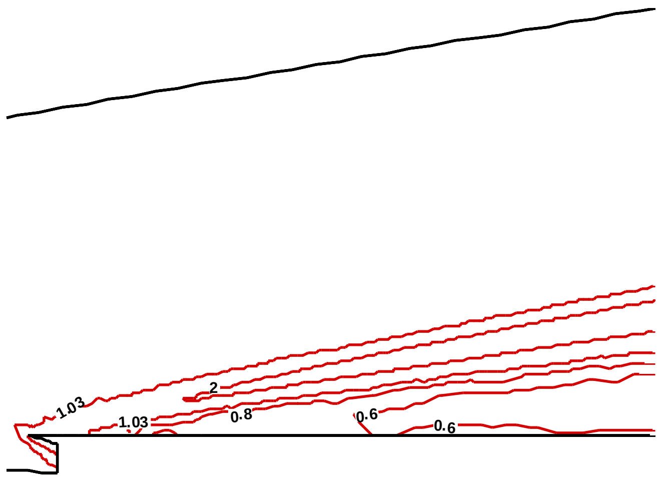


(c)

Figure 3. 24 Comparison of the adaptive mesh with cell quality control for a hypersonic 15^o-compression ramp.



(a)



(b)

Figure 3. 25 Comparison of normalized density contour with cell quality control for a hypersonic 15^{o}-compression ramp (a) with CQC; (b) without CQC.

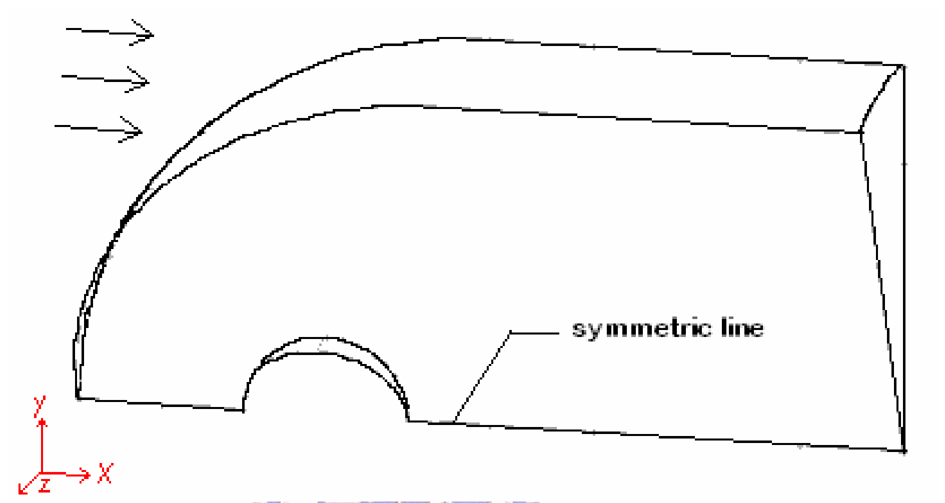


Figure 3. 26 Sketch of a hypersonic flow over a one to sixteenth sphere (N_2 gas, Kn_∞=0.01035, D=1.28 cm, T_w=300 K, T_∞=66.25 K, M_∞=4.2).

1896

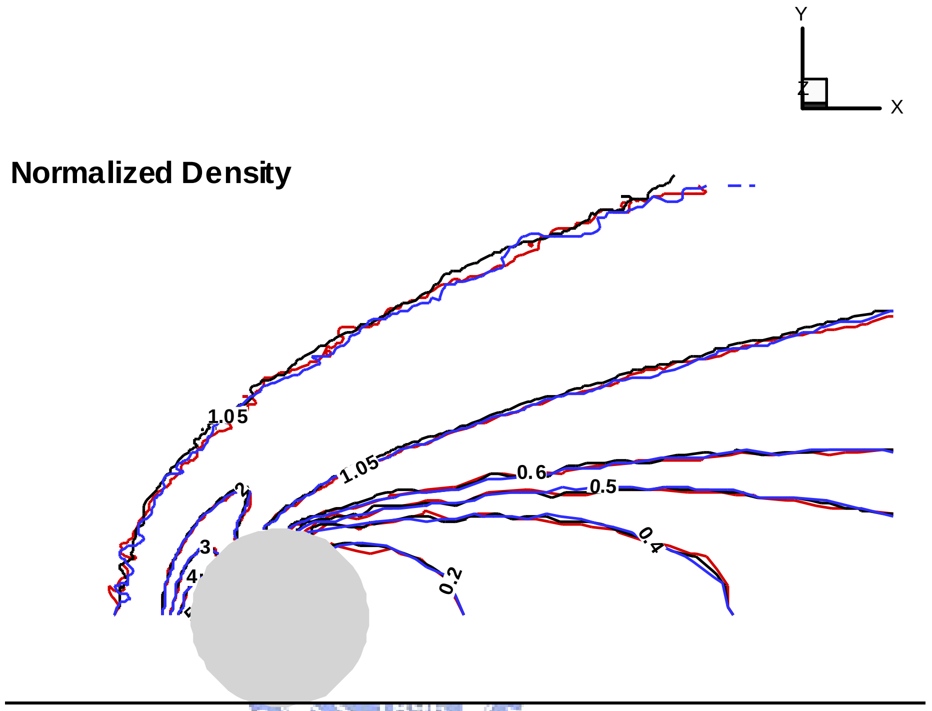
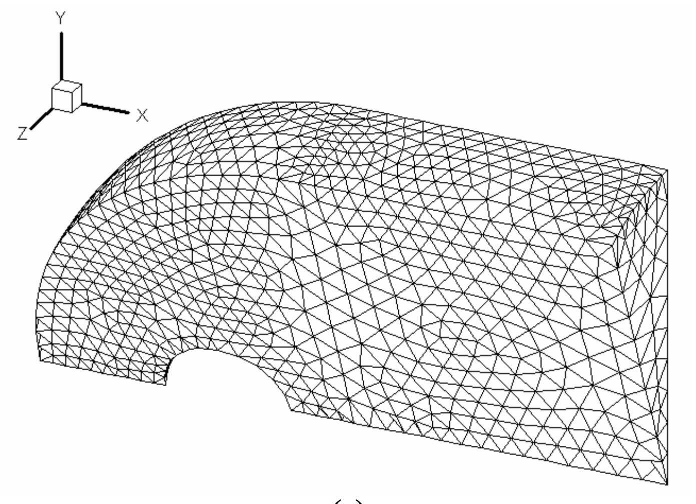
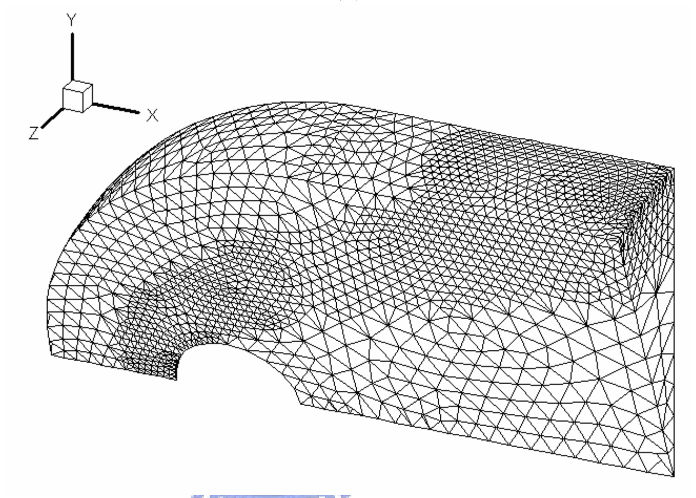


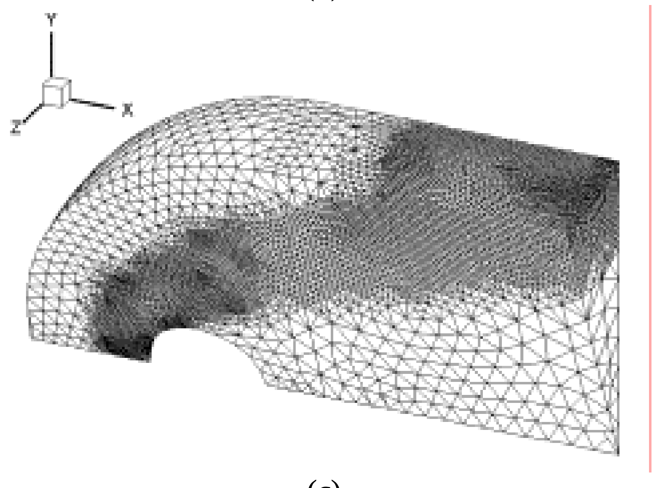
Figure 3. 27 Normalized density contours of a 3-D, unstructured tetrahedral cell for a hypersonic sphere flow with three different cross sections.



(a)

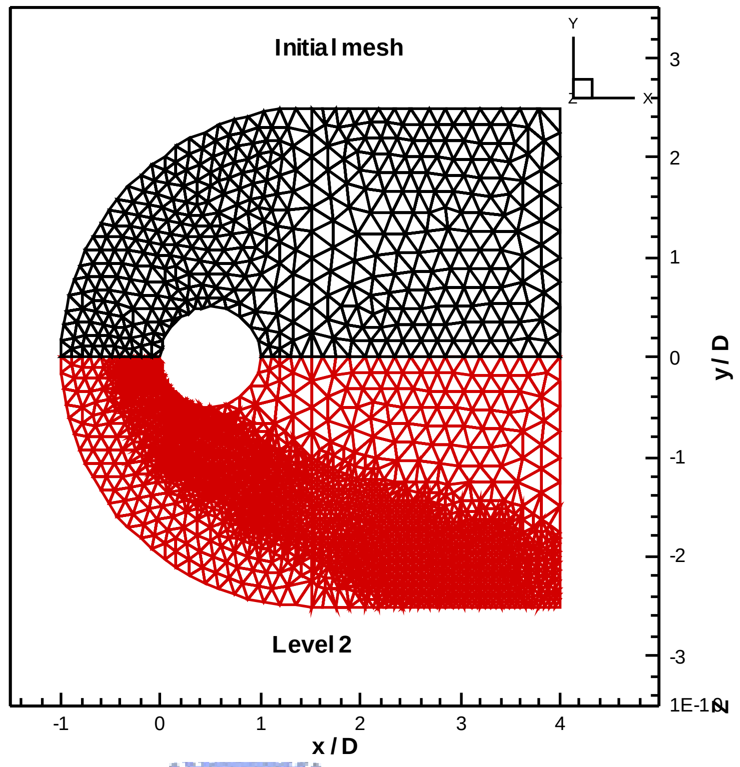


(b)

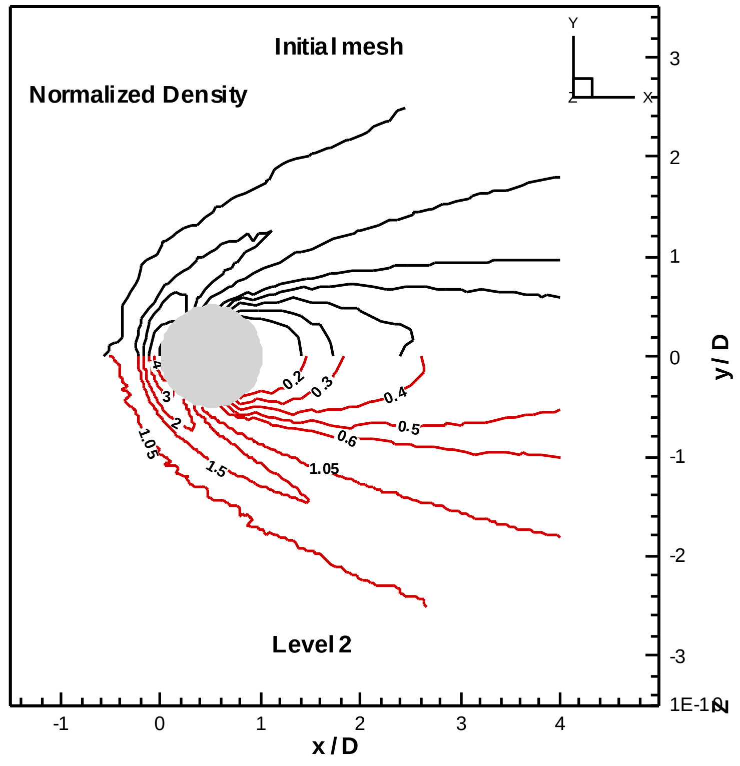


(c)

Figure 3. 28 Evolution of a 3-D, unstructured tetrahedral cells for a hypersonic sphere flow (a) initial (5,353); (b) level-1 (22,510); (c) level-2 (164,276).

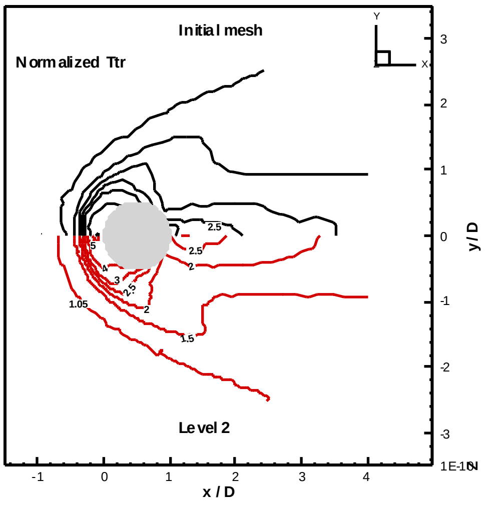


(a)

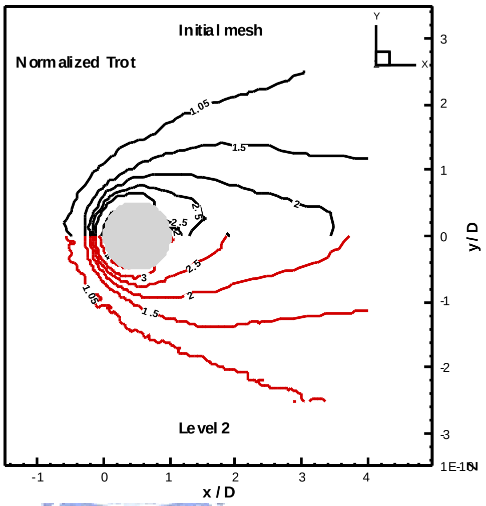


(b)

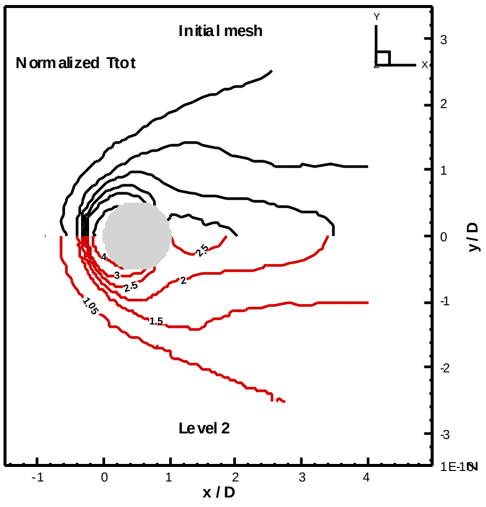
Figure 3. 29 Comparison of normalized density of a hypersonic sphere flow with different meshes (a) mesh distribution; (b) normalized contour.



(a)



(b)



(c)

Figure 3. 30 Comparison of normalized temperatures of a hypersonic sphere flow with different meshes (a) translational; (b) rotational; (c) total.

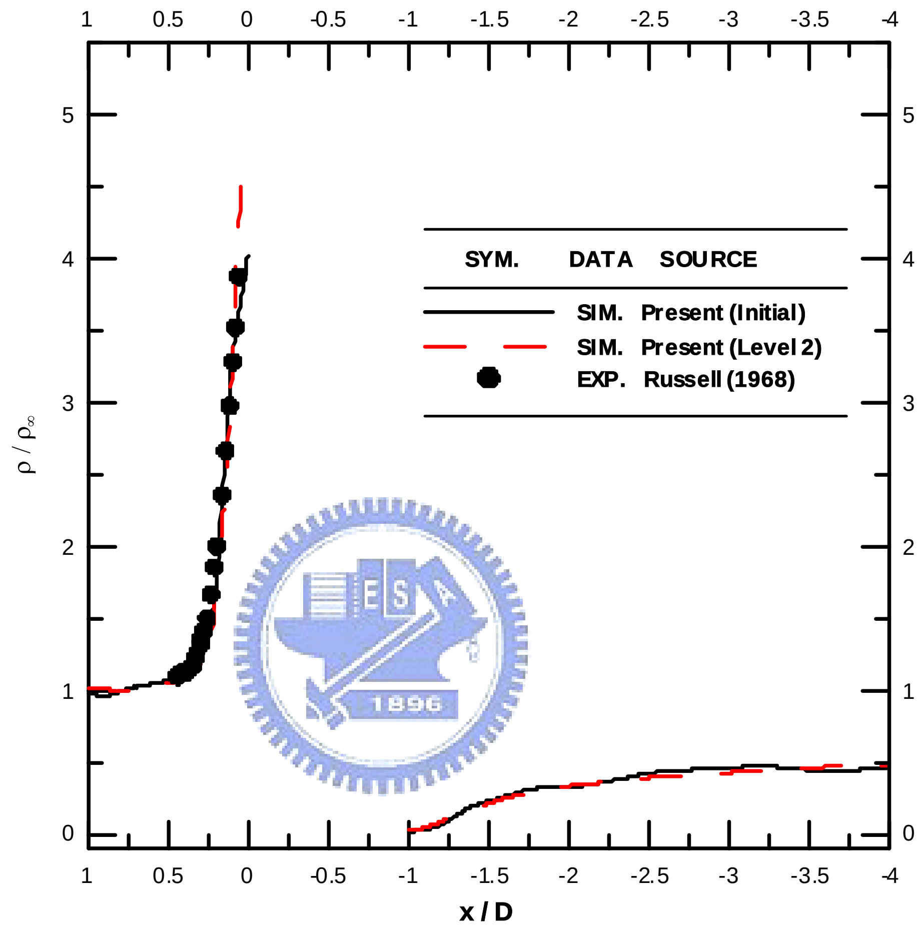
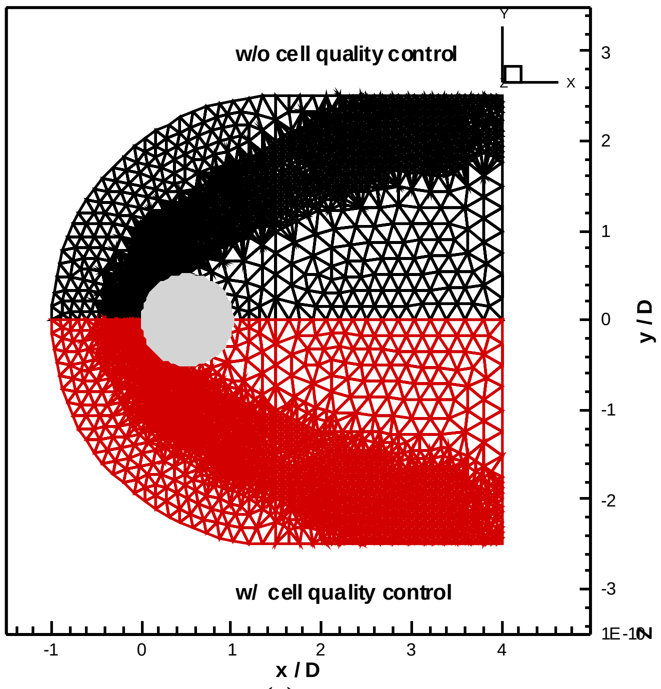
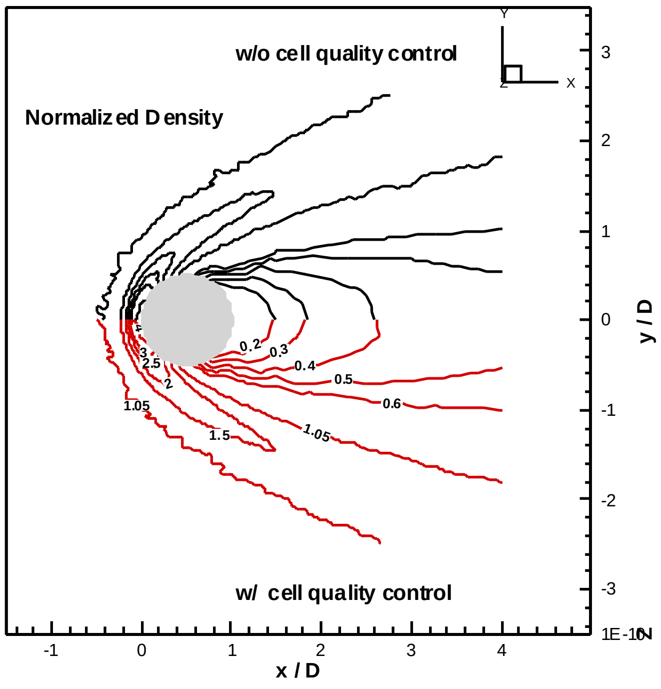


Figure 3. 31 Comparison of normalized density of a hypersonic sphere flow along the stagnation line with different meshes (a) initial; (b) level-2.



(a)



(b)

Figure 3. 32 Comparisons of the adaptive mesh and normalized density contour with or without cell quality control at x-y plane for a hypersonic sphere flow.

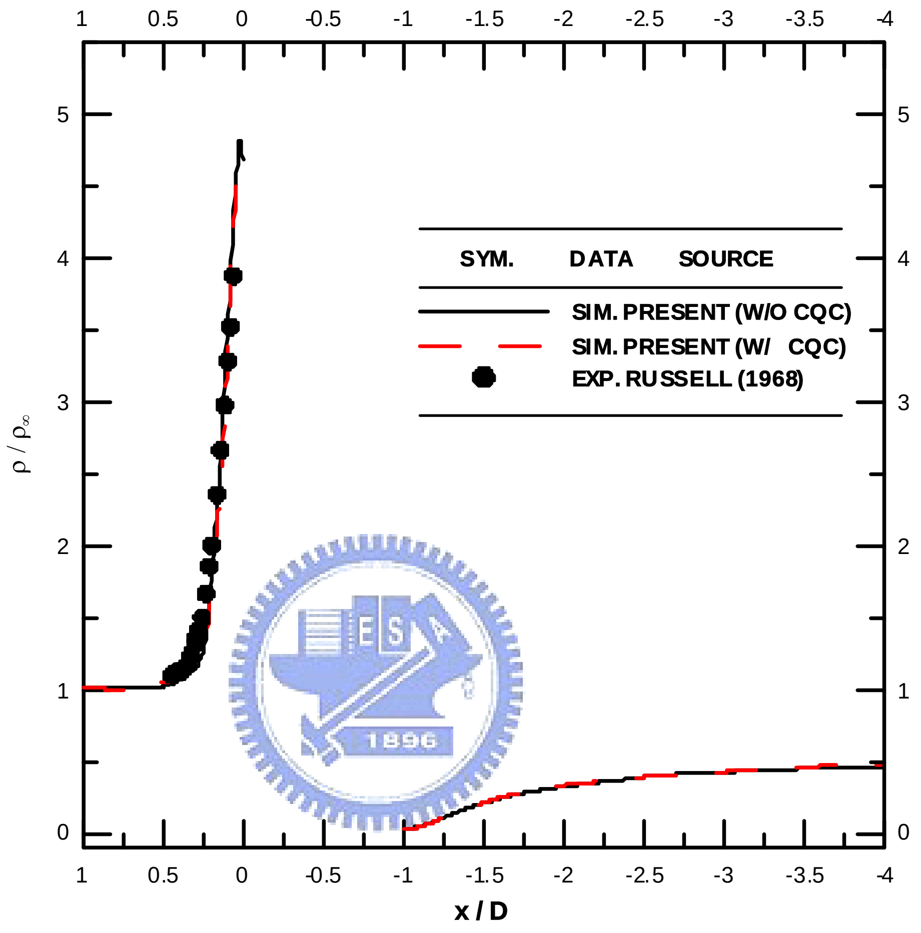


Figure 3. 33 Density distribution of a hypersonic sphere flow along the stagnation line with or without cell quality control.

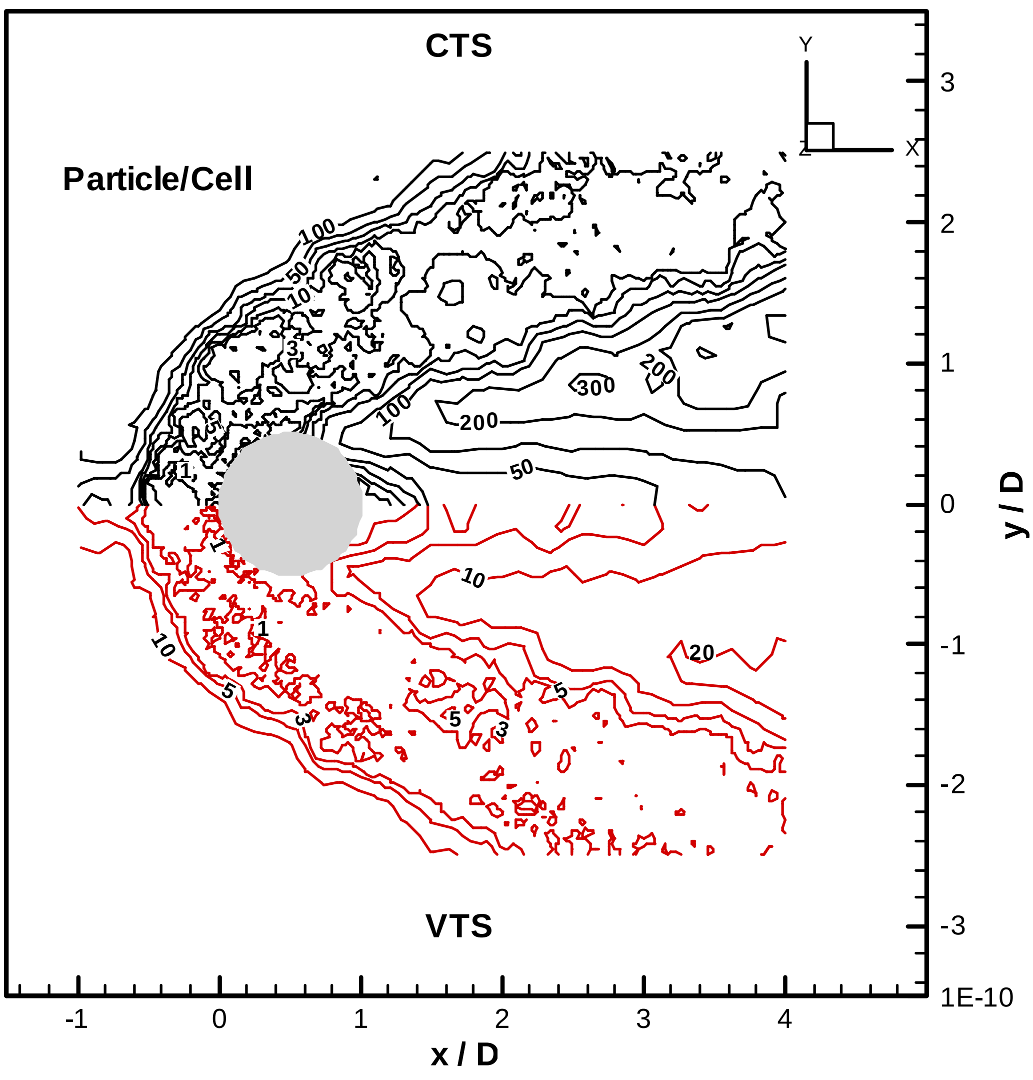


Figure 3. 34 Particle distribution of a hypersonic sphere flow using variable time-step (VTS) and constant time-step (CTS) schemes.

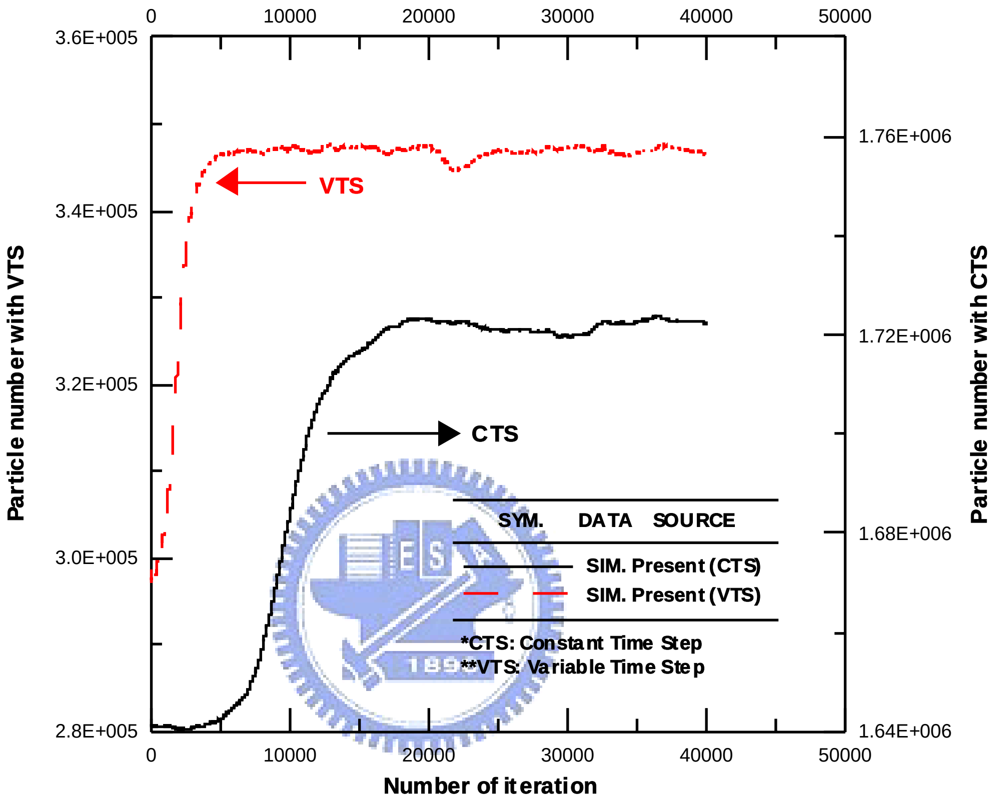


Figure 3. 35 Comparison of particle count of a hypersonic sphere flow using variable time-step (VTS) and constant time-step (CTS) schemes.

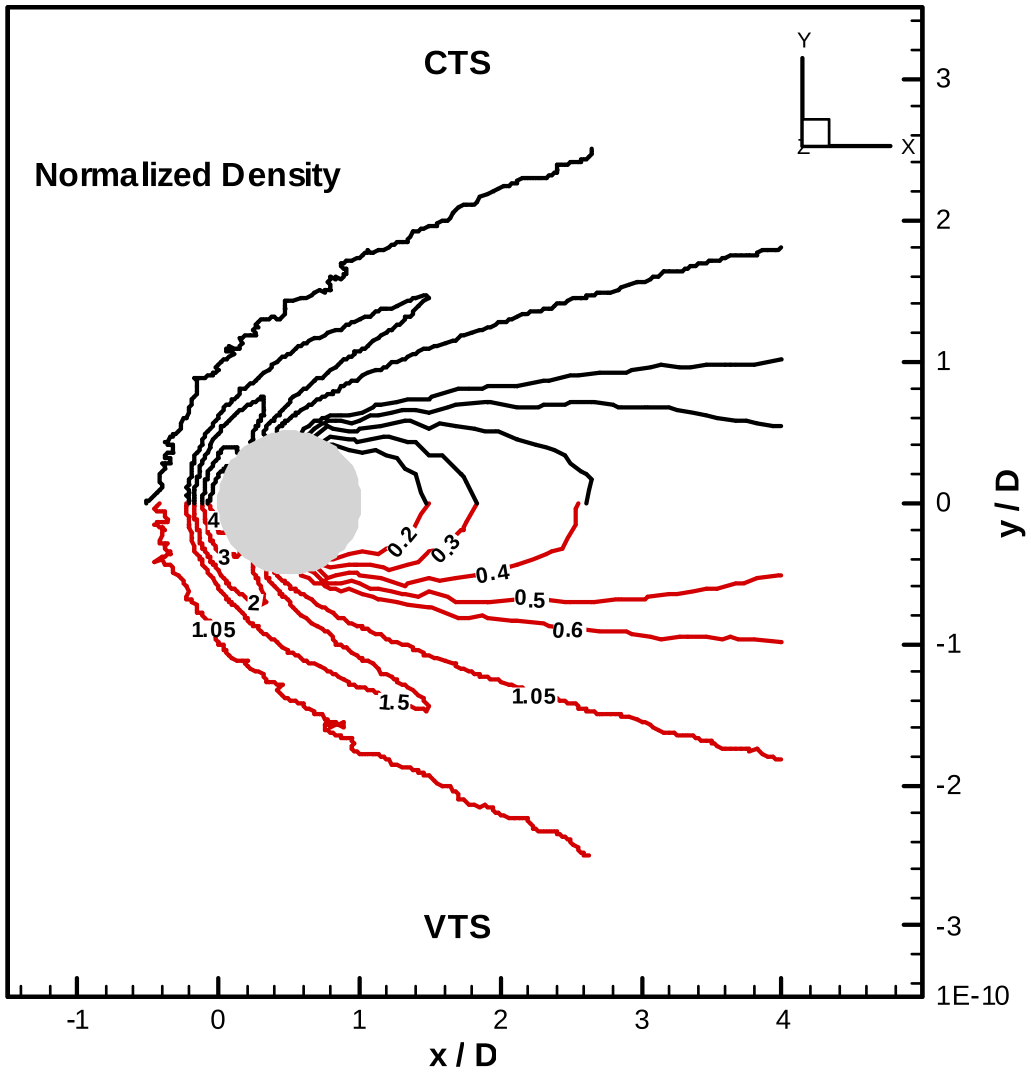
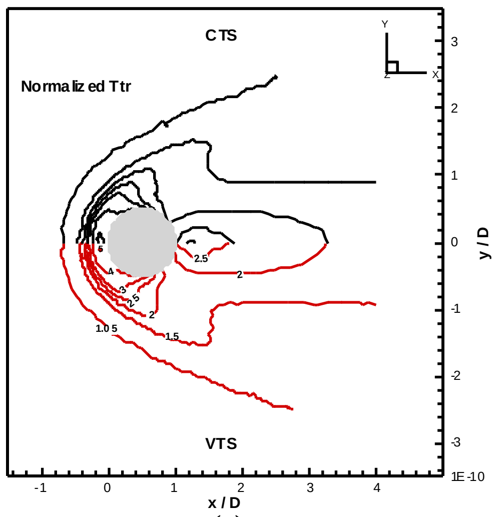
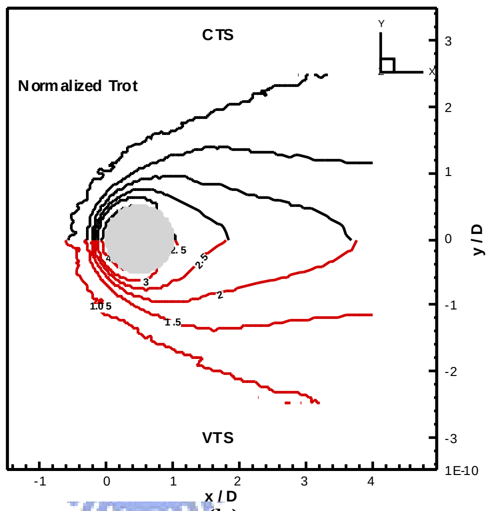


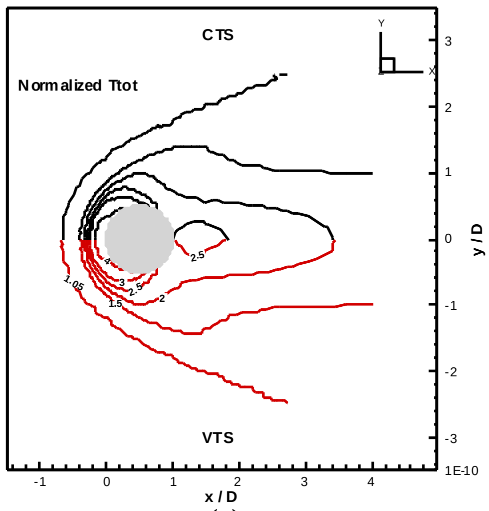
Figure 3. 36 Comparison of normalized density contour over a hypersonic cylinder flow using variable time-step (VTS) and constant time-step (CTS) schemes.



(a)



(b)



(c)

Figure 3. 37 Comparison of normalized temperature contour over a hypersonic cylinder flow using variable time-step (VTS) and constant time-step (CTS) schemes (a) translational; (b) rotational; (c) total.

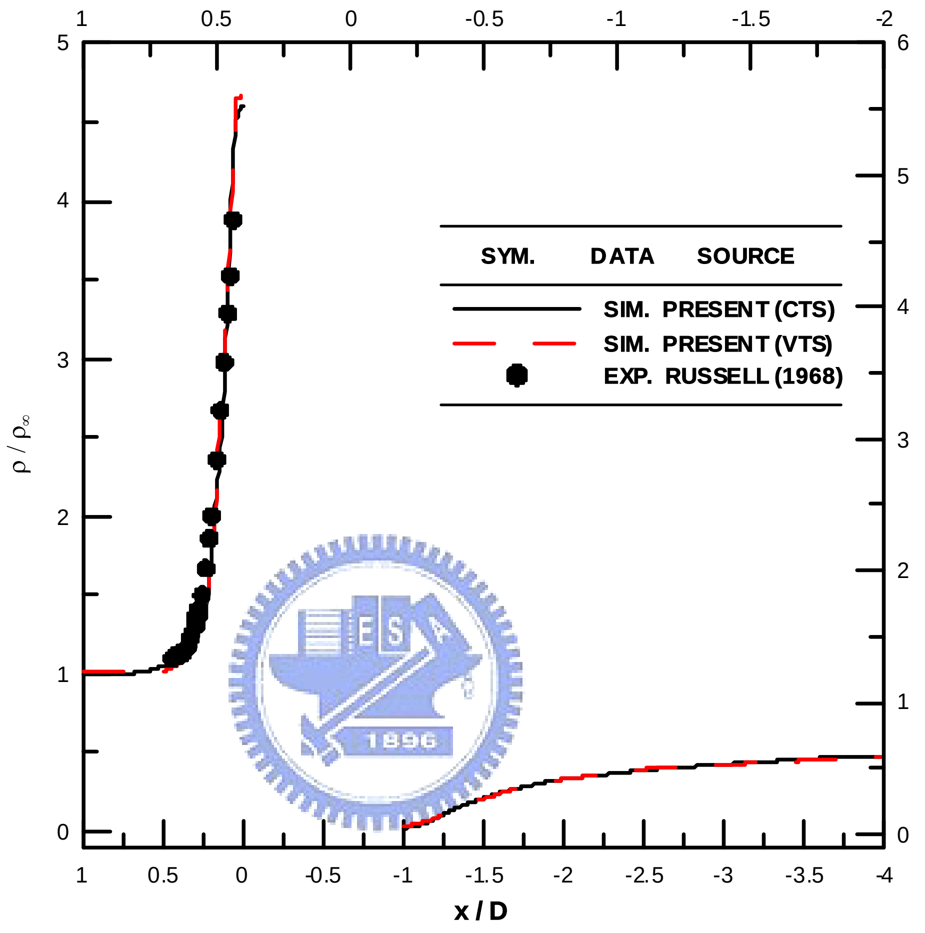


Figure 3. 38 Comparison of normalized density of a hypersonic sphere flow along the stagnation line using variable time-step (VTS) and constant time-step (CTS) schemes.

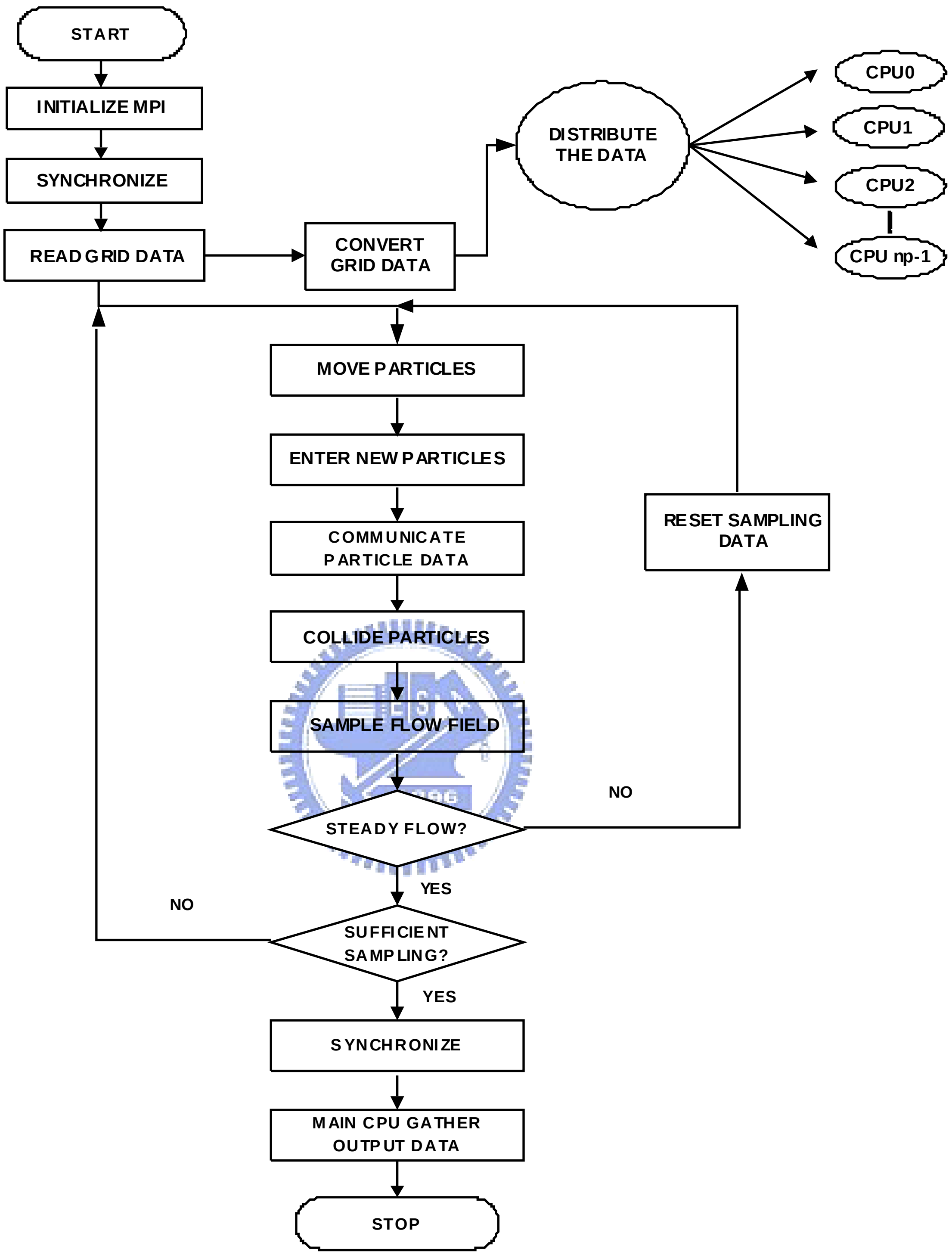


Figure 4. 1 The flowchart of the parallel DSMC method.

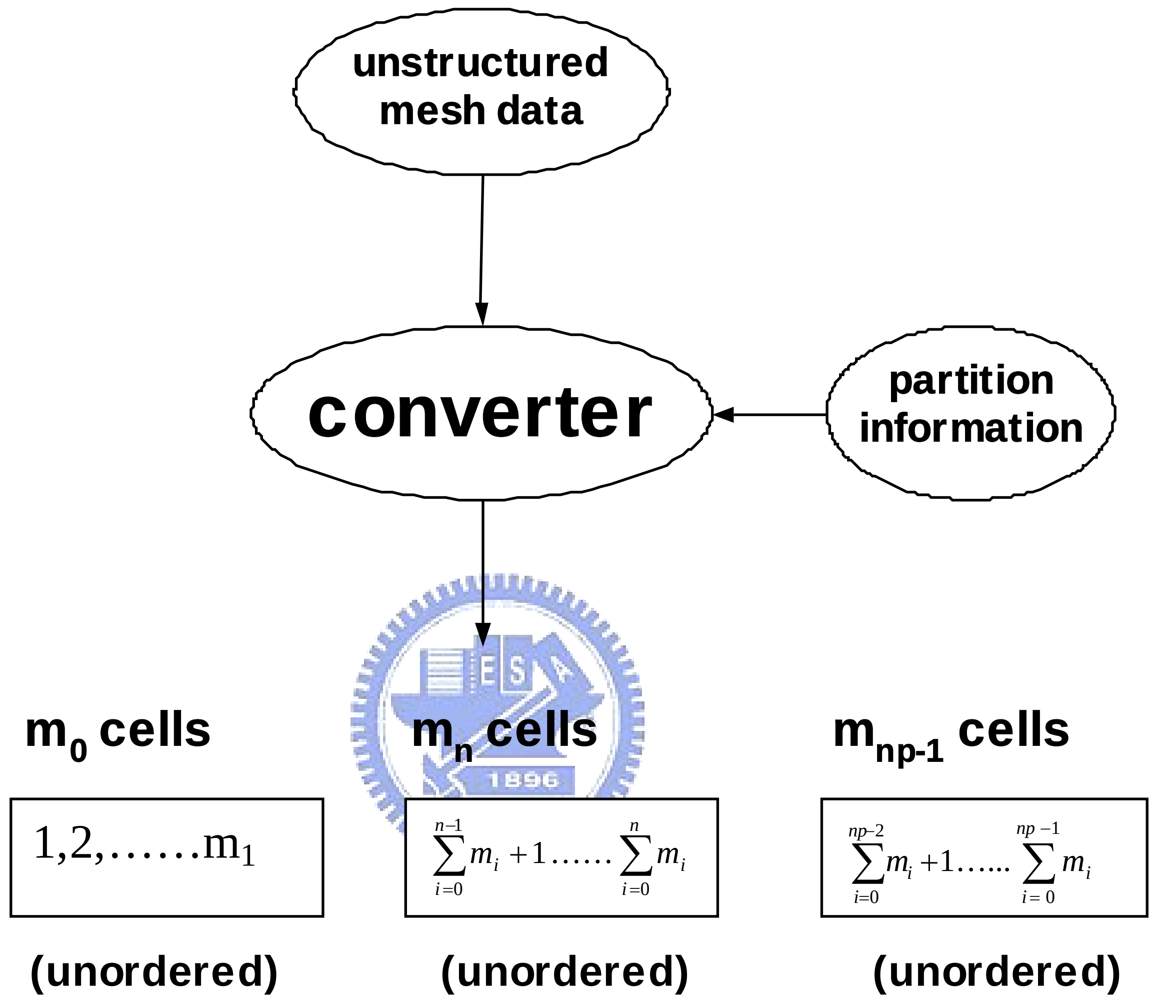


Figure 4. 2 Schematic diagram of the proposed cell numbering scheme (m_n is the number of cells in the nth processor, where the starting and ending cell number are $\sum_{i=1}^{n-1} m_i+1$ and $\sum_{i=1}^{n} m_i$, respectively. np is the total number of processors) [84].

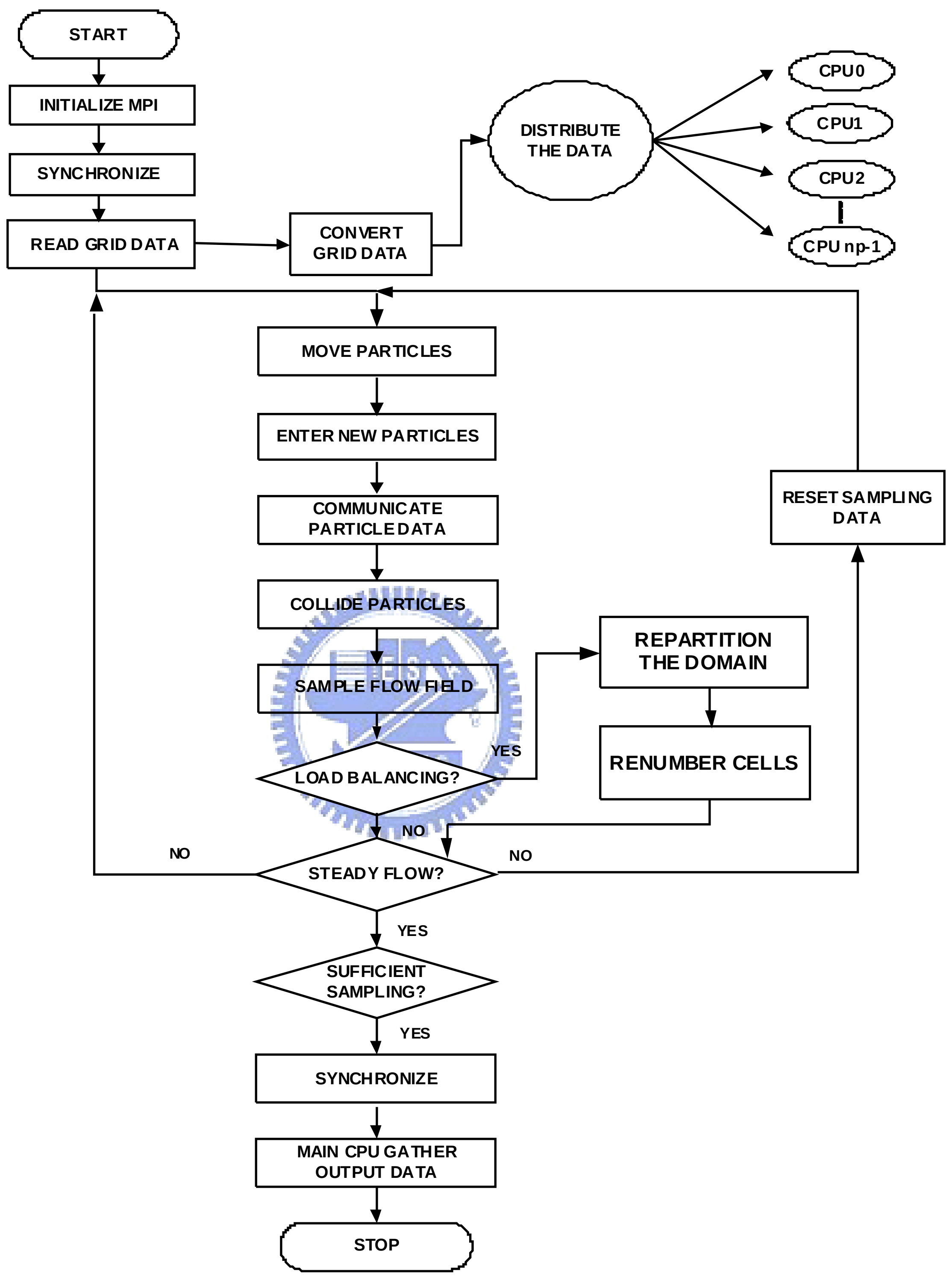


Figure 4. 3 The flowchart of the parallel DSMC method with dynamic domain decomposition method.

```
Call SAR                  to determined remapping occurs or not

If repartition is true then

    Call PJINPUT          to assign the data of each sub-domain for
                          repartition

    Call PJOSTLE          to repartition domain by loading

    Call SAVEDATA
                          to record the data of transferred particles
                          and cells

    Call Barrier          to synchronize the processor

    Call SENDDATA
                          to send the transferred data to destination
                          processor

    Call RECVDATA
                          to receive the transferred data from the
                          buffer

    Call REORDER          to reorder the cell number of neighboring

    Call NBRC             to reset the number of neighboring CPU

End if
```

Figure 4. 4 Sketch of the dynamic domain decomposition method.

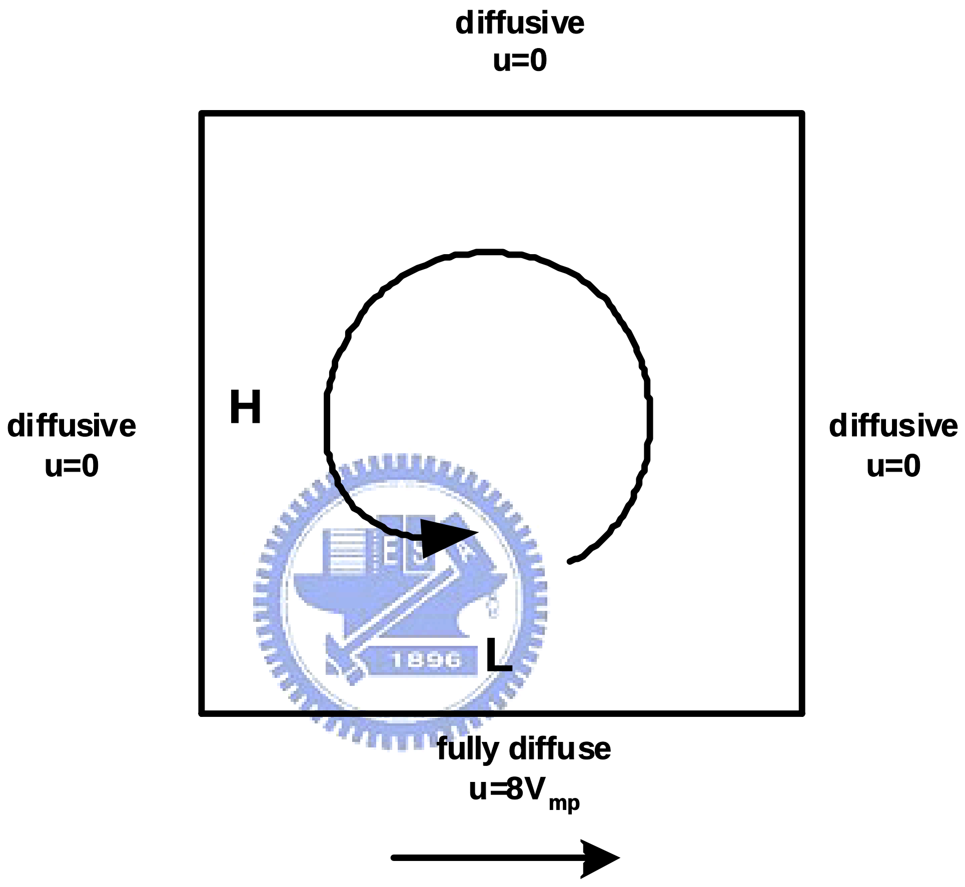


Figure 4. 5 Sketch of a two-dimensional high-speed driven cavity flow (Ar gas, V_p=8*C_{mp}, T_w=300 K, L/H=1, L=0.32 m, Kn=0.04).

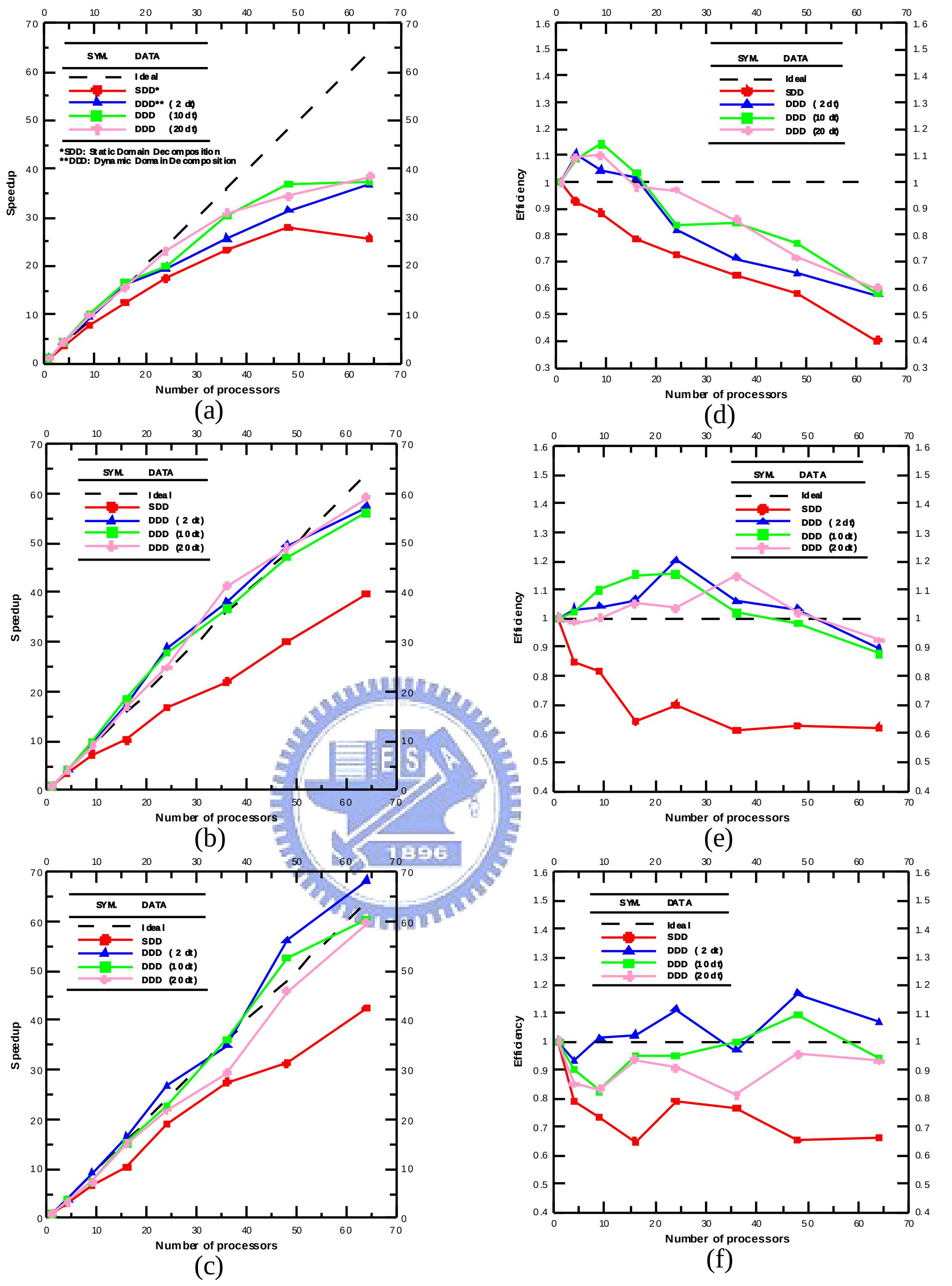


Figure 4. 6 Parallel speedup and efficiency as a function of number of processors for high-speed driven cavity flow at different problem sizes on IBM-SP2 machine (maximum 64 processors).

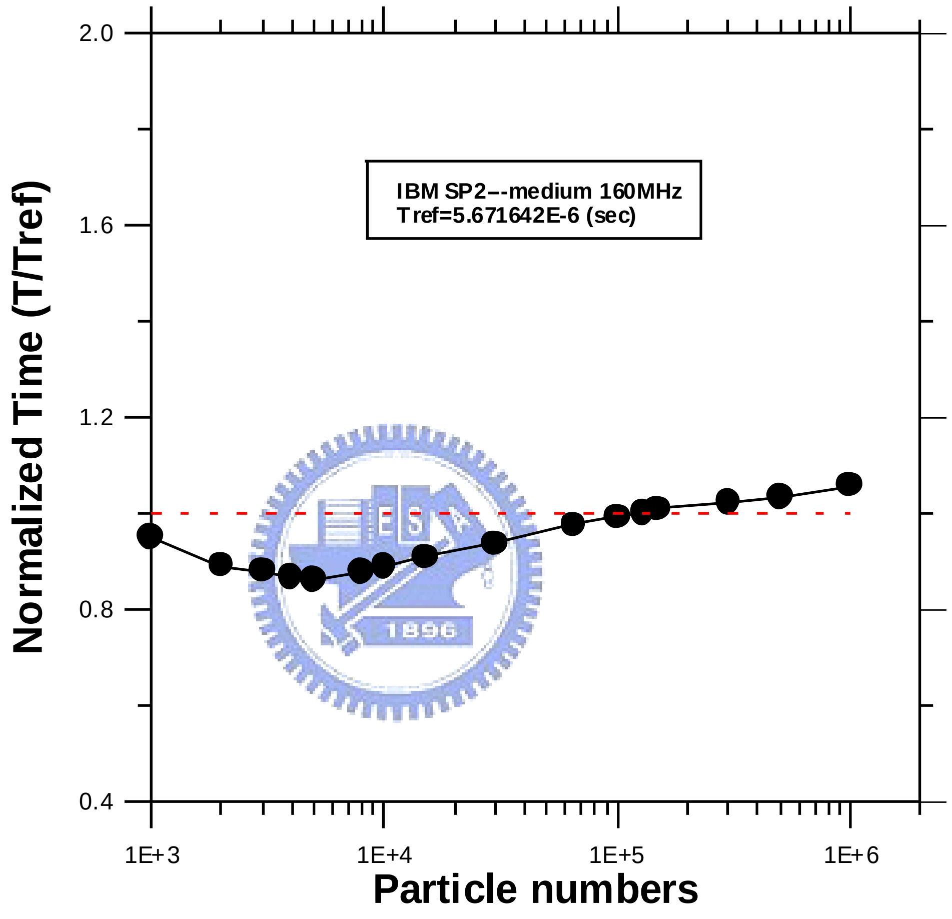
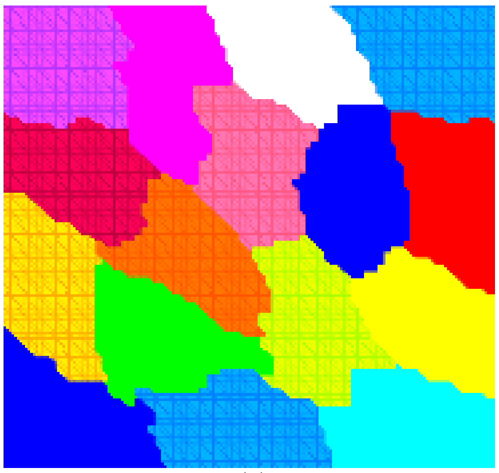
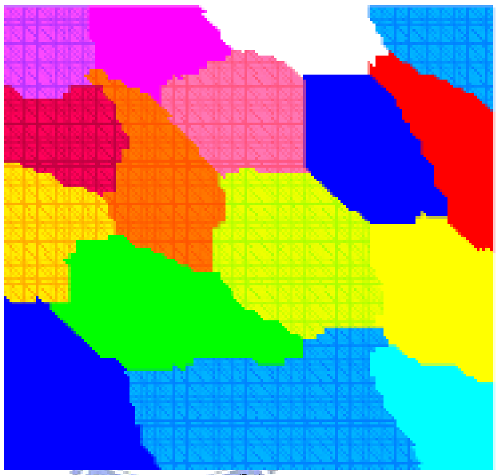


Figure 4. 7 Normalized computational time per particle on a single IBM-SP2 processor.


(a)


(b)

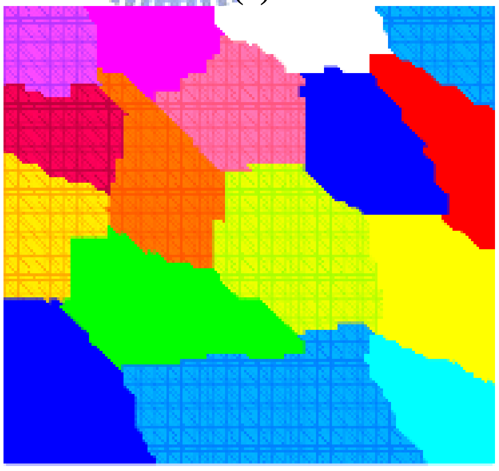

(c)

Figure 4. 8 Evolution of domain decomposition for large problem size using 64 processors, when activating SAR scheme at intervals of 2Δt, during the simulation for a bottom, lid-driven cavity flow (a) initial; (b) intermediate; (c) final.

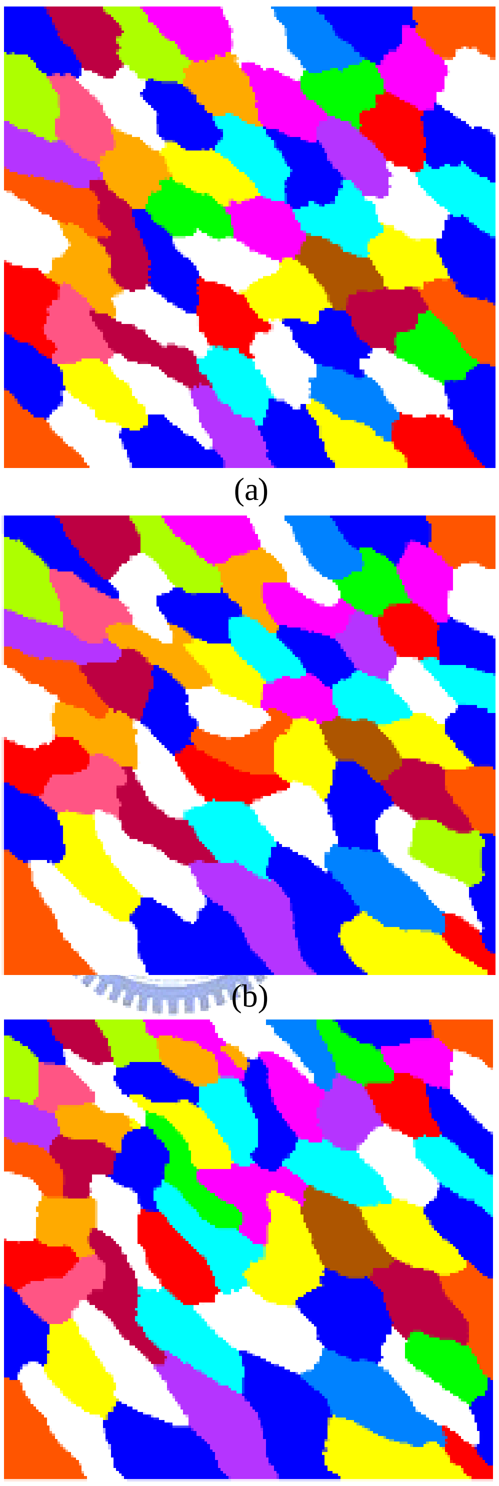


Figure 4. 9 Evolution of domain decomposition for large problem size using 64 processors, when activating SAR scheme at intervals of 2Δt, during the simulation for a bottom, lid-driven cavity flow. (a) initial; (b) intermediate; (c) final.

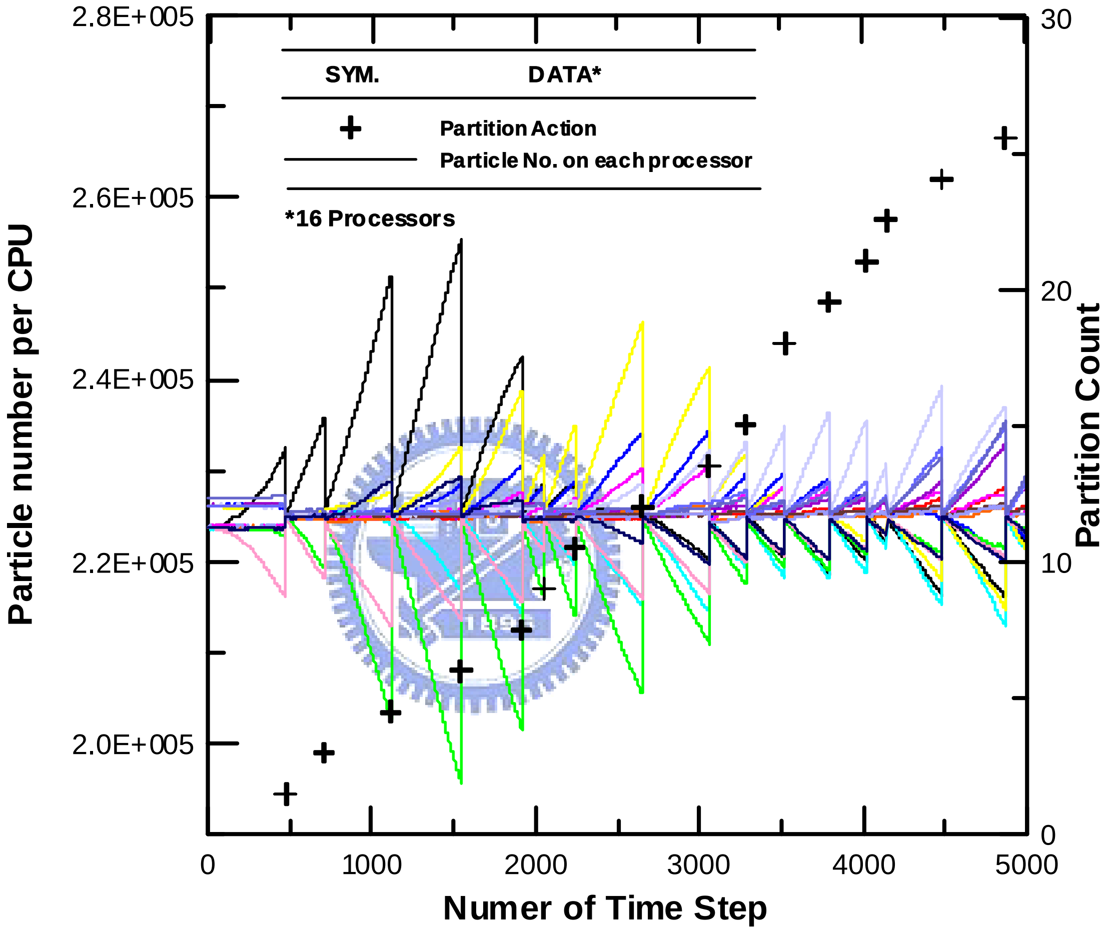


Figure 4. 10 Number of particles in each processor and the number of repartitions as a function of the number of simulation time-steps for the large problem size using 16 processors when activating SAR at intervals of 2Δt.

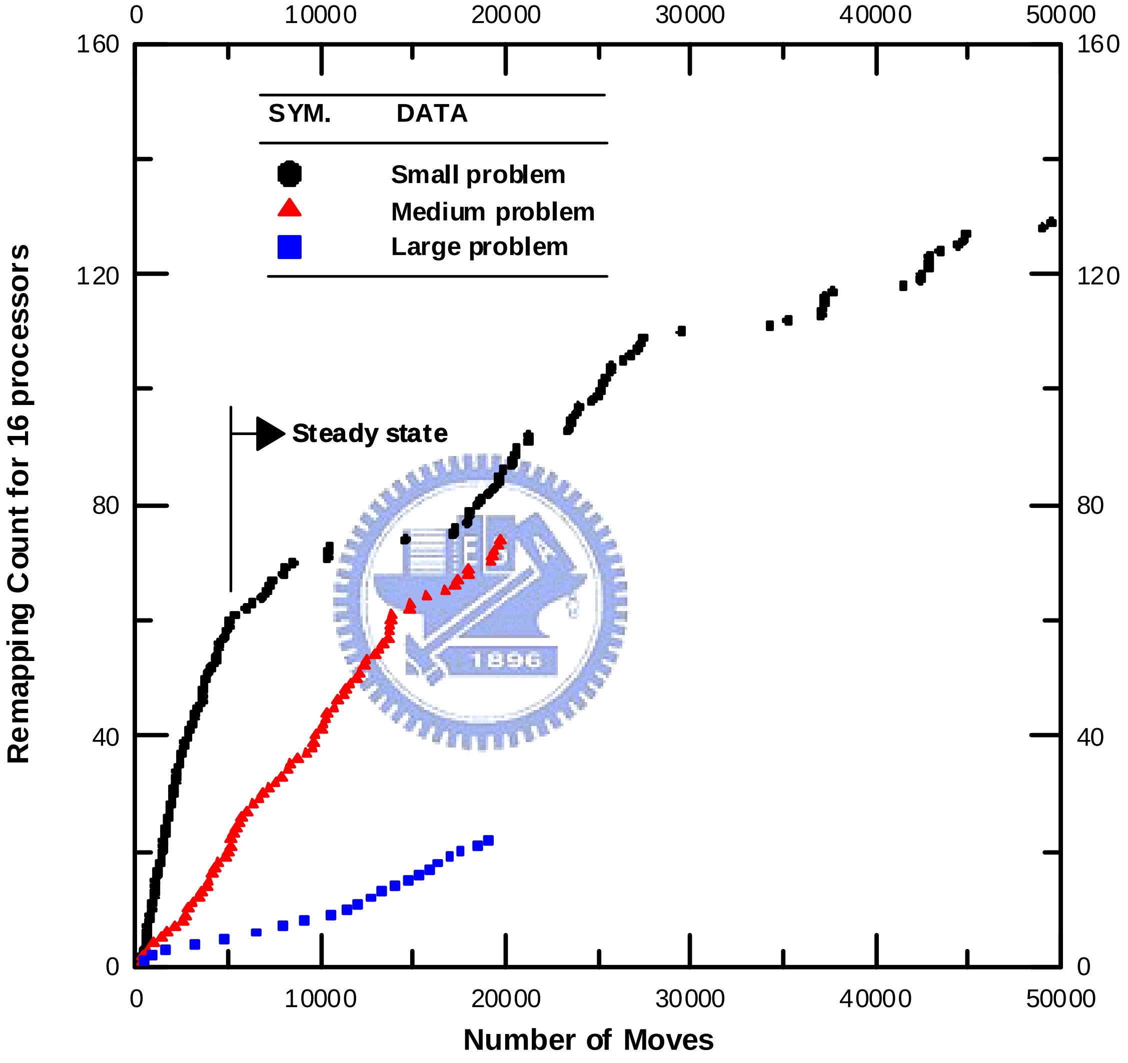


Figure 4. 11 Number of repartitions as a function of simulation time-steps at 16 processors.

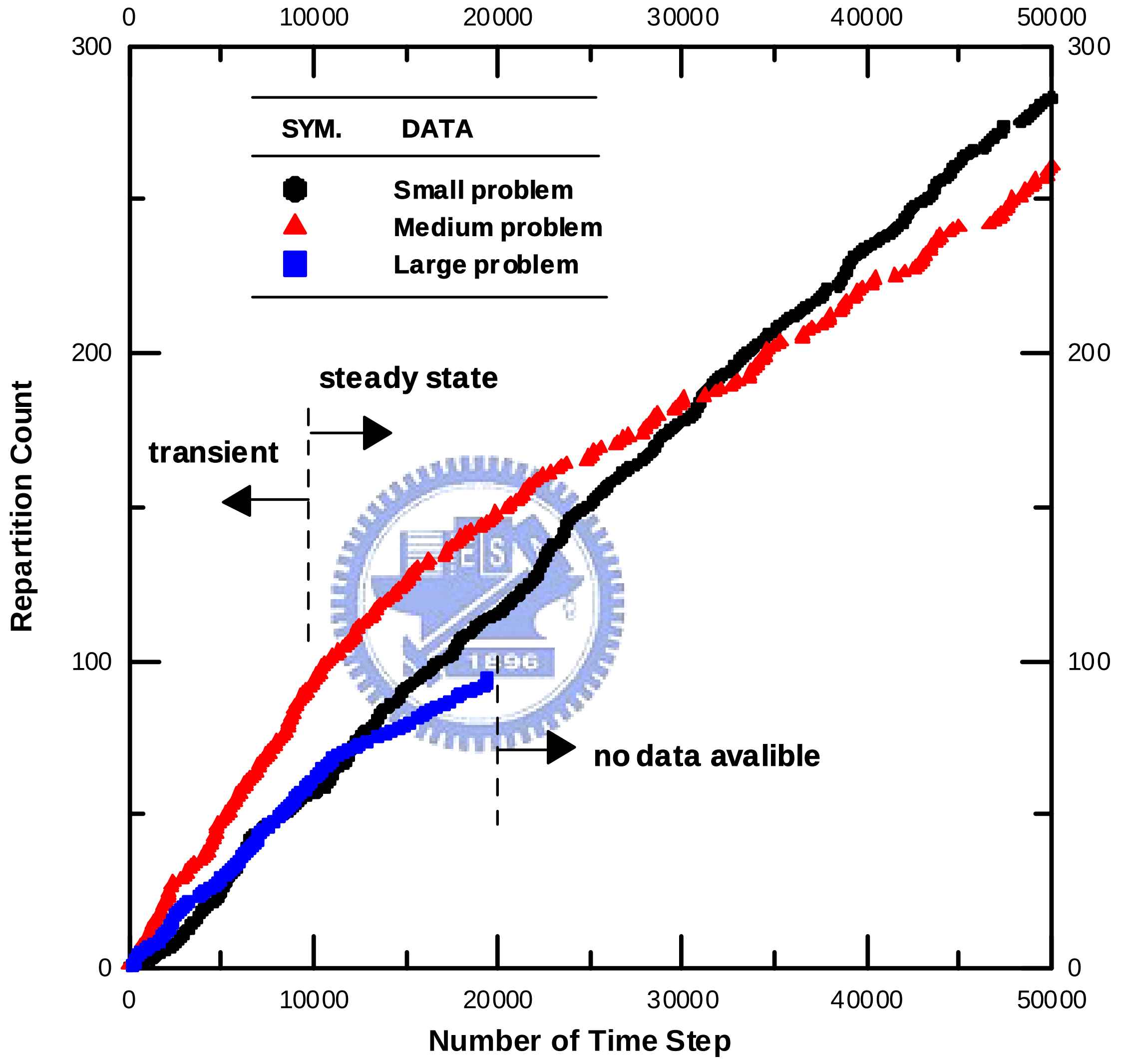


Figure 4. 12 Number of repartitions as a function of simulation time-steps at 64 processors.

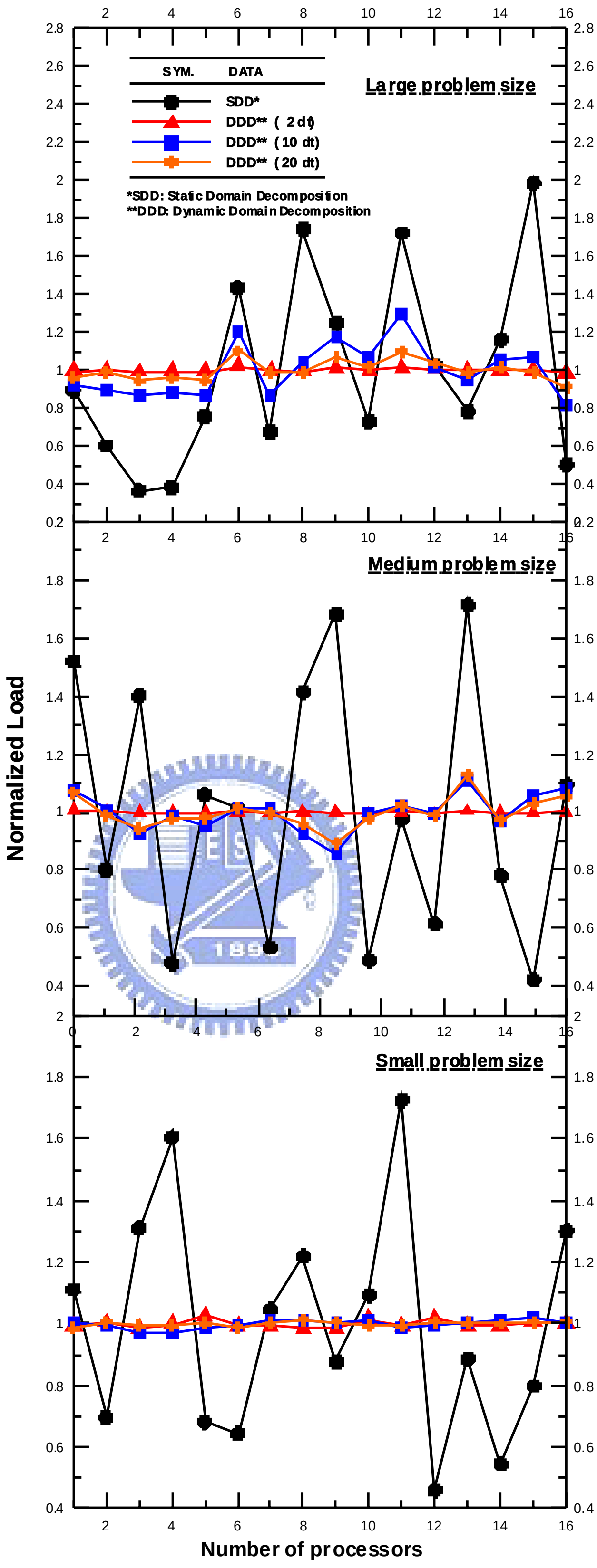


Figure 4. 13 Final normalized particle numbers on each processor for three problem sizes at 64 processors.

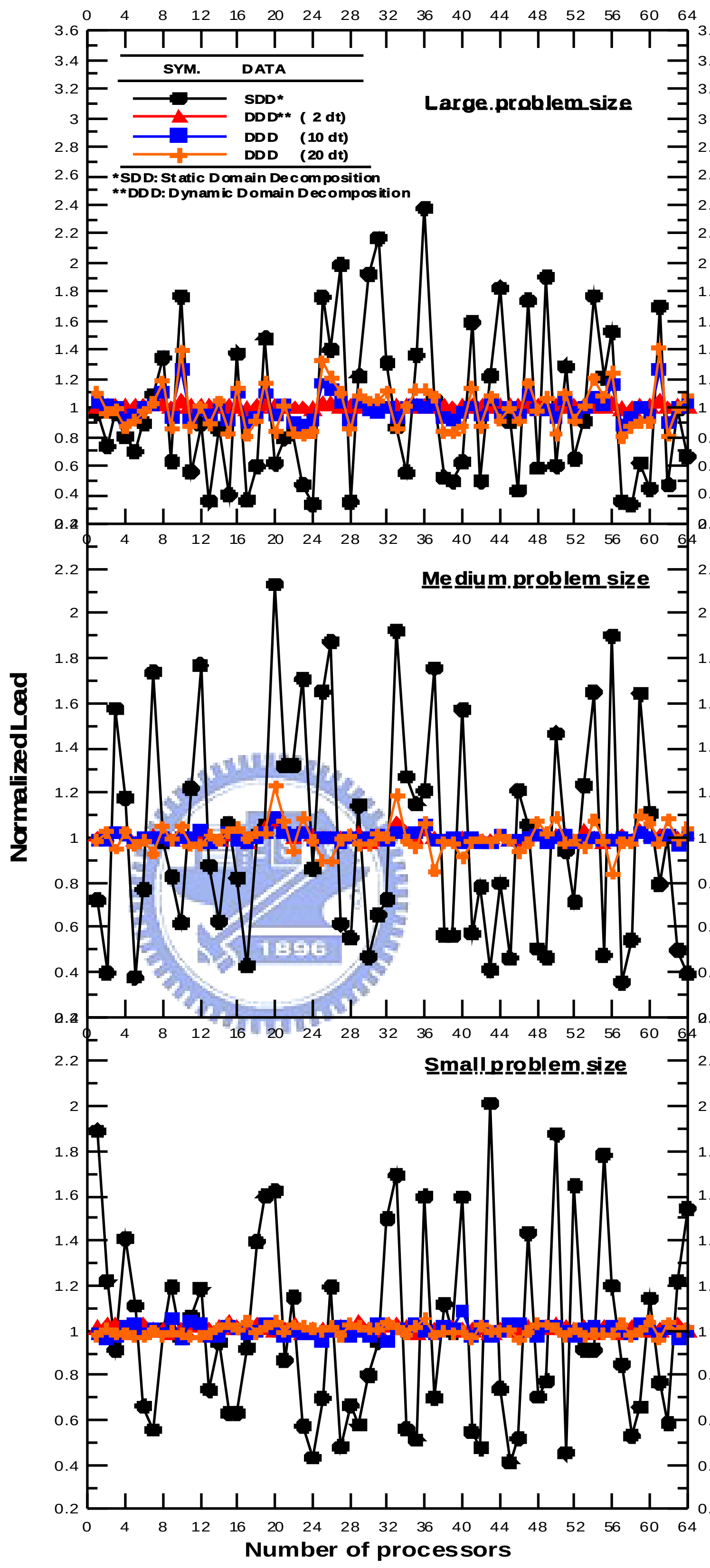


Figure 4. 14 Final normalized particle numbers on each processor for three problem sizes at 64 processors.

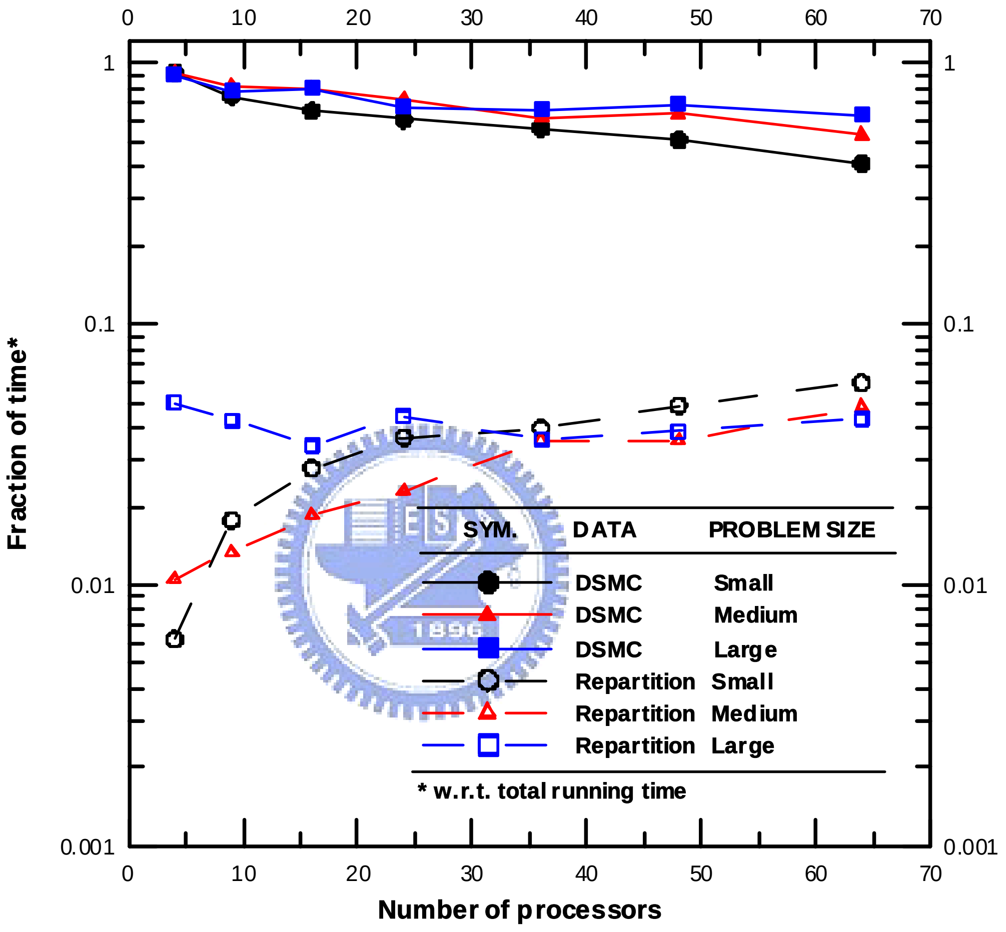


Figure 4. 15 Fraction of time for the DSMC computation and repartition as a function of number of processors.

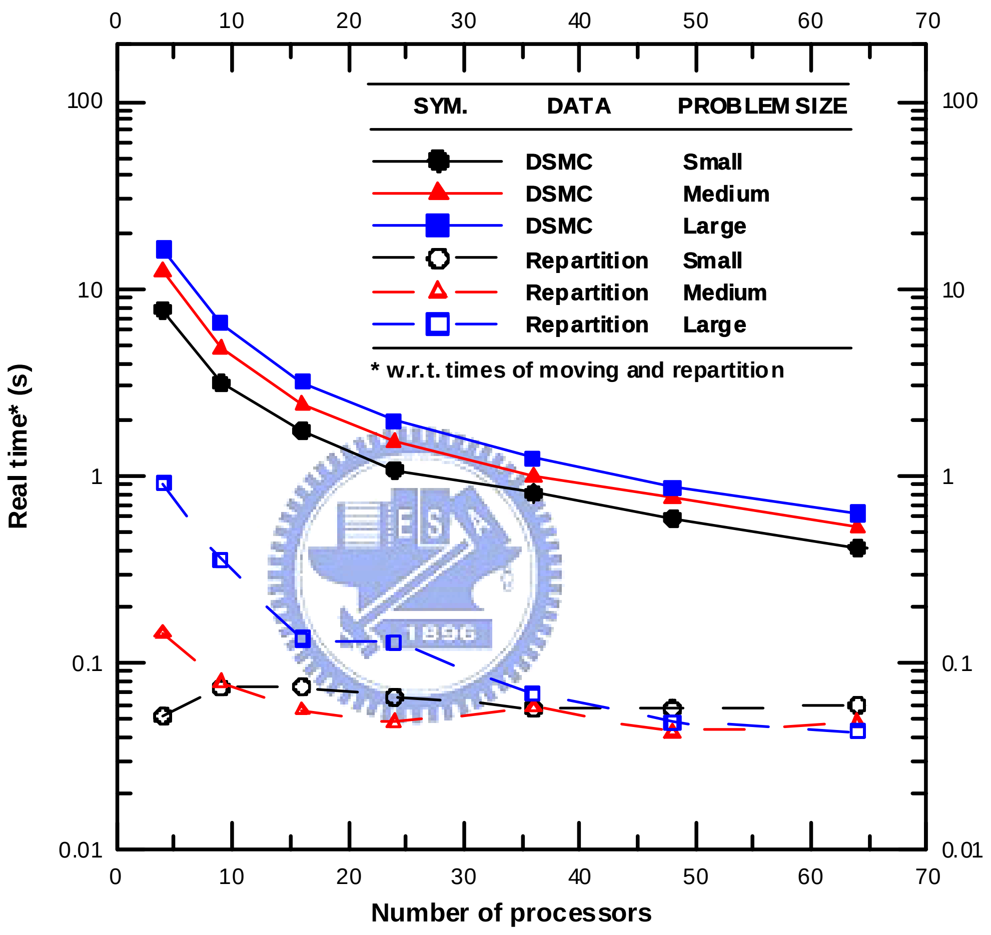
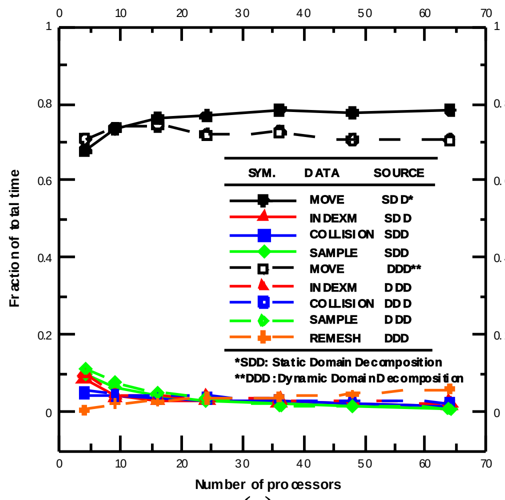
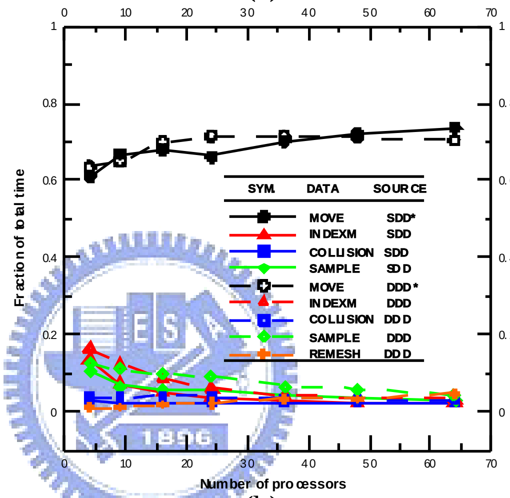


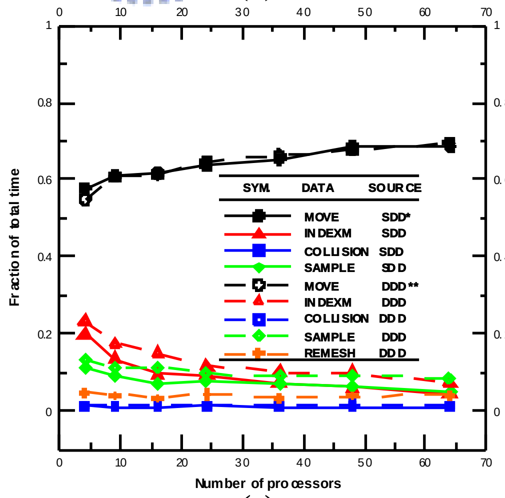
Figure 4. 16 Real CPU running time of doing useful DSMC work and repartition per time with different problem size.



(a)



(b)



(c)

Figure 4. 17 Relative costs within DSMC with different problem size on IBM-SP2 (a) small problem; (b) medium problem; (c) large problem.

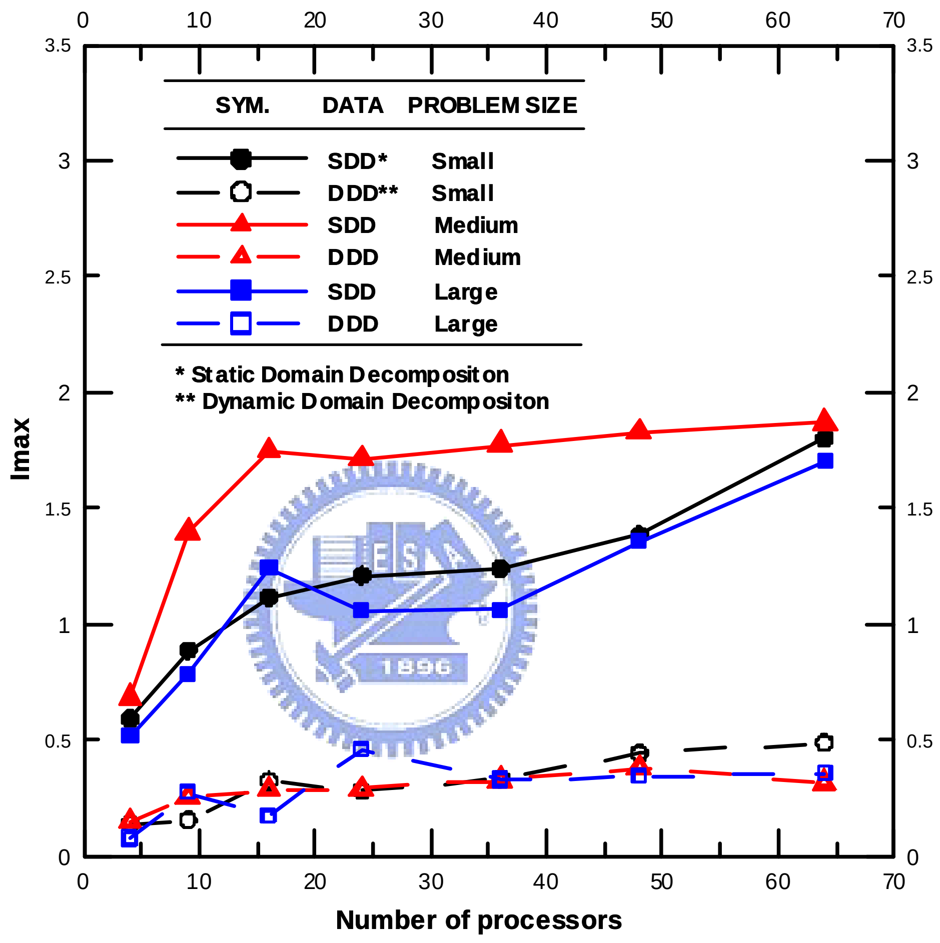
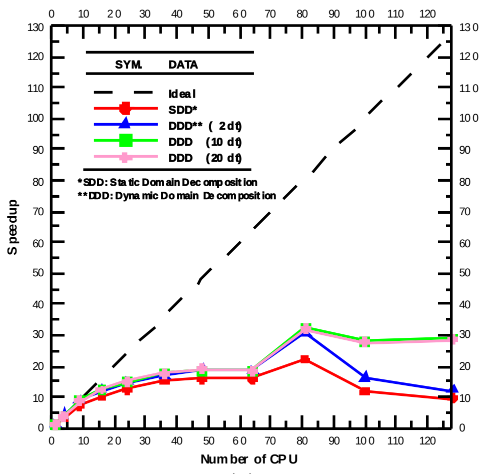
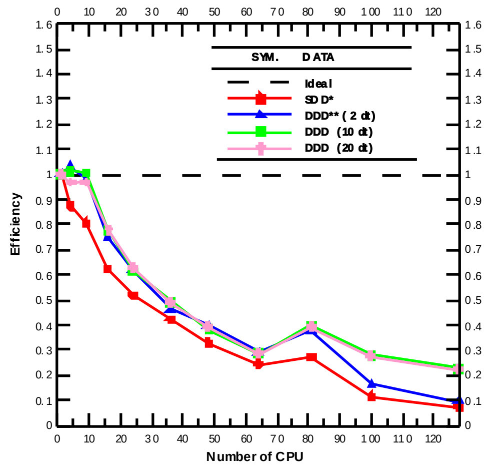


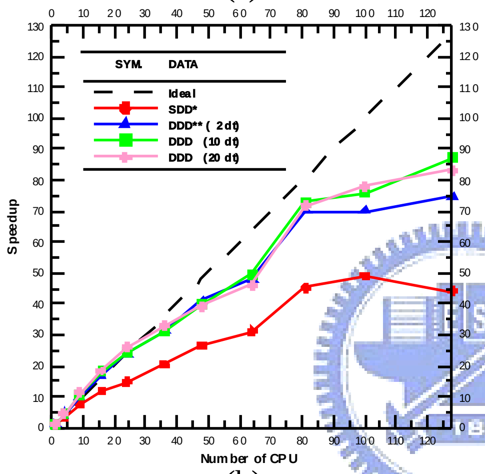
Figure 4. 18 Degree of imbalance as a function of number of processors for static and dynamic domain decomposition.



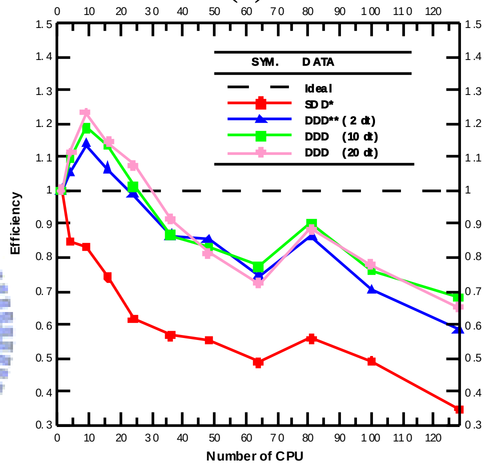
(a)



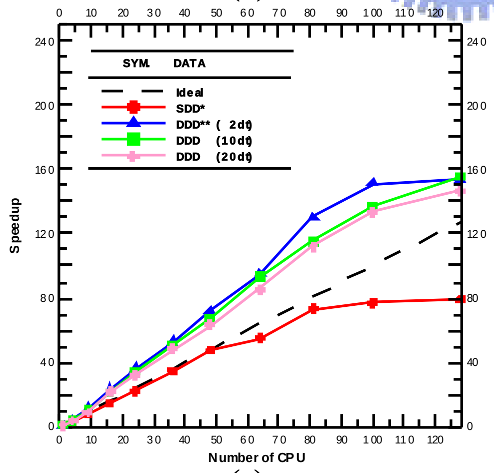
(d)



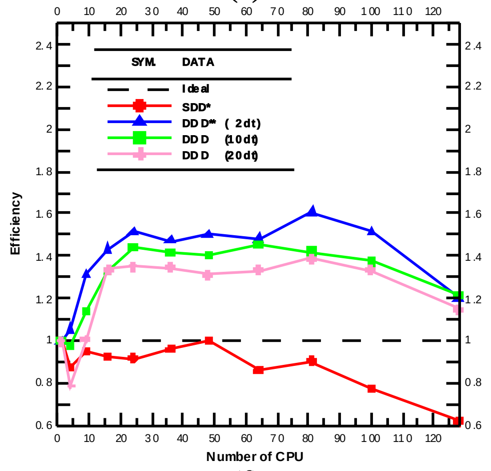
(b)



(e)

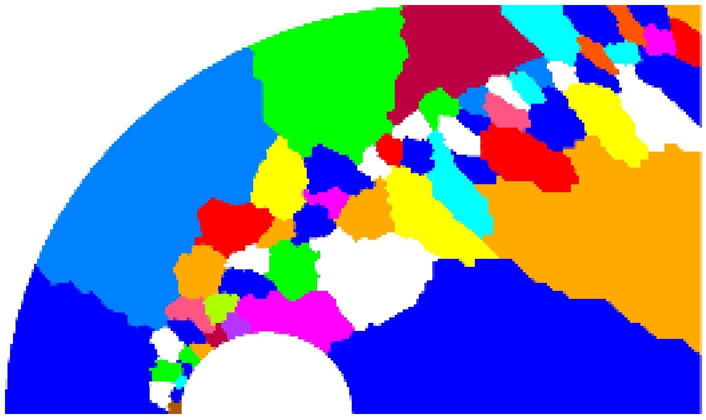


(c)

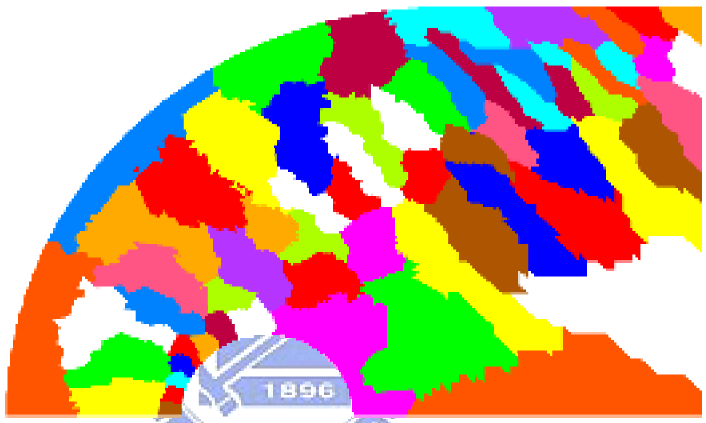


(f)

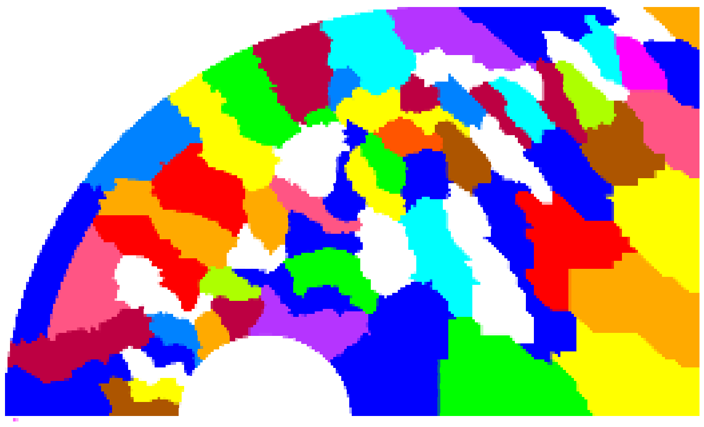
Figure 4. 19 Parallel speedup and efficiency as a function of number of processors for high-speed driven cavity flow at different problem sizes on IBM-SMP machine (maximum 128 processors).



(a)



(b)



(c)

Figure 4. 20 Partition development for a hypersonic cylinder flow (64 CPUs) (a) initial; (b) medium; (c) final.

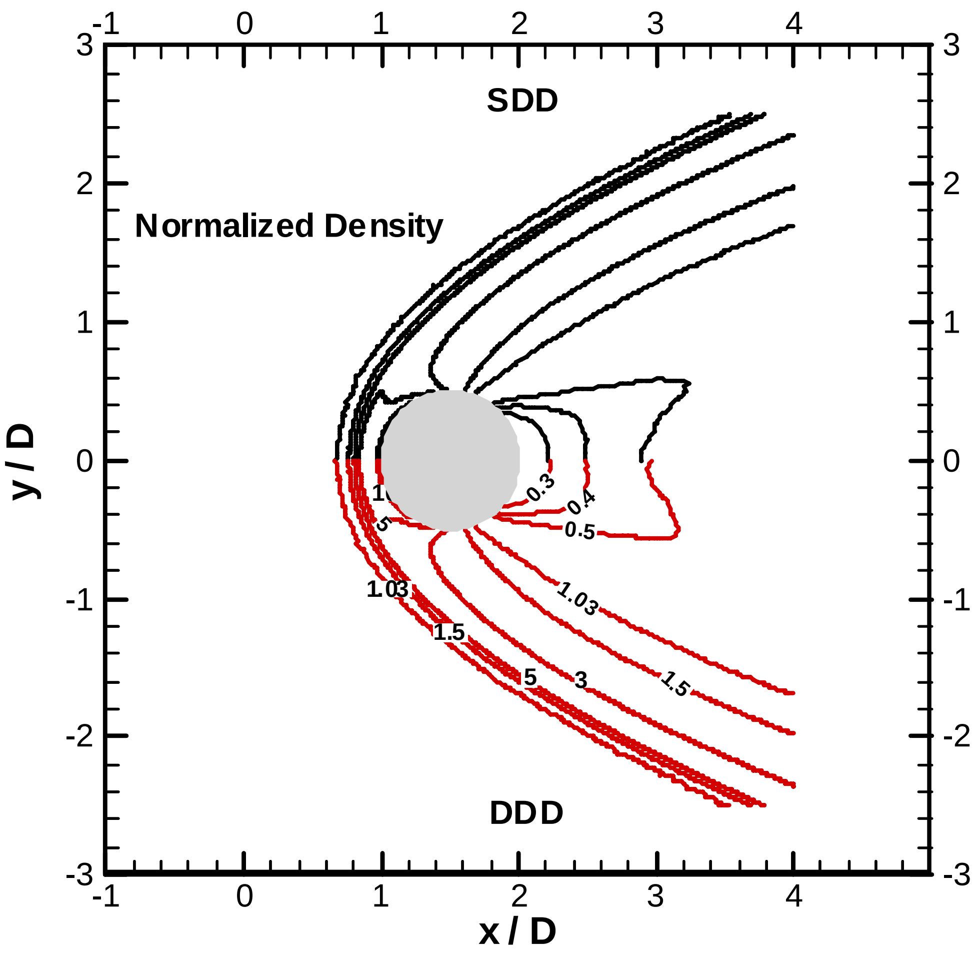
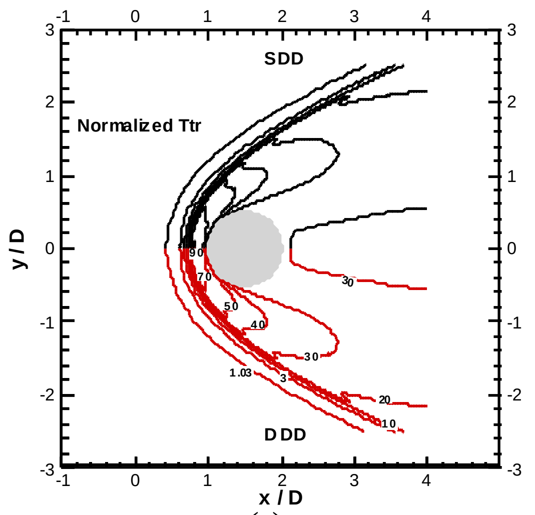
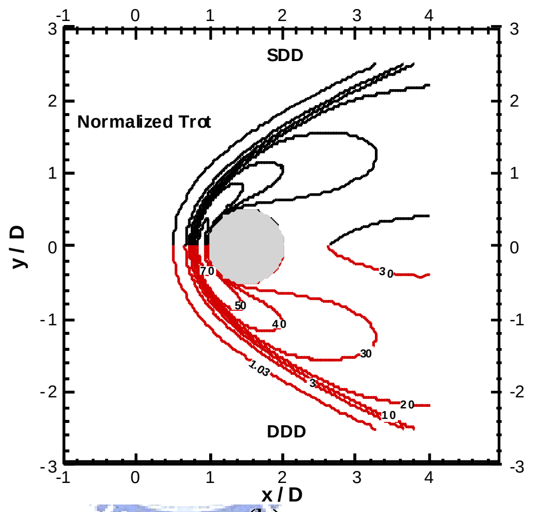


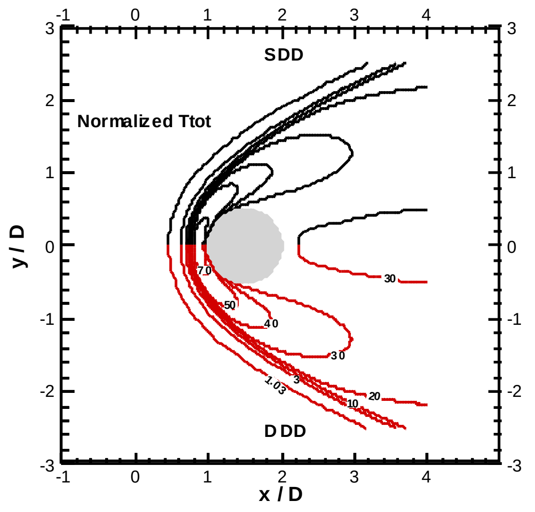
Figure 4. 21 Comparison of the normalized density contour of a hypersonic cylinder flow using static and dynamic domain decomposition methods.



(a)

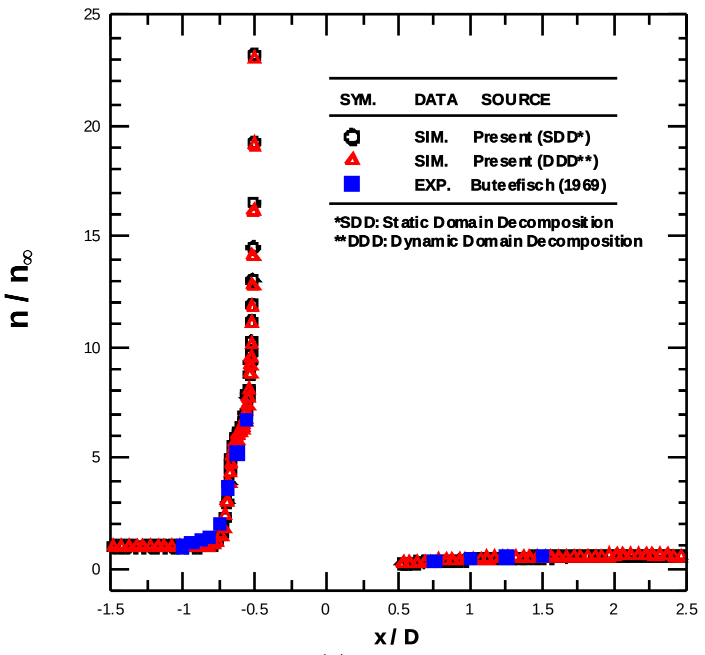


(b)

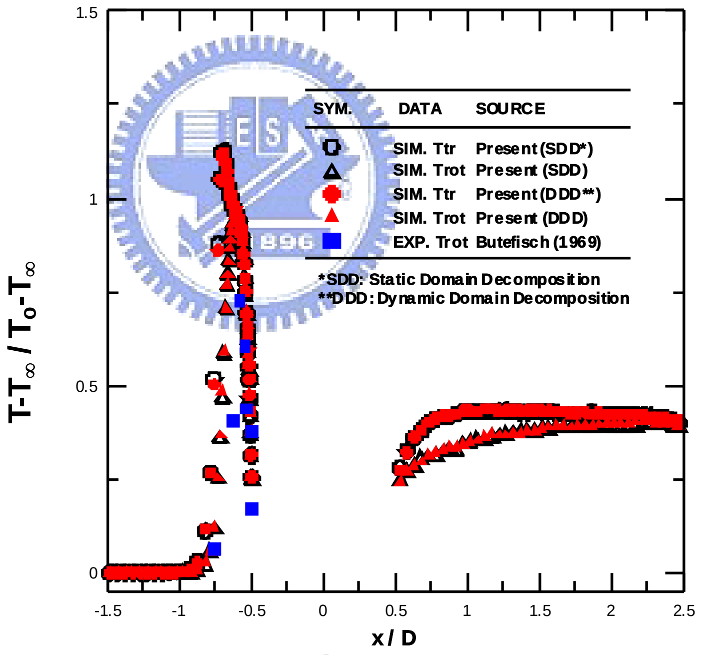


(c)

Figure 4. 22 Comparison of the normalized temperature contours of a hypersonic cylinder flow using static and dynamic domain decomposition methods.

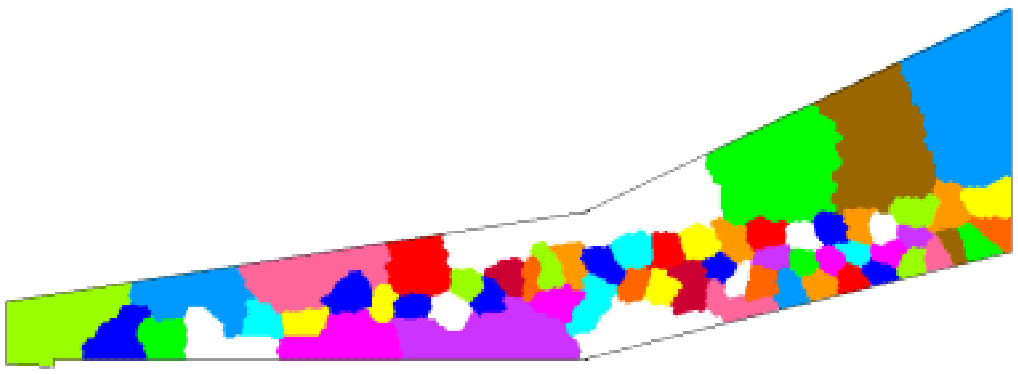


(a)

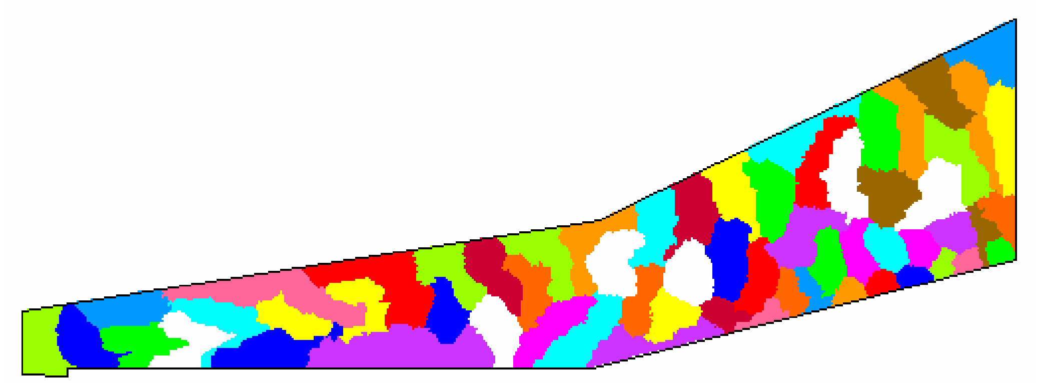


(b)

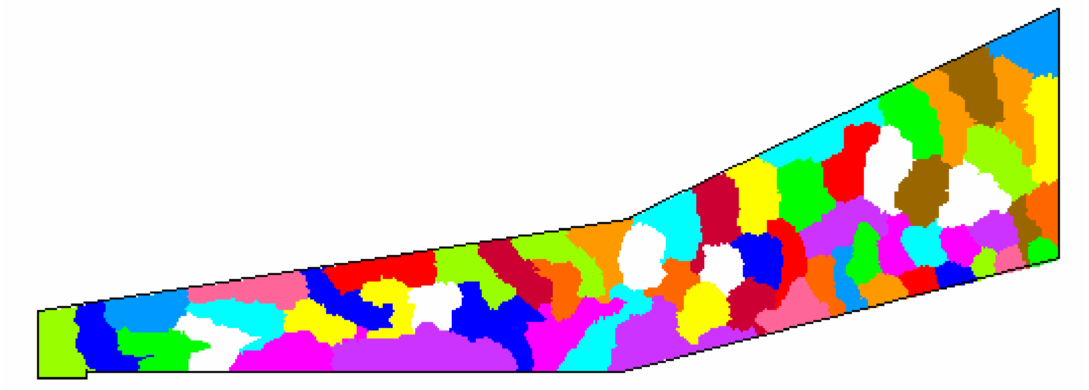
Figure 4. 23 Normalized density and temperatures of a hypersonic cylinder flow along the stagnation line using static and dynamic domain decomposition methods.



(a)



(b)



(c)

Figure 4. 24 Partition development for a 15^{o}-compression ramp flow (64 CPUs) (a) initial; (b) medium; (c) final.

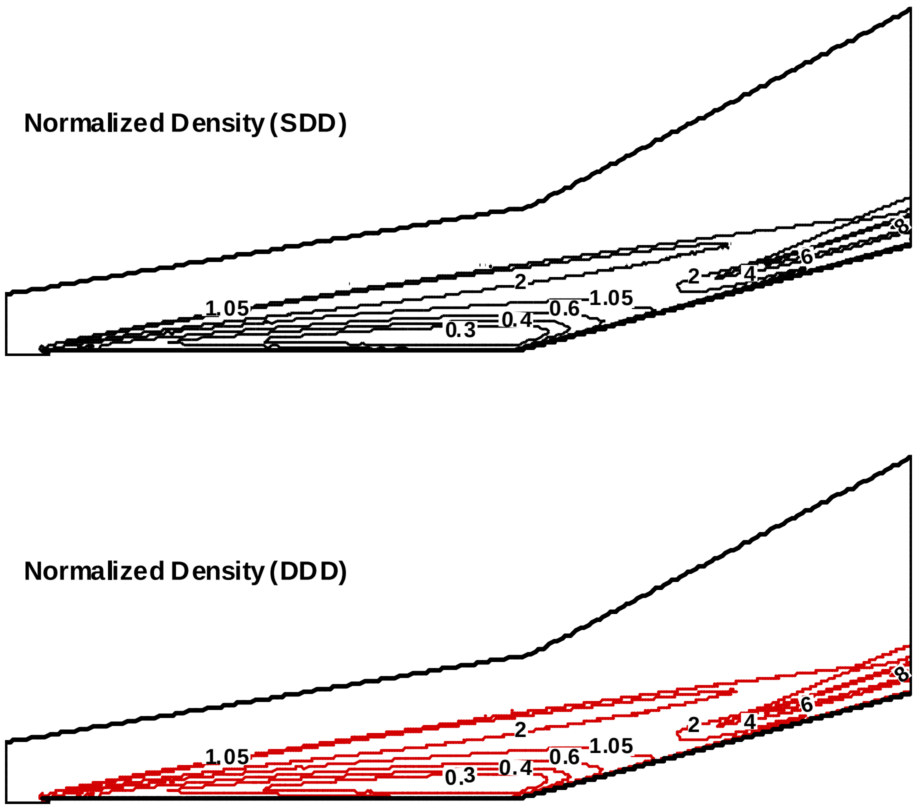


Figure 4. 25 Comparison of normalized density contour of a hypersonic 15^{o}-compression ramp flow using static and dynamic domain decomposition methods.

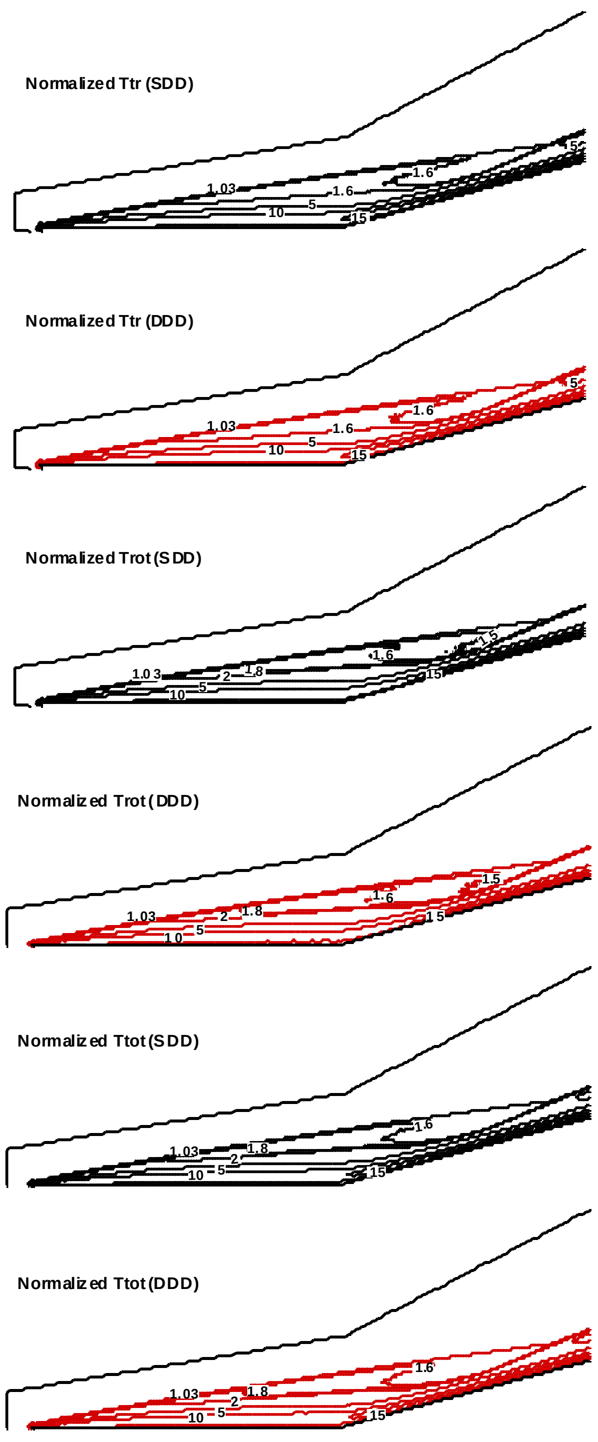
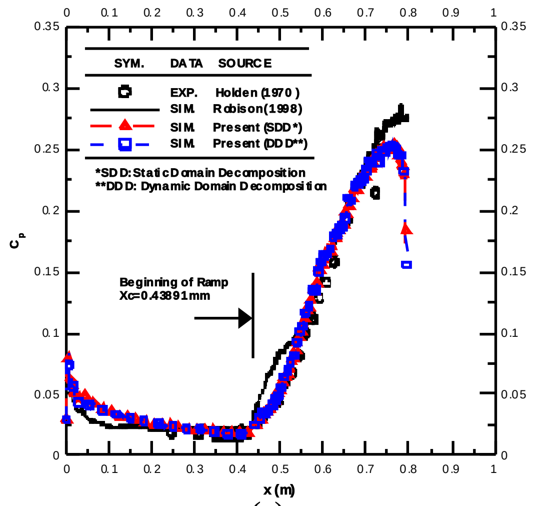
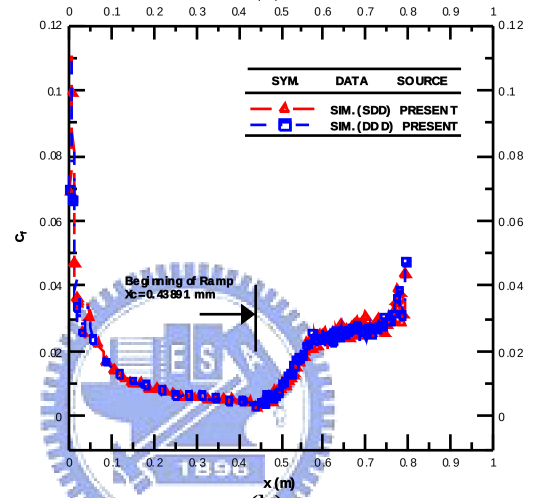


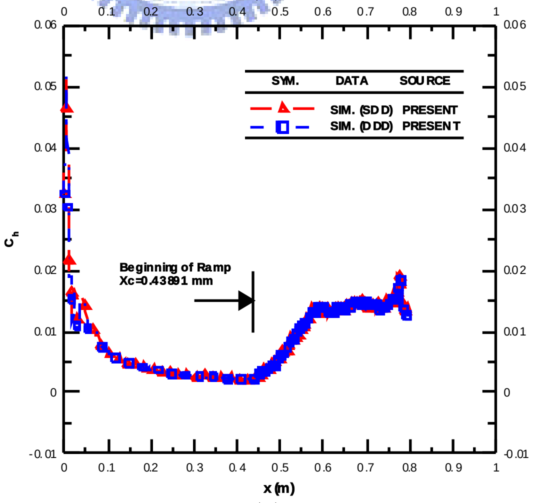
Figure 4. 26 Comparison of normalized temperature contours of a hypersonic 15^{o}-compression ramp flow using static and dynamic domain decomposition methods.



(a)

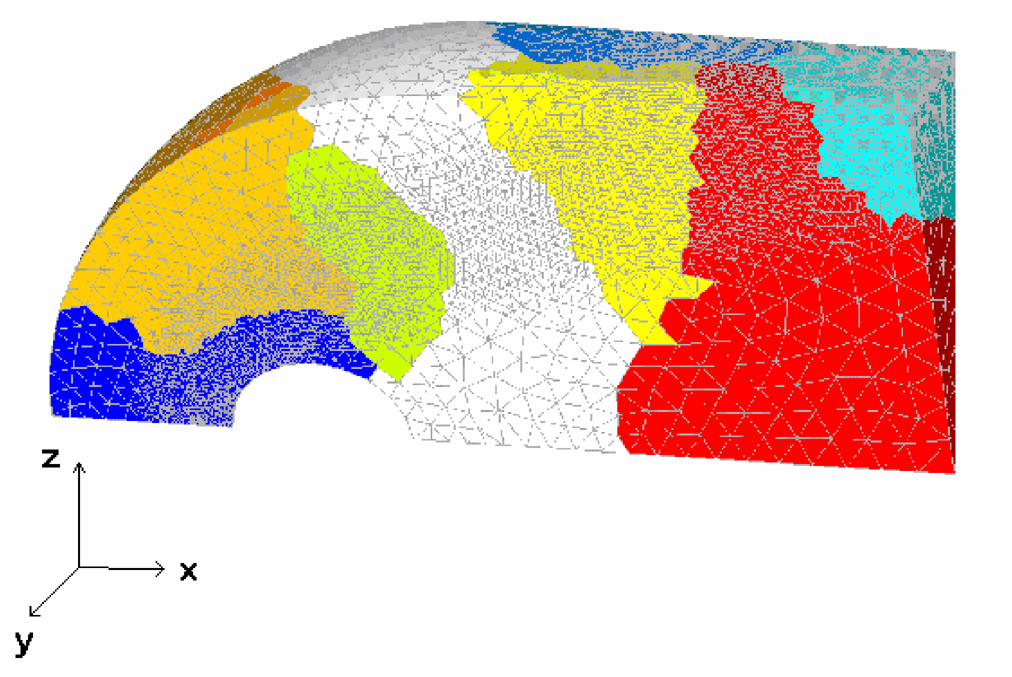


(b)

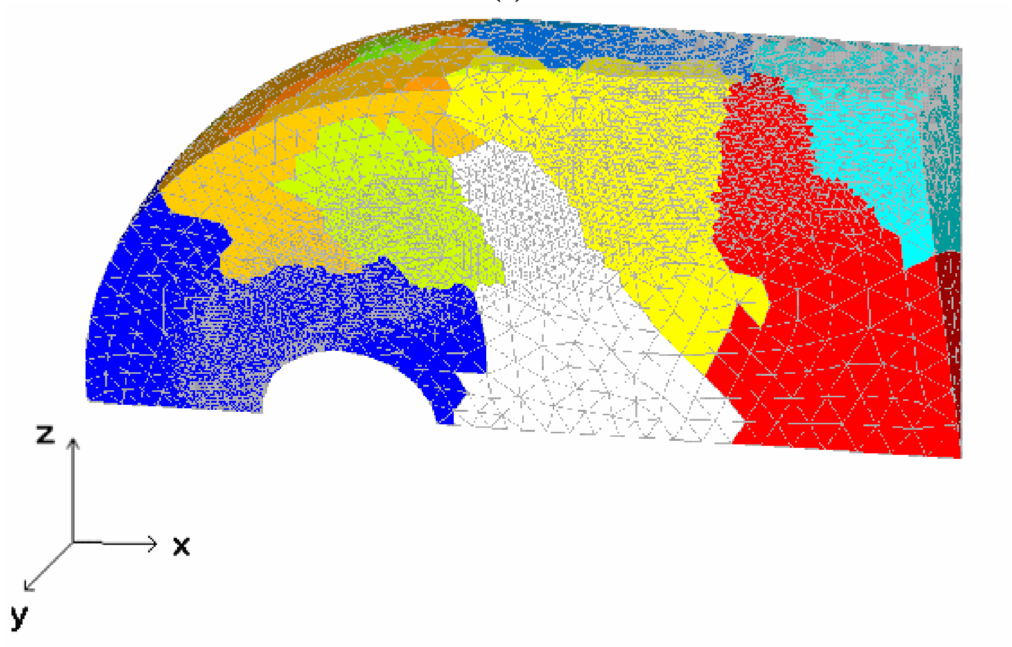


(c)

Figure 4. 27 Comparison of the coefficients along the solid wall of a hypersonic 15^{o}-compression ramp flow using static and dynamic domain decomposition methods (a) pressure; (b) shear stress; (c) heat transfer.



(a)



(b)

Figure 4. 28 Development of partition of a supersonic sphere flow using dynamic domain decomposition method on the surface of simulation domain.

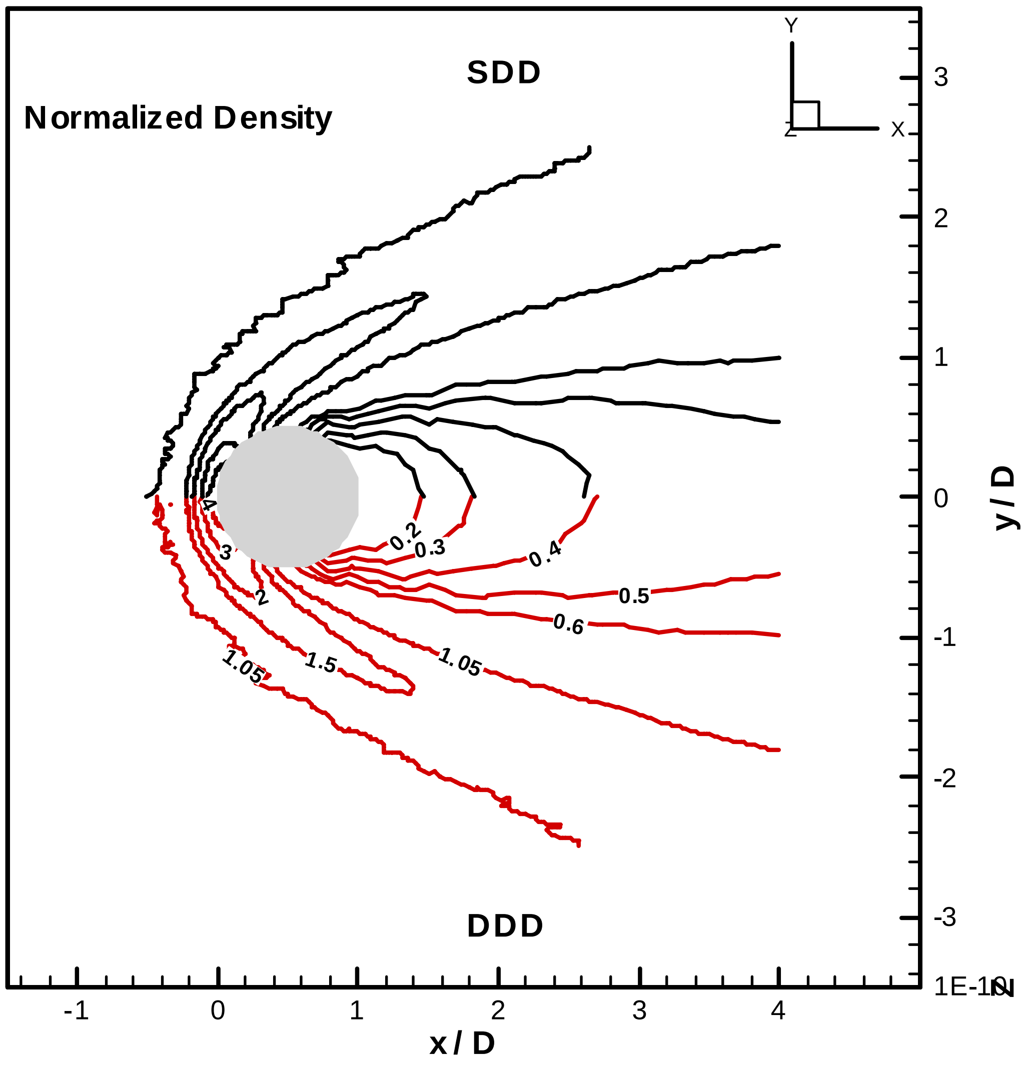
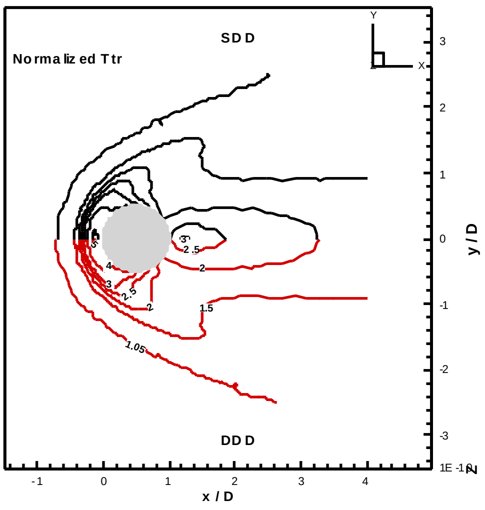
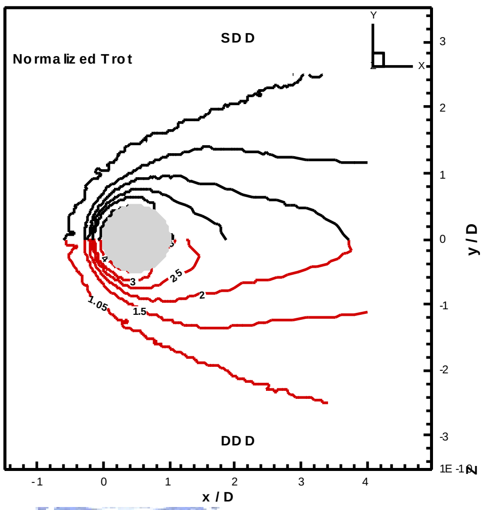


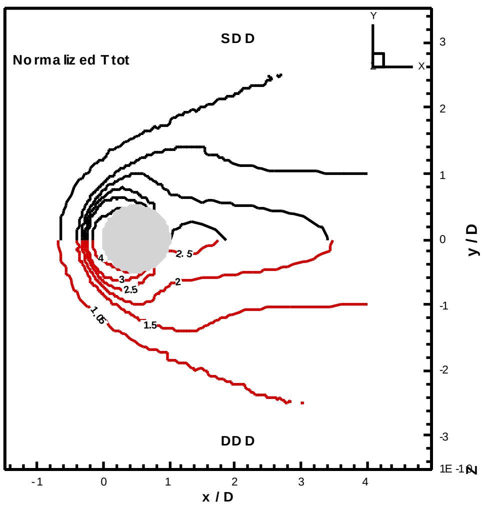
Figure 4. 29 Comparison of normalized density contour of a hypersonic sphere flow by using static and dynamic domain decomposition methods.



(a)



(b)



(c)

Figure 4. 30 Comparison of normalized temperature contours of a hypersonic sphere flow by using static and dynamic domain decomposition methods.

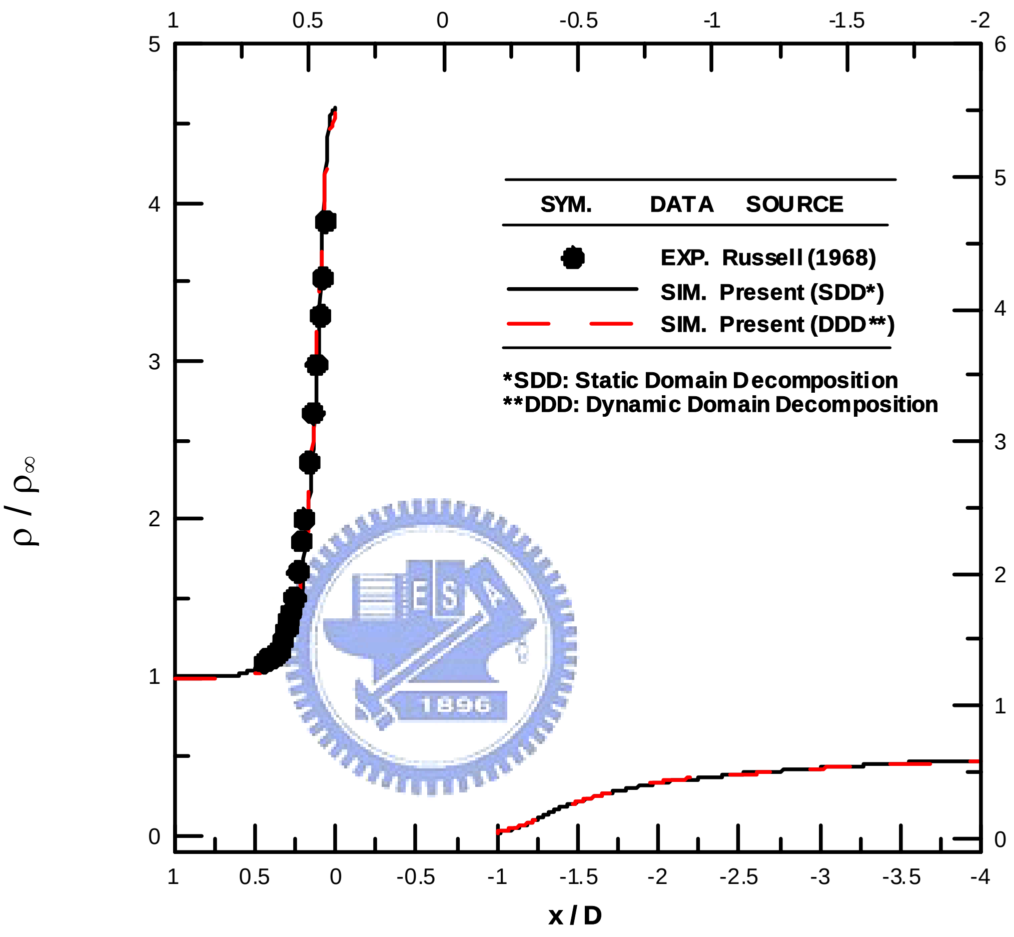


Figure 4. 31 Comparison of normalized density along the stagnation line of a hypersonic sphere flow by using static and dynamic domain decomposition methods.

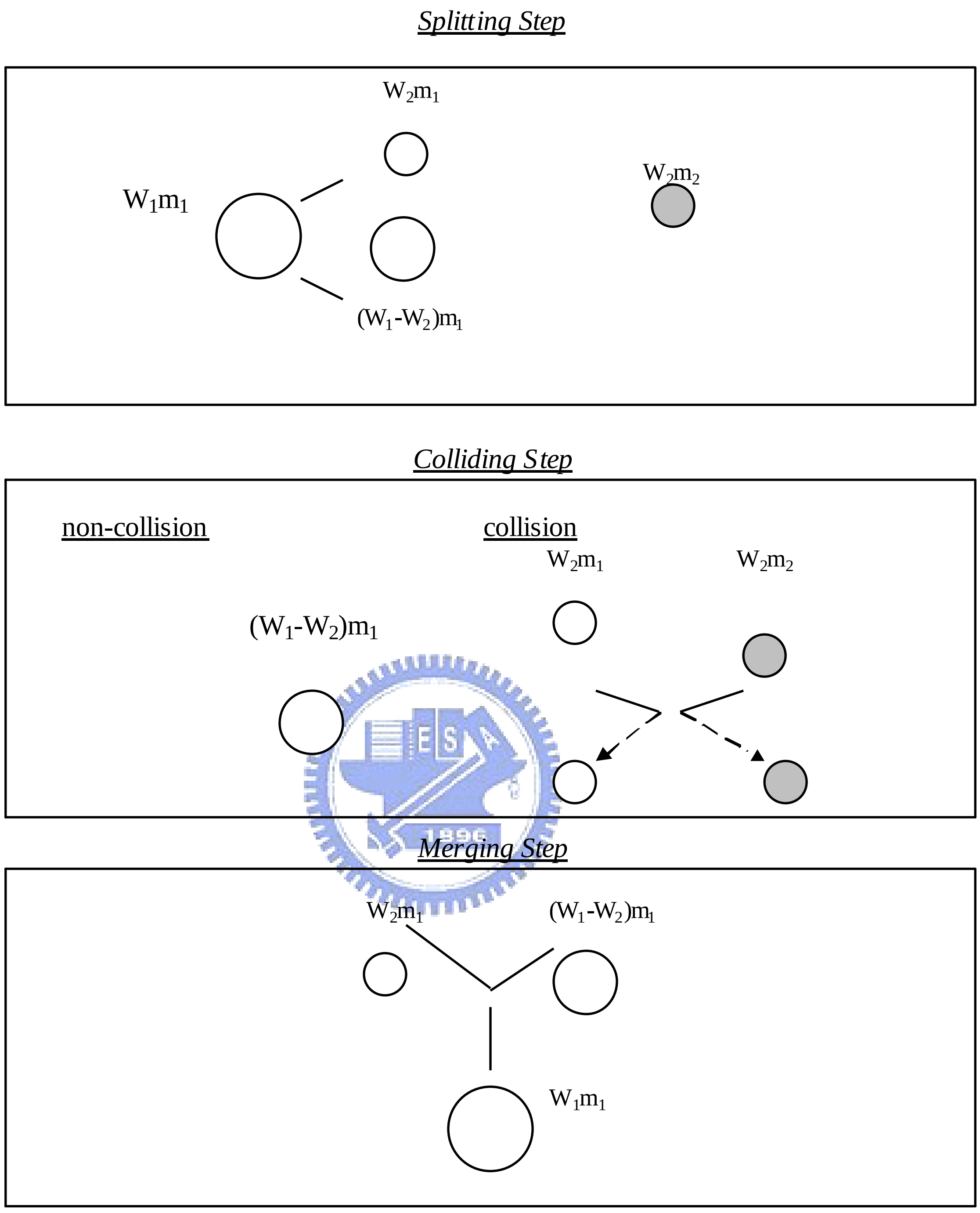
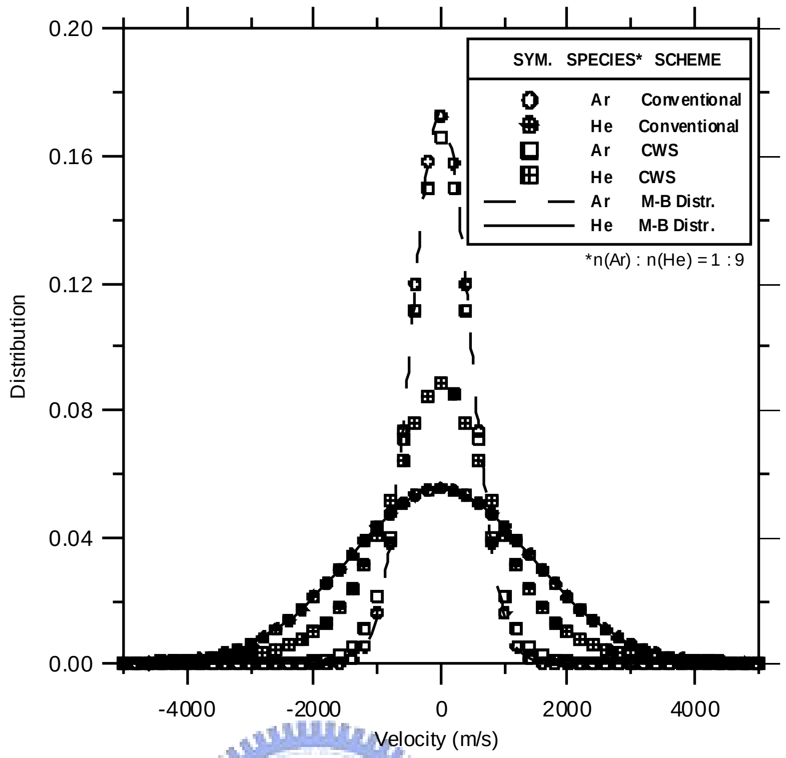
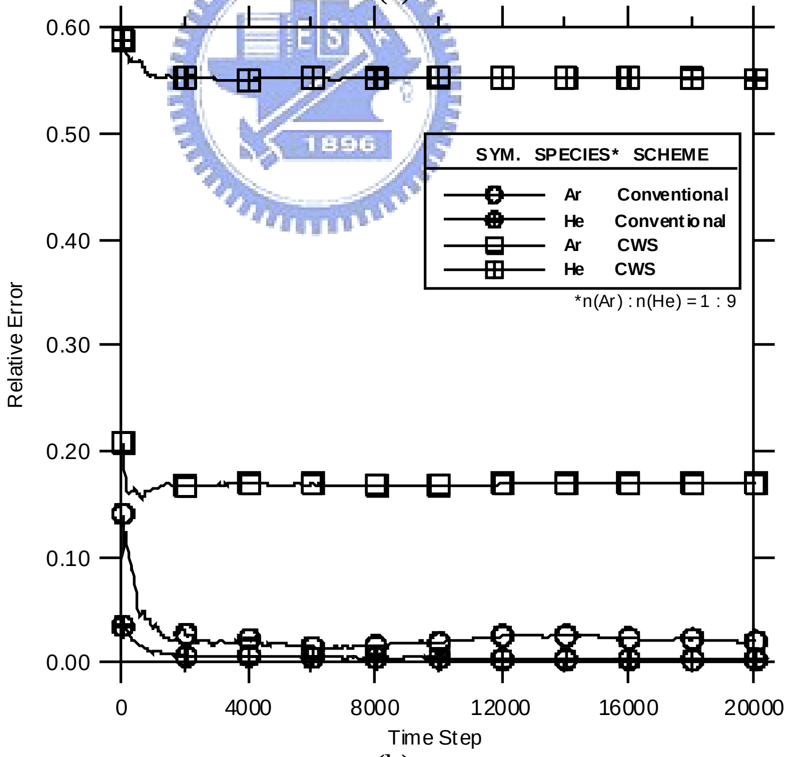


Figure 5. 1 Schematic diagram of CWS for non-reactive flow.

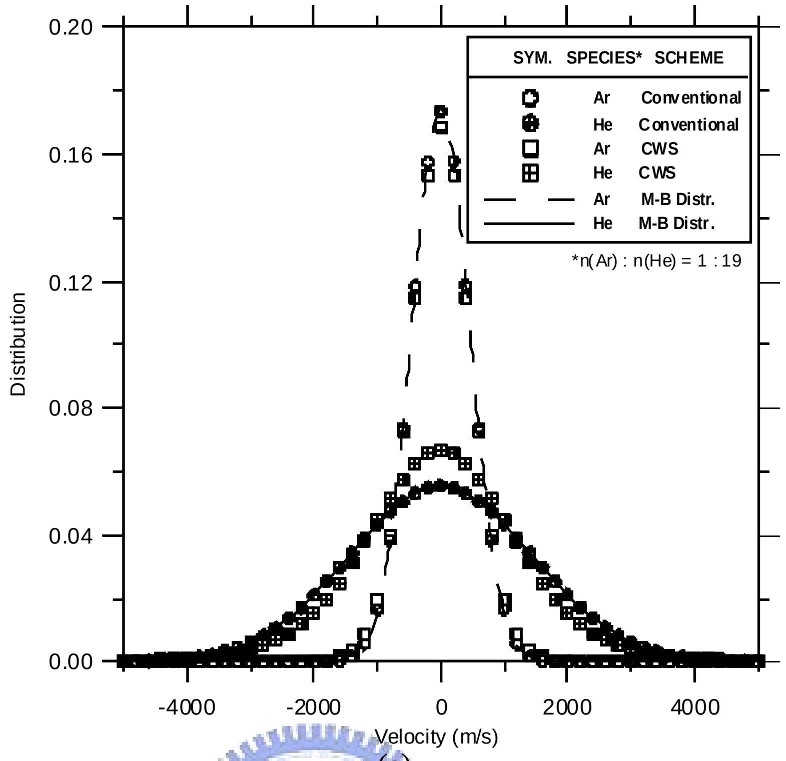


(a)

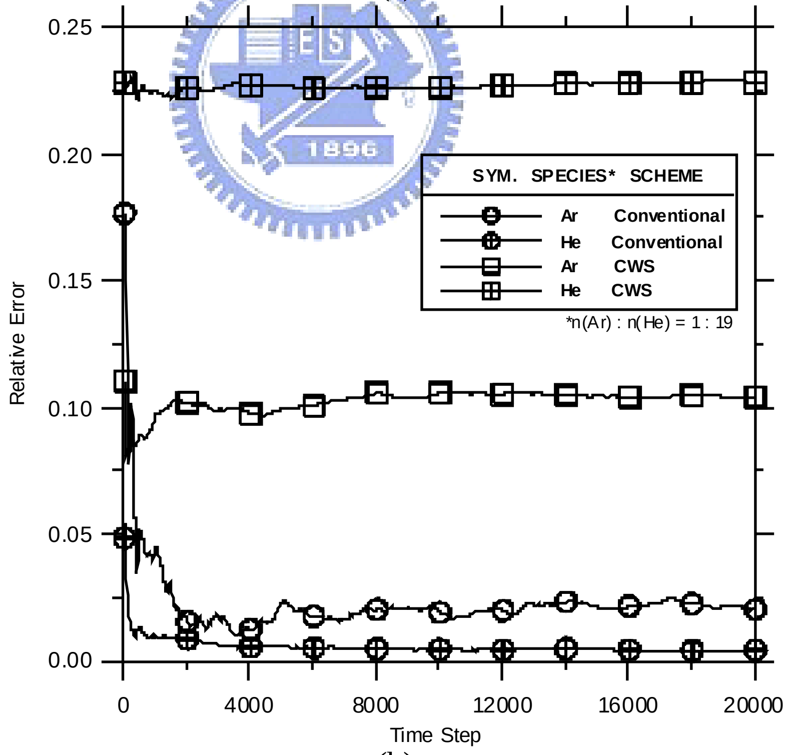


(b)

Figure 5. 2 Simulation of two-component with weight ratio 1:9 (a) velocity distribution; (b) relative error as a function of the number of simulation time-steps.

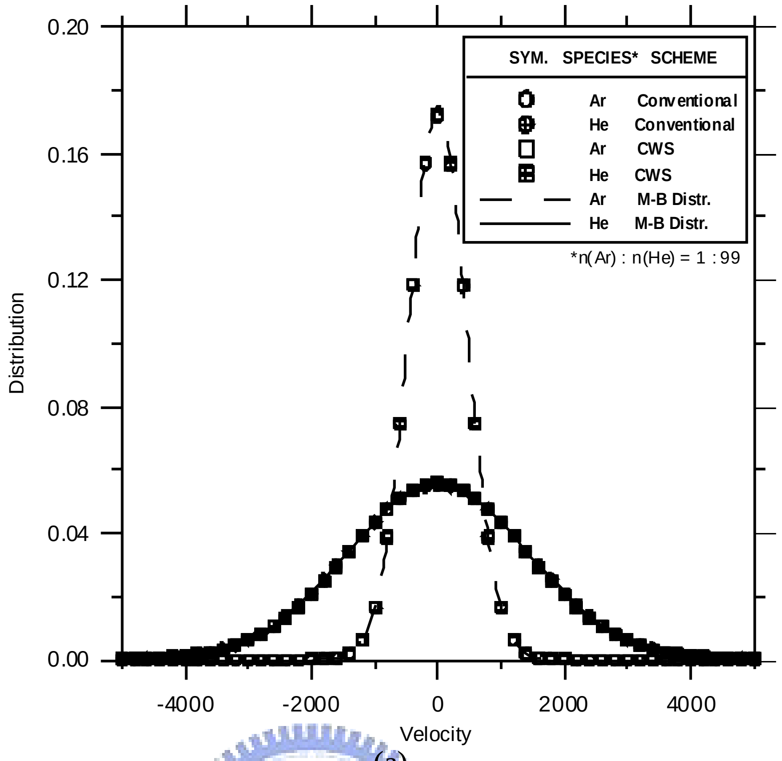


(a)

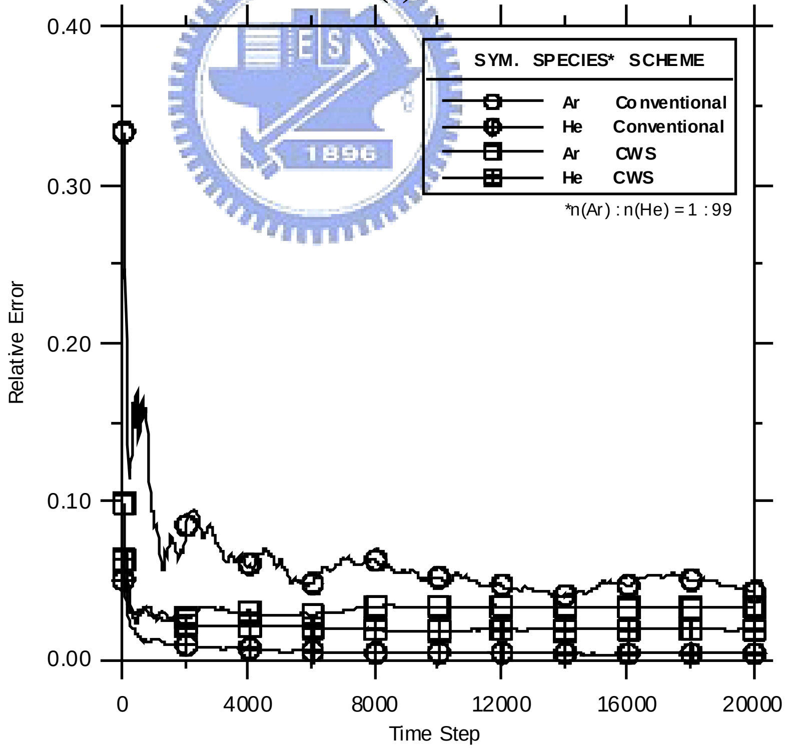


(b)

Figure 5. 3 Simulation of two-component with weight ratio 1:19 (a) velocity distribution; (b) relative error as a function of the number of simulation time-steps.

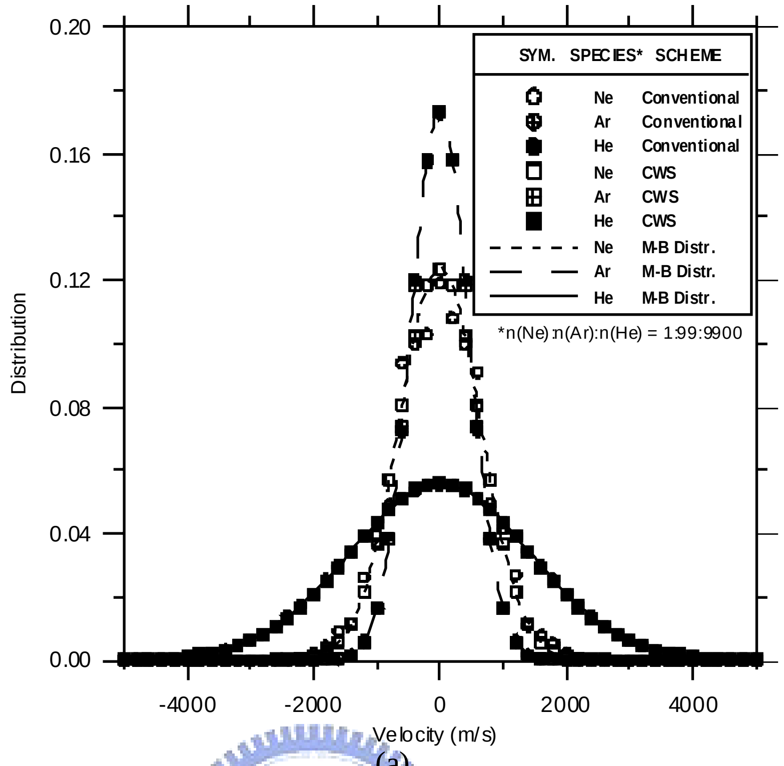


(a)

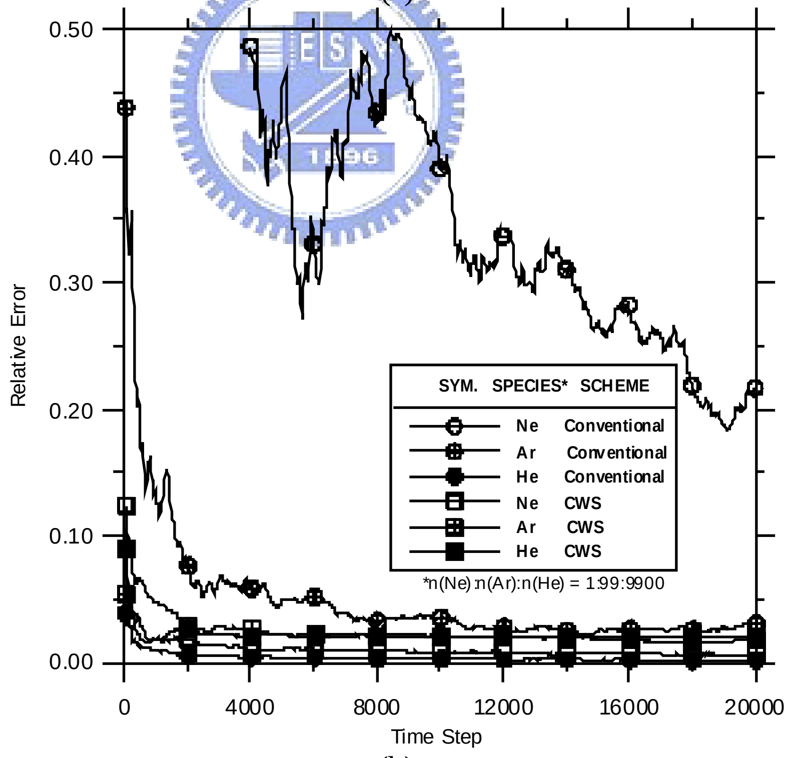


(b)

Figure 5. 4 Simulation of two-component with weight ratio 1:99 (a) velocity distribution; (b) relative error as a function of the number of simulation time-steps..



(a)



(b)

Figure 5. 5 Simulation of three-component with weight ratio 1:99:9900 (a) velocity distribution; (b) relative error as a function of the number of simulation time-steps.

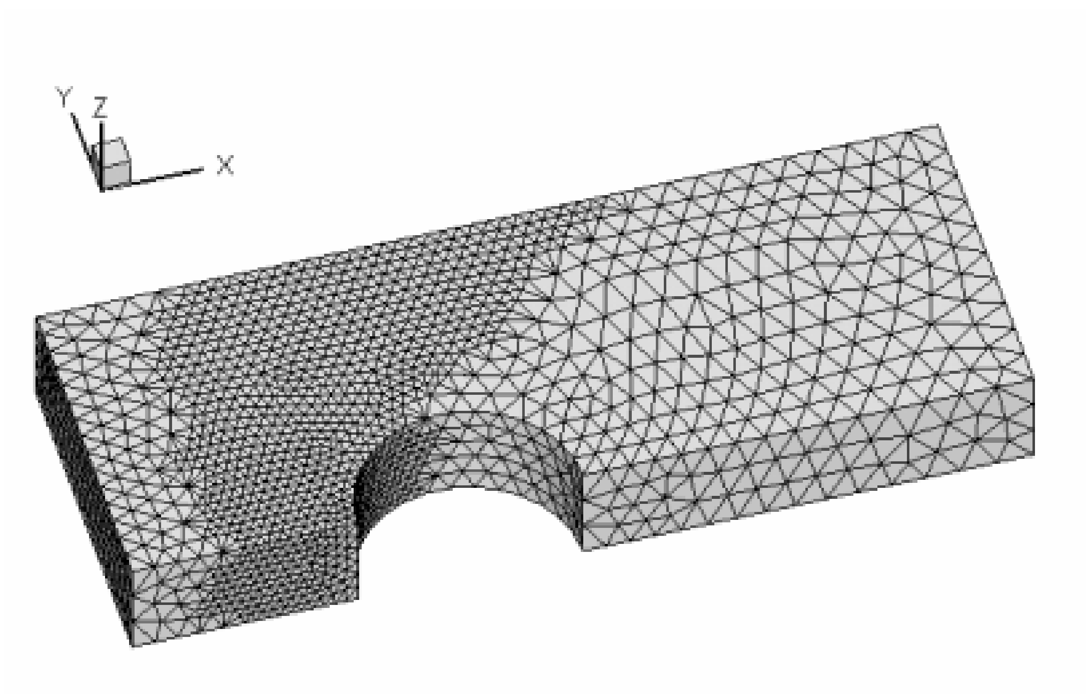
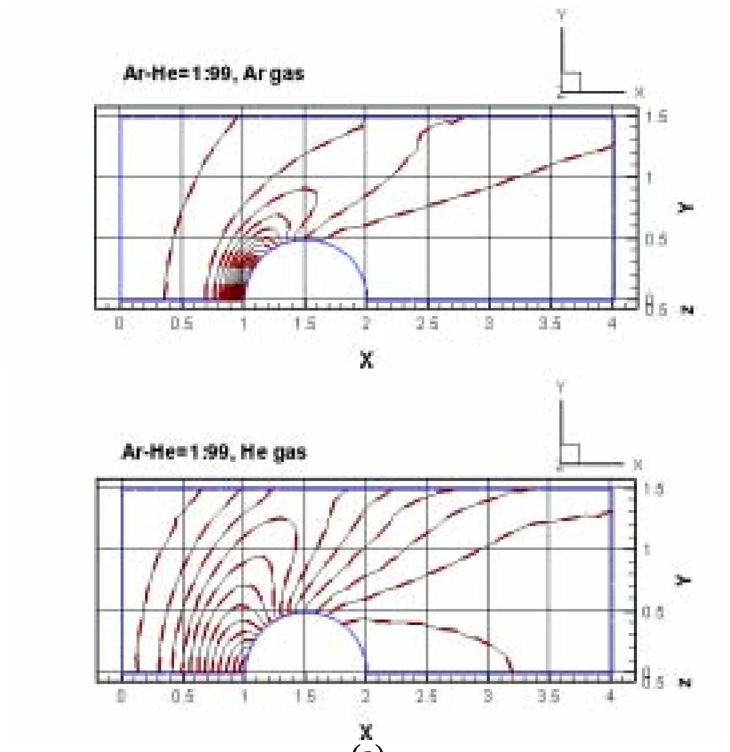
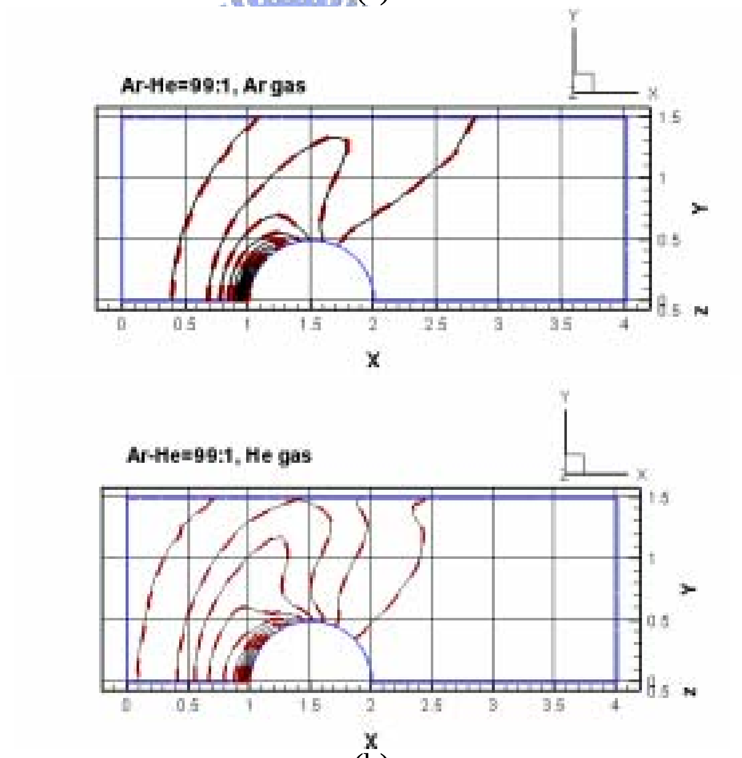


Figure 5. 6 A 3-D, unstructured tetrahedral cells for a hypersonic cylinder flow.

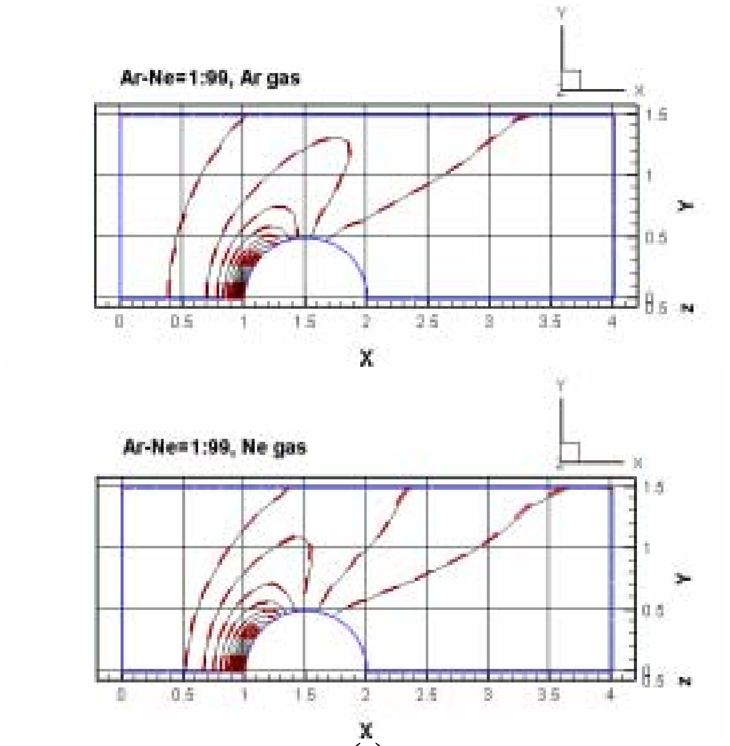


(a)

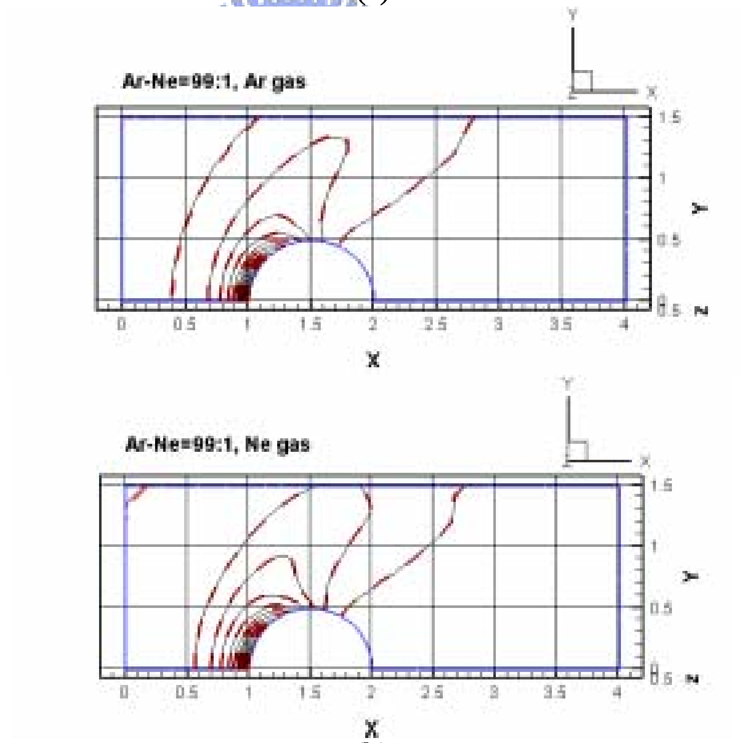


(b)

Figure 5. 7 Contours of number density of Ar-He mixture gas with mole fraction 1:99 and 99:1 (a) Ar:He=1:99; (b) Ar:He=99:1.

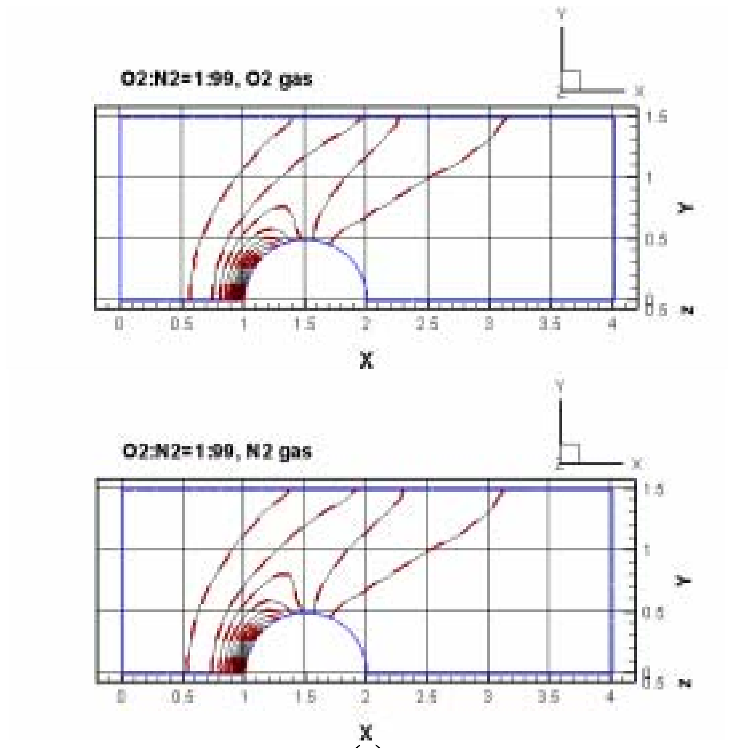


(a)

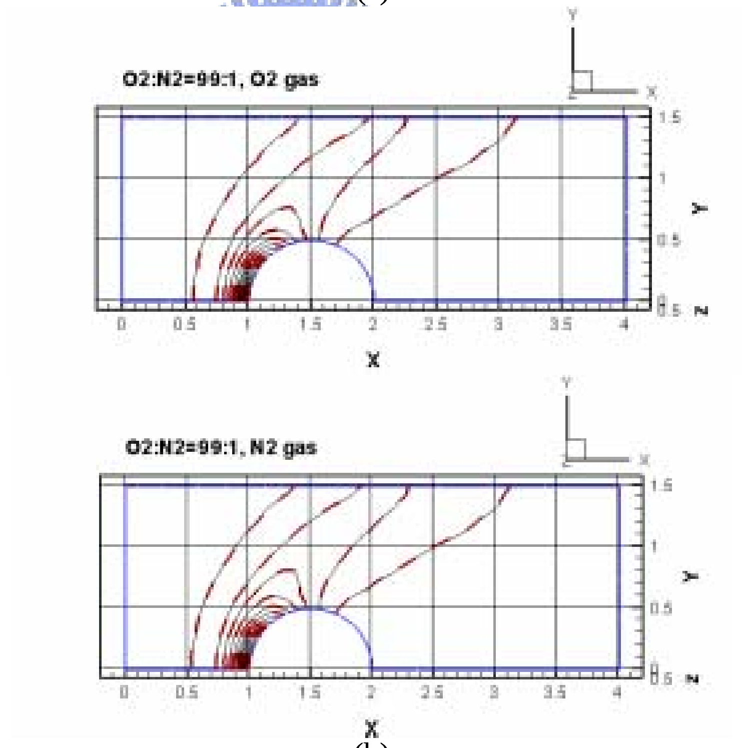


(b)

Figure 5. 8 Contours of number density of Ar-Ne mixture gas with mole fraction 1:99 and 99:1 (a) Ar:Ne=1:99; (b) Ar:Ne=99:1.



(a)



(b)

Figure 5. 9 Contours of number density of O_2-N_2 mixture gas with mole fraction 1:99 and 99:1 (a) O_2:N_2=1:99; (b) O_2:N_2=99:1.

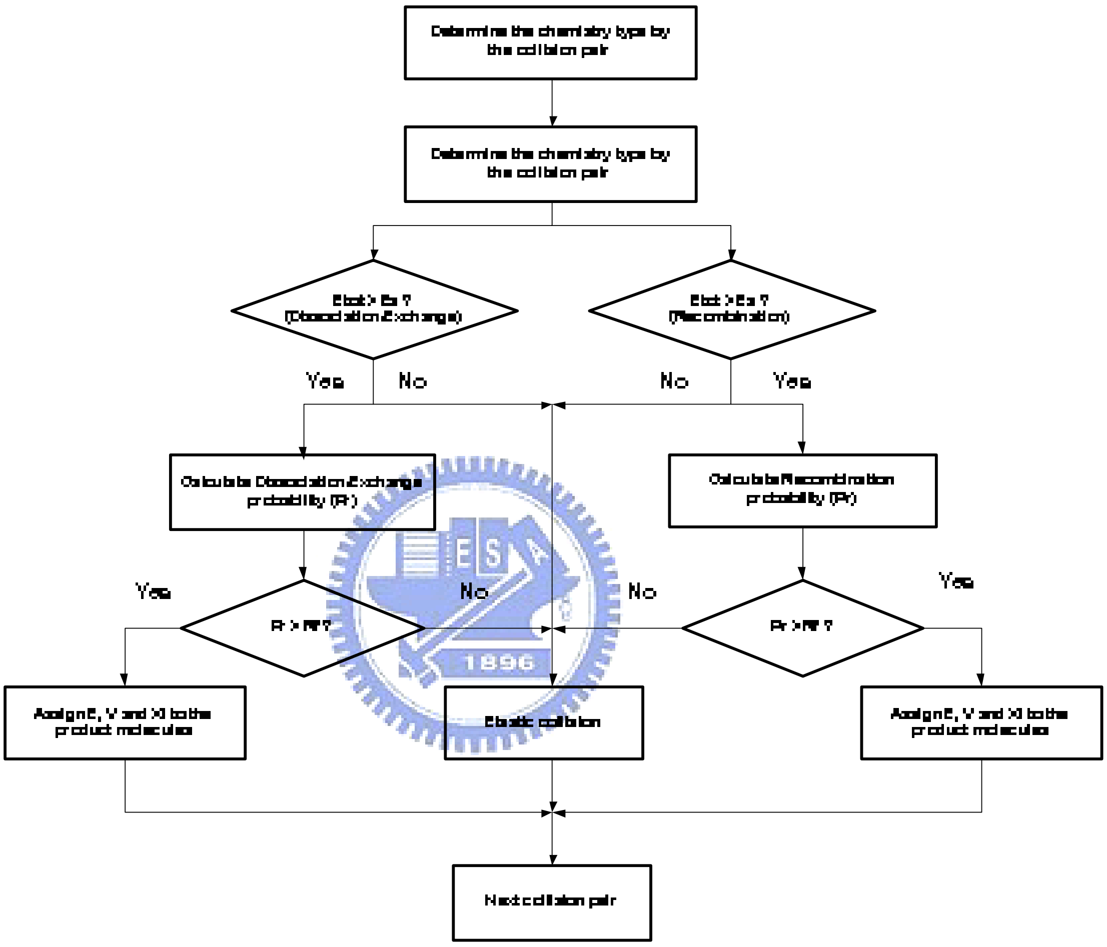
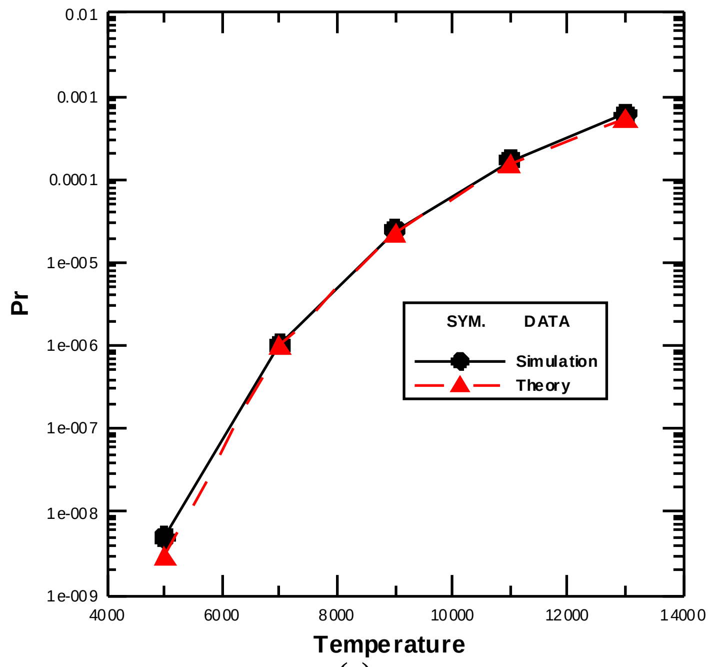
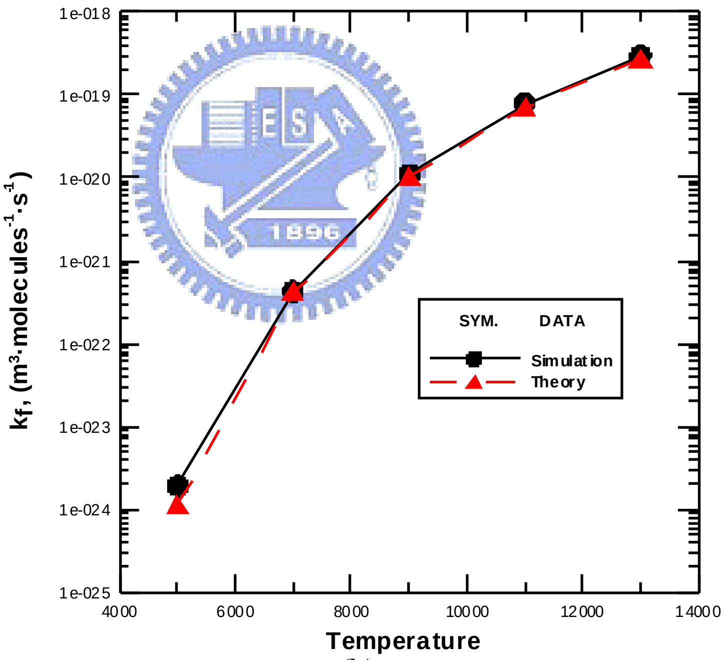

Determine the chemistry type by the collision pair
Determine the chemistry type by the collision pair
Etot > Ea ? (Dissociation/Exchange)
Etot > Ea ? (Recombination)
Yes
No
No
Yes
Calculate Dissociation/Exchange probability (Pr)
Calculate Recombination probability (Pr)
Yes
No
No
Yes
Pr > Rf ?
Pr > Rf ?
Assign E, V and XI to the product molecules
Elastic collision
Assign E, V and XI to the product molecules
Next collision pair

Figure 6. 1 The flow chart of chemical reaction in the PDSC.

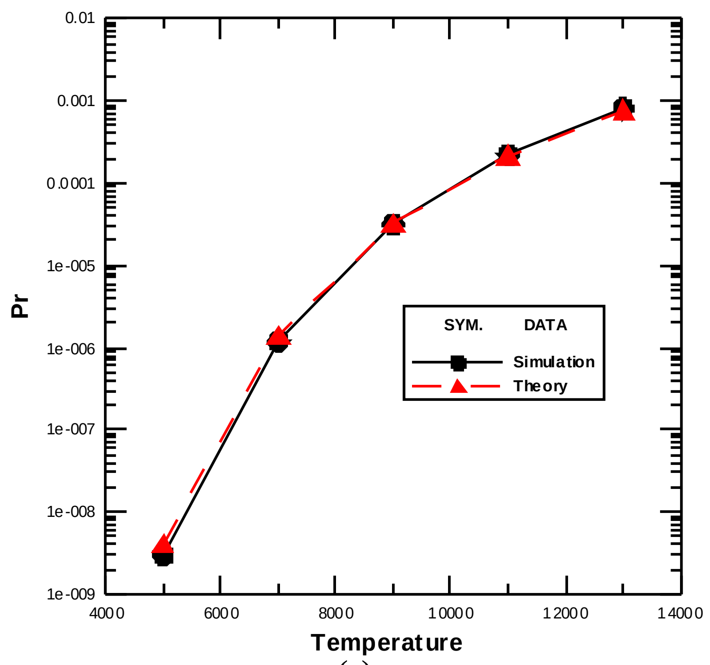


(a)

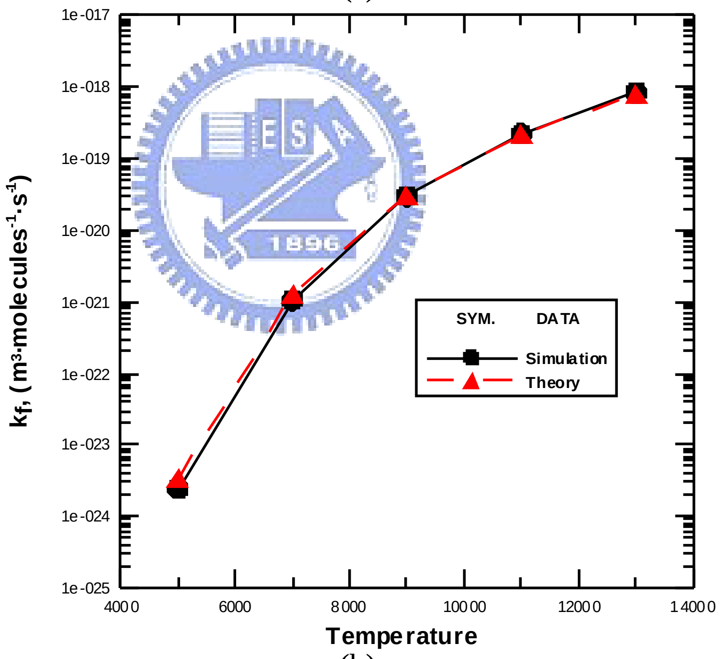


(b)

Figure 6. 2 Comparison of simulation and theoretical data of $N_2 + N_2 \rightarrow N + N + N_2$ reaction (a) reaction probability; (b) rate constant.

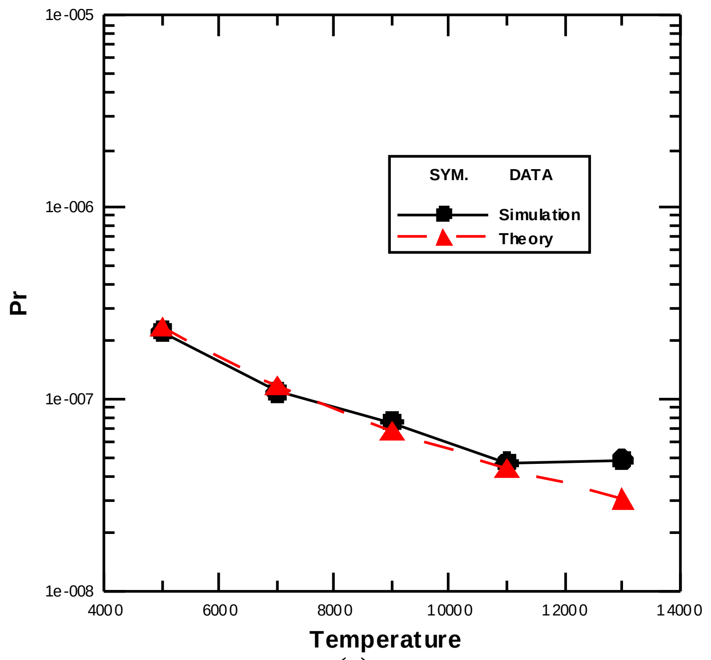


(a)

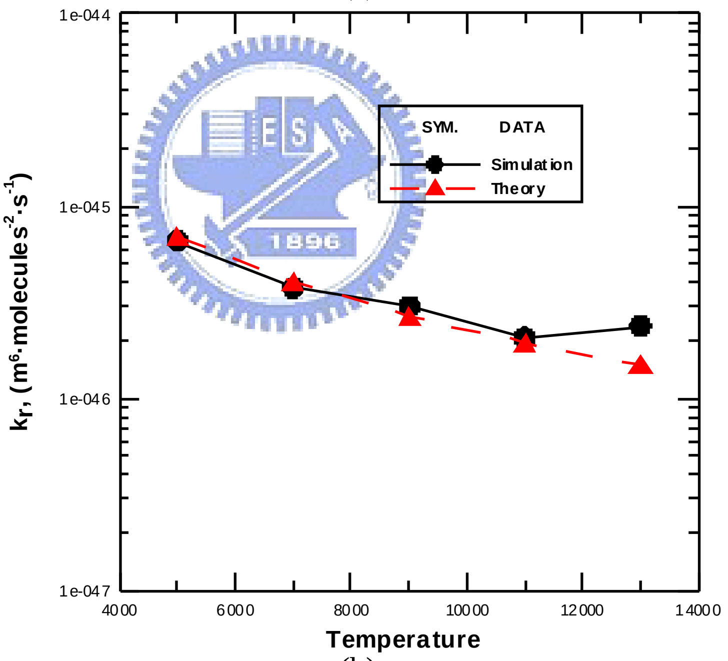


(b)

Figure 6. 3 Comparison of simulation and theoretical data of $N_2 + N \rightarrow N + N + N$ reaction (a) reaction probability; (b) rate constant.

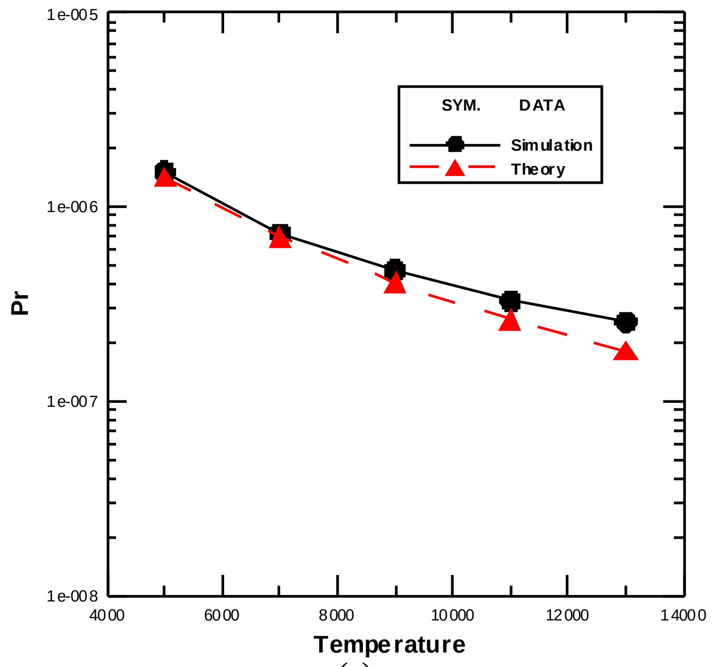


(a)

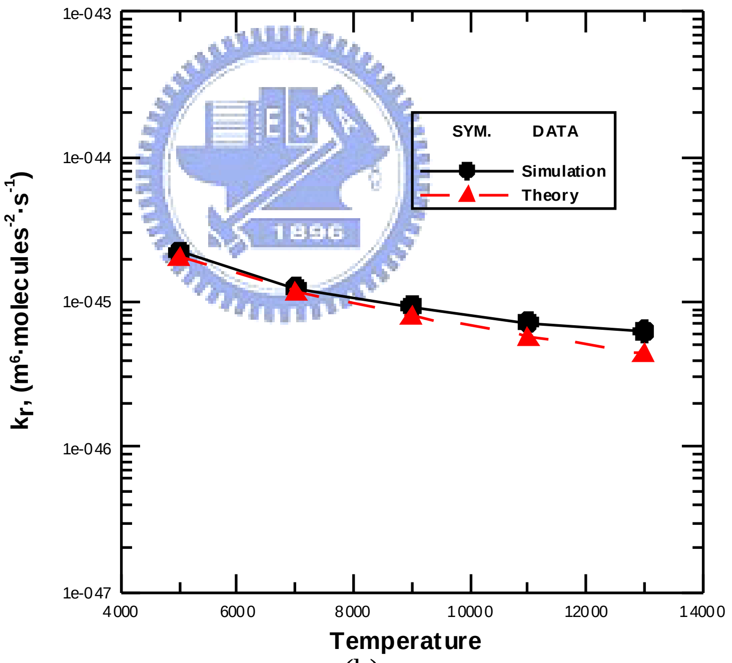


(b)

Figure 6. 4 Comparison of simulation and theoretical data of $N + N + N_2 \rightarrow N_2 + N_2$ reaction (a) reaction probability; (b) rate constant.

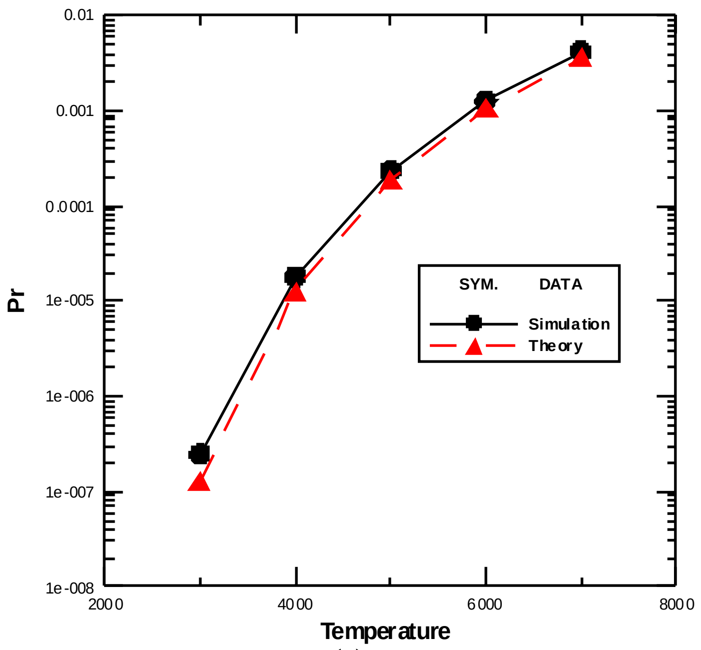


(a)

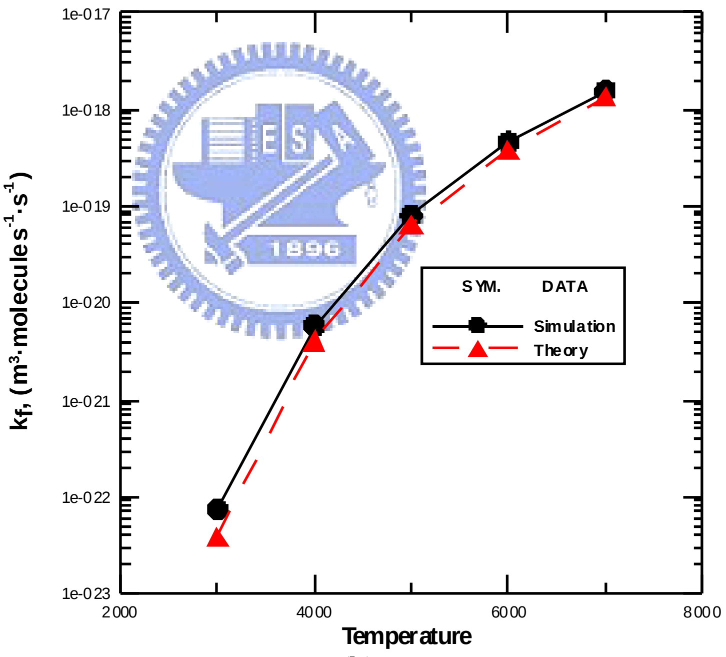


(b)

Figure 6. 5 Comparison of simulation and theoretical data of $N + N + N \rightarrow N_2 + N$ reaction (a) reaction probability; (b) rate constant.

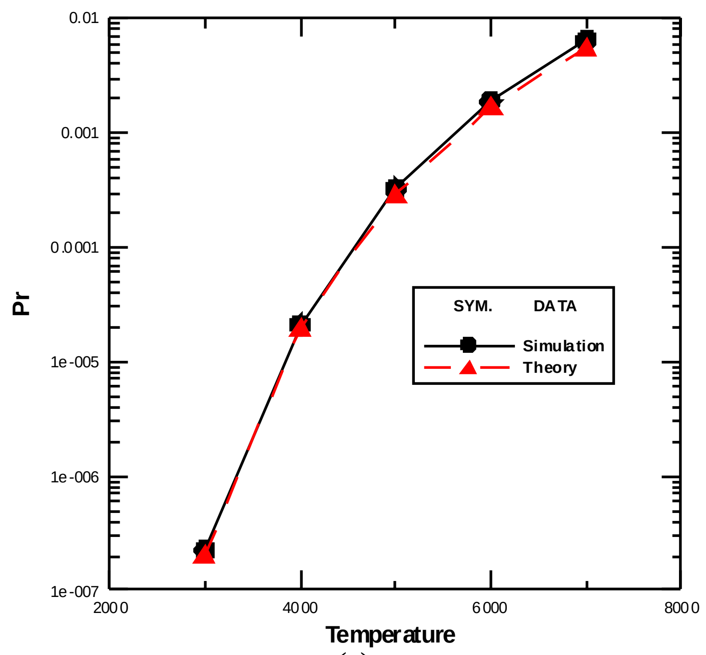


(a)

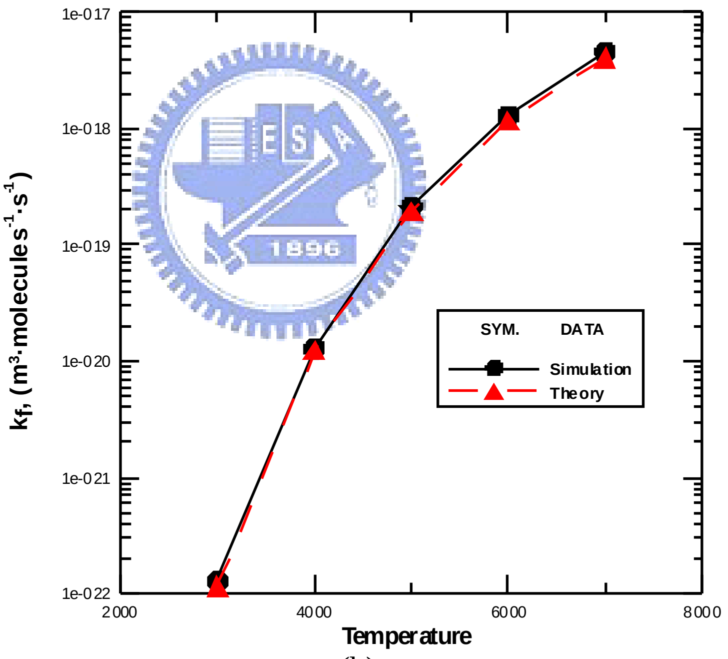


(b)

Figure 6. 6 Comparison of simulation and theoretical data of $N_2 + N_2 \rightarrow N + N + N_2$ reaction (a) reaction probability; (b) rate constant.

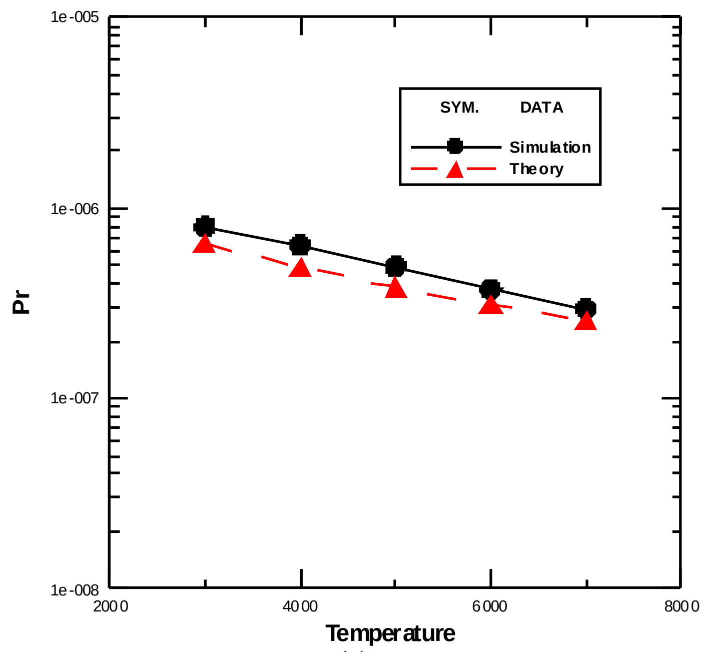


(a)

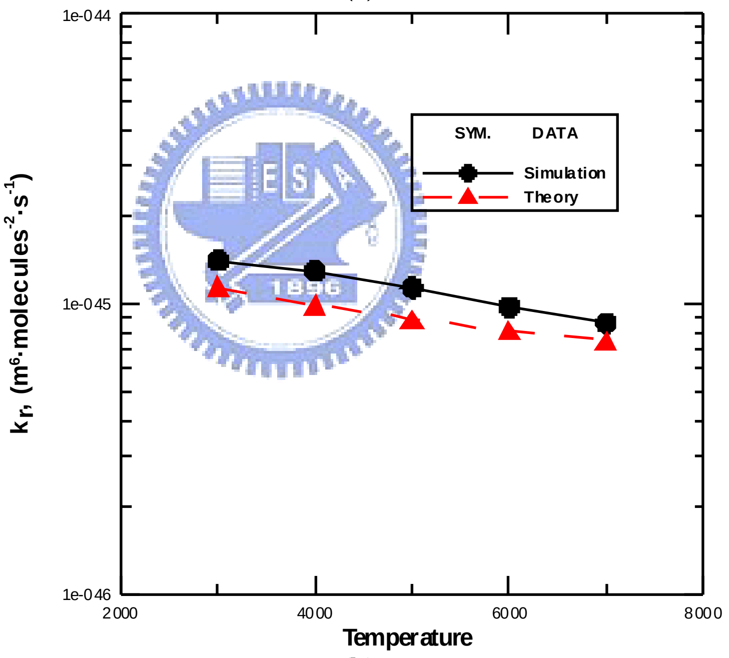


(b)

Figure 6. 7 Comparison of simulation and theoretical data of $O_2 + O \rightarrow O + O + O$ reaction (a) reaction probability; (b) rate constant.

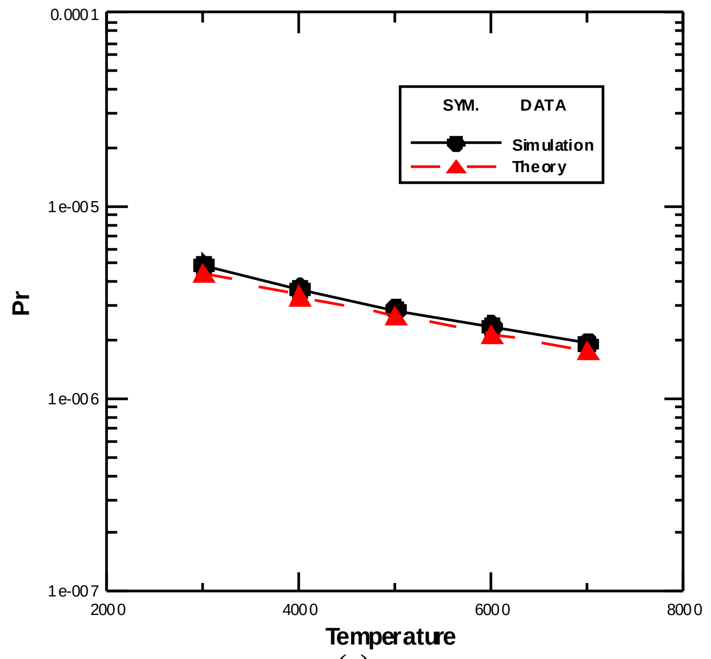


(a)

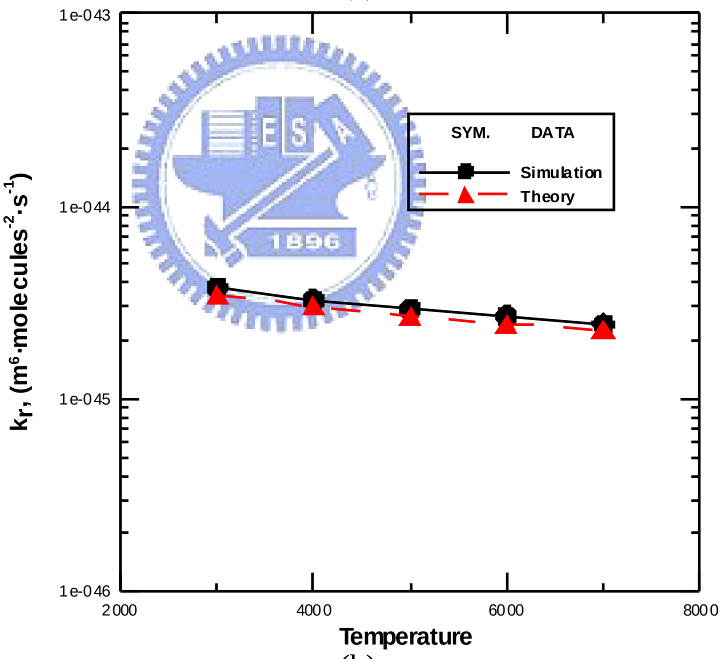


(b)

Figure 6. 8 Comparison of simulation and theoretical data of $O + O + O_2 \rightarrow O_2 + O_2$ reaction (a) reaction probability; (b) rate constant.

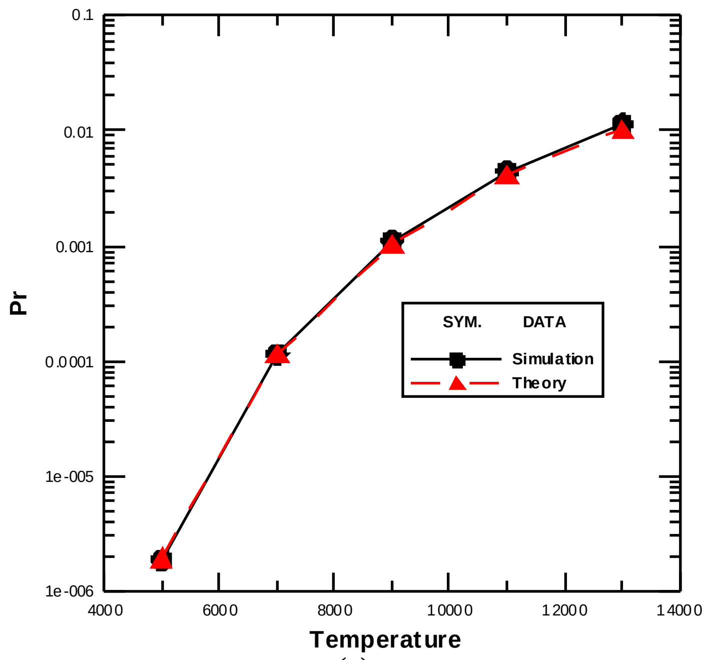


(a)

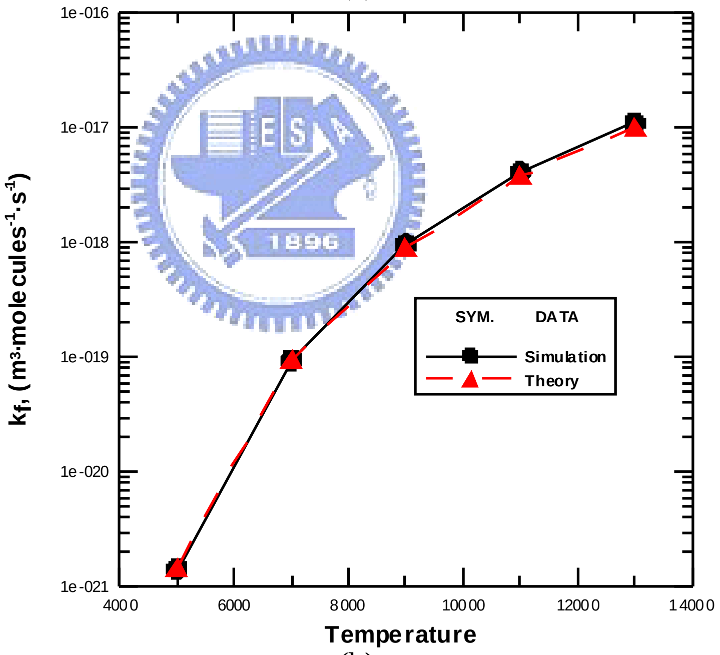


(b)

Figure 6. 9 Comparison of simulation and theoretical data of $O + O + O \rightarrow O_2 + O$ reaction (a) reaction probability; (b) rate constant.

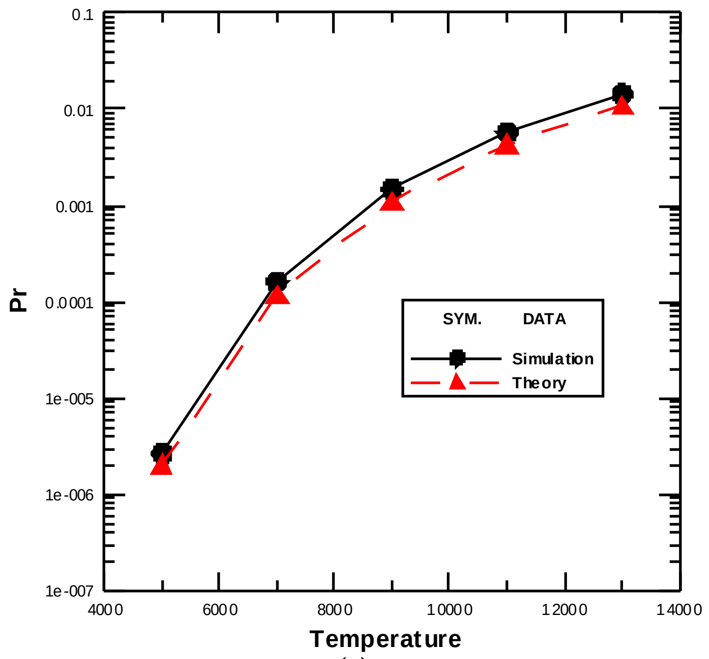


(a)

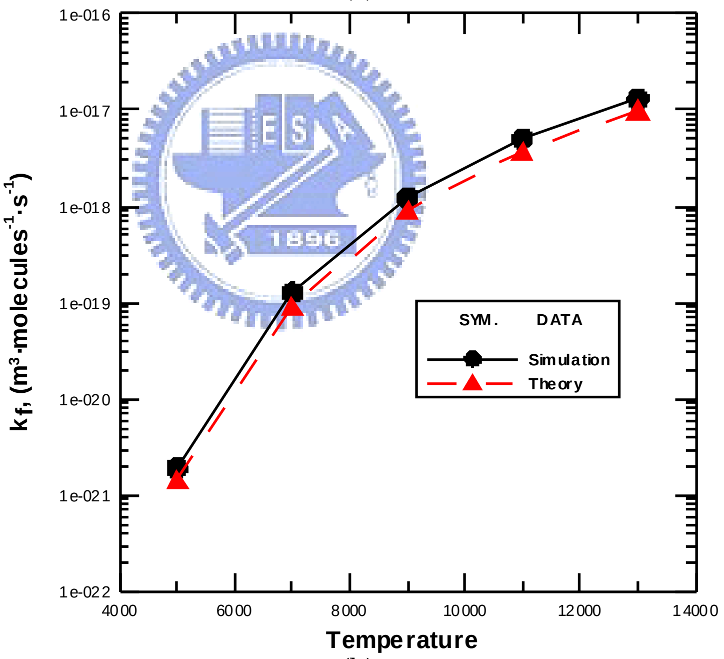


(b)

Figure 6. 10 Comparison of simulation and theoretical data of $NO + N_2 \rightarrow N + O + N_2$ reaction (a) reaction probability; (b) rate constant.

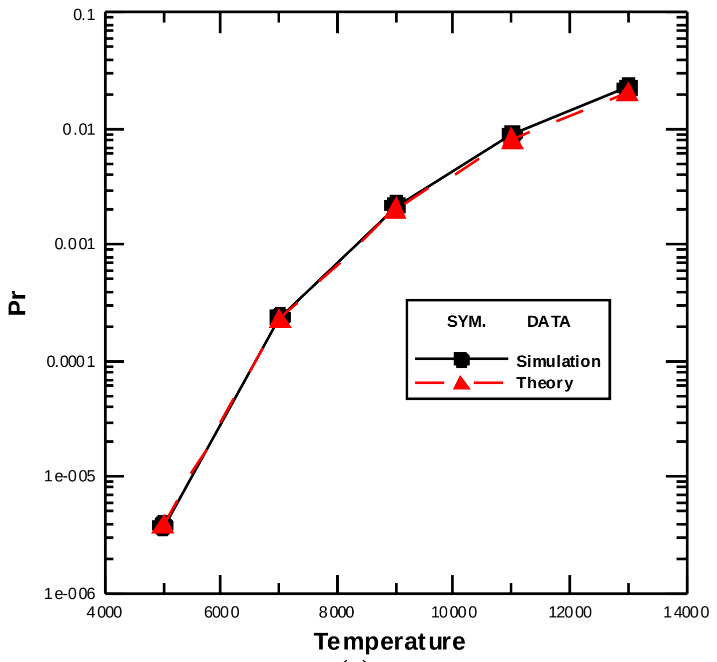


(a)

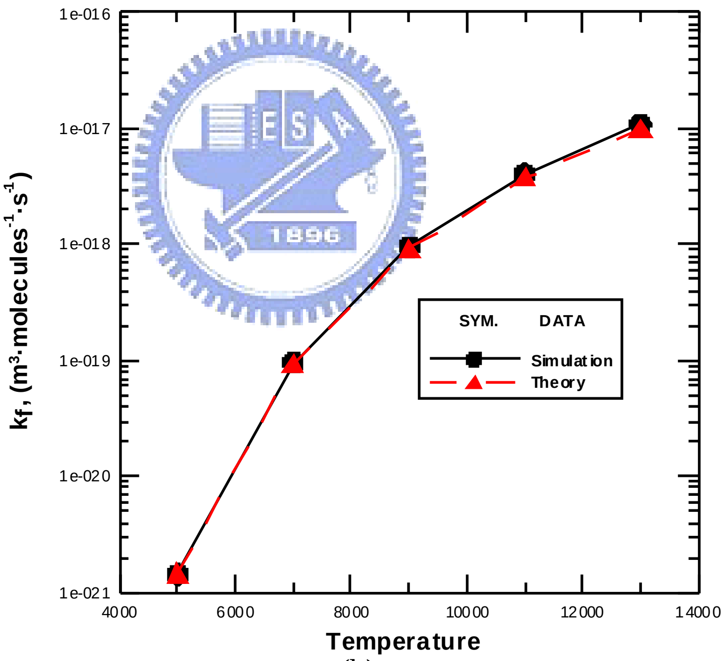


(b)

Figure 6. 11 Comparison of simulation and theoretical data of $NO + O_2 \rightarrow N + O + O_2$ reaction (a) reaction probability; (b) rate constant.

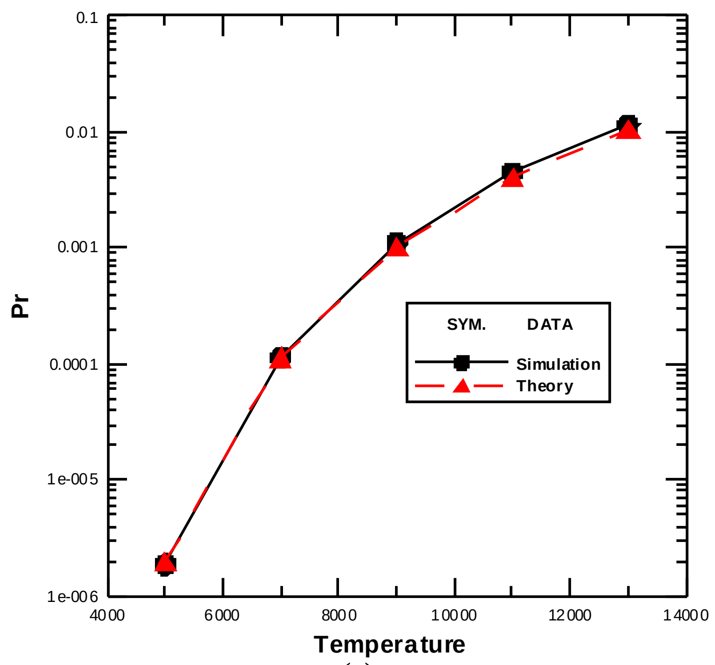


(a)

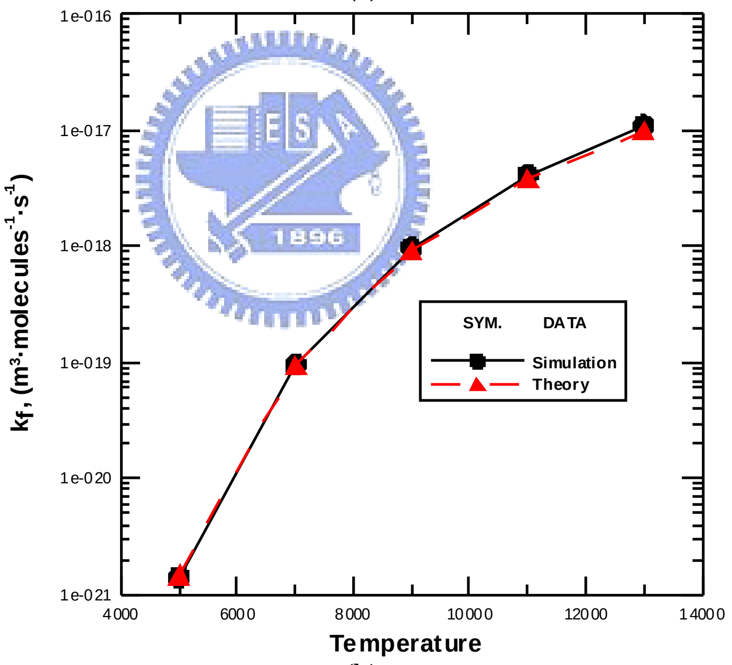


(b)

Figure 6. 12 Comparison of simulation and theoretical data of $NO + NO \rightarrow N + O + NO$ reaction (a) reaction probability; (b) rate constant.

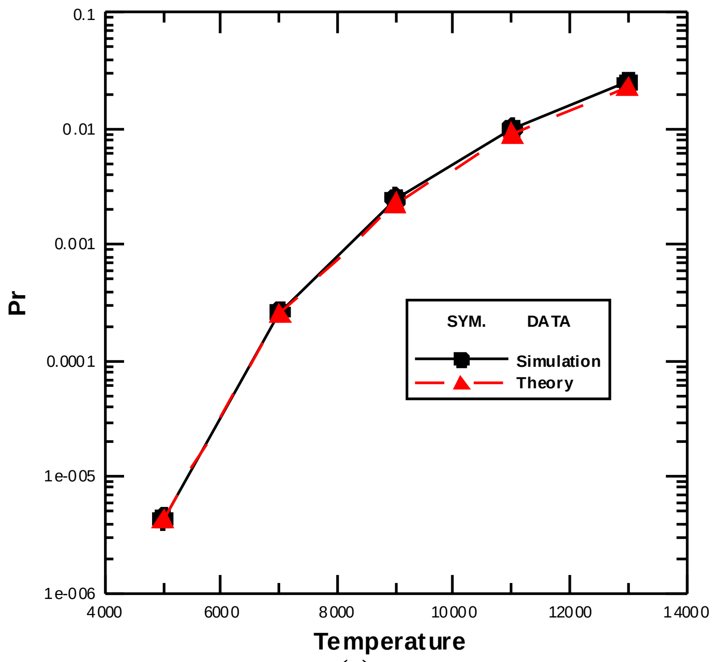


(a)

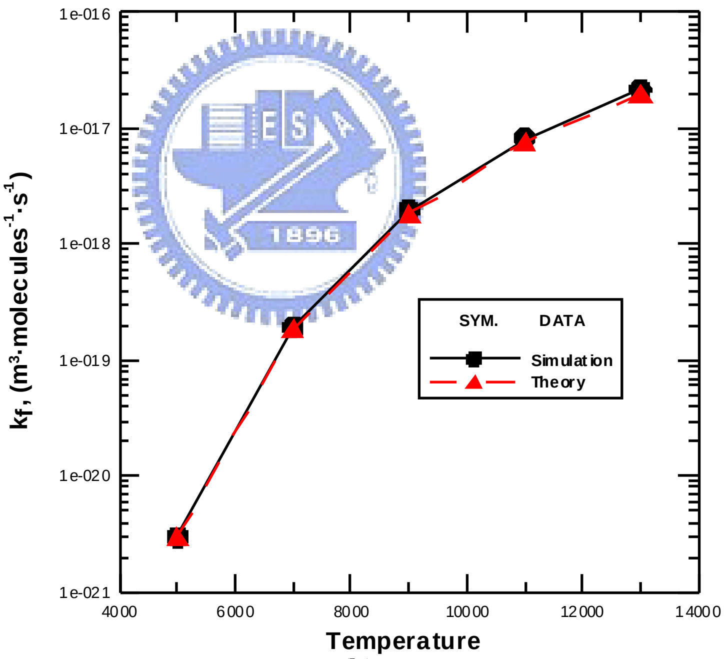


(b)

Figure 6. 13 Comparison of simulation and theoretical data of $NO + N \rightarrow N + O + N$ reaction (a) reaction probability; (b) rate constant.

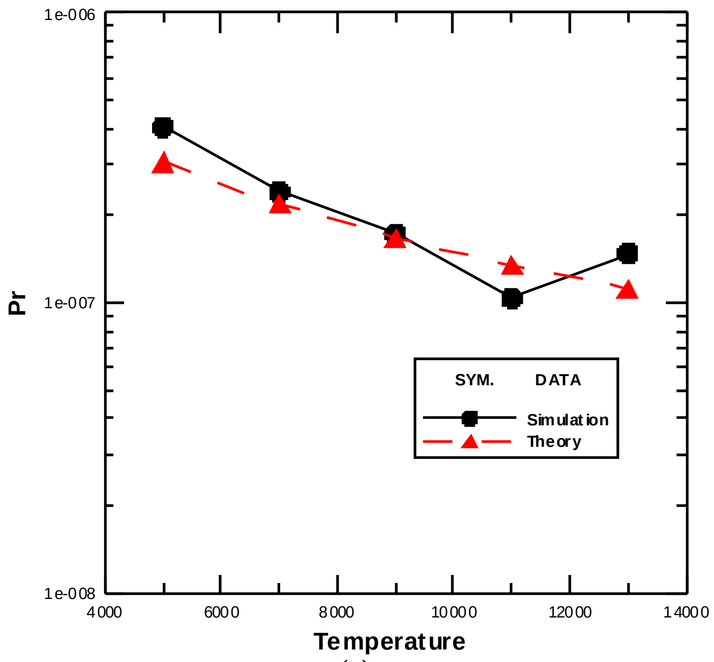


(a)

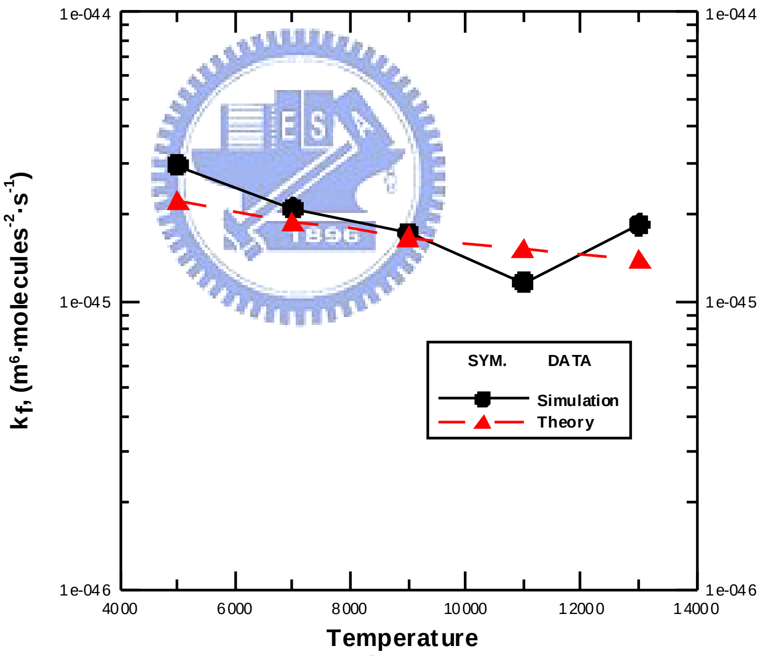


(b)

Figure 6. 14 Comparison of simulation and theoretical data of $NO + O \rightarrow N + O + O$ reaction (a) reaction probability; (b) rate constant.

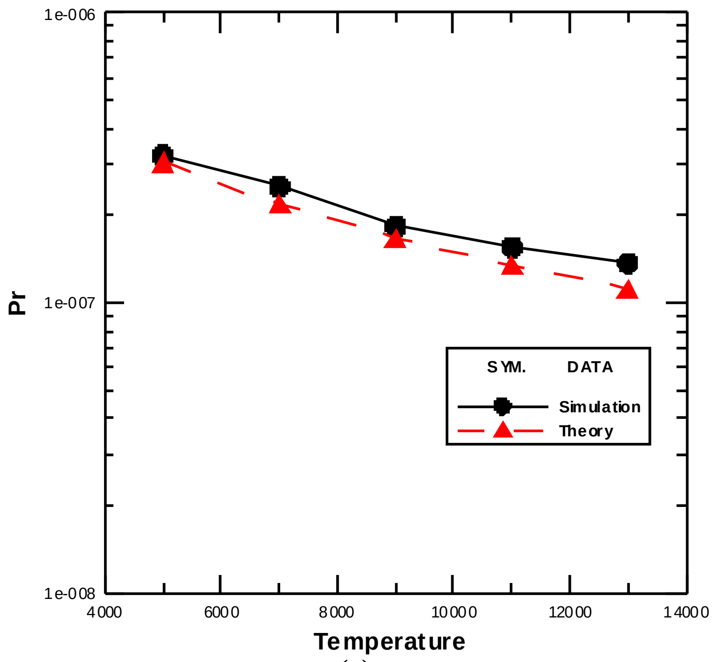


(a)

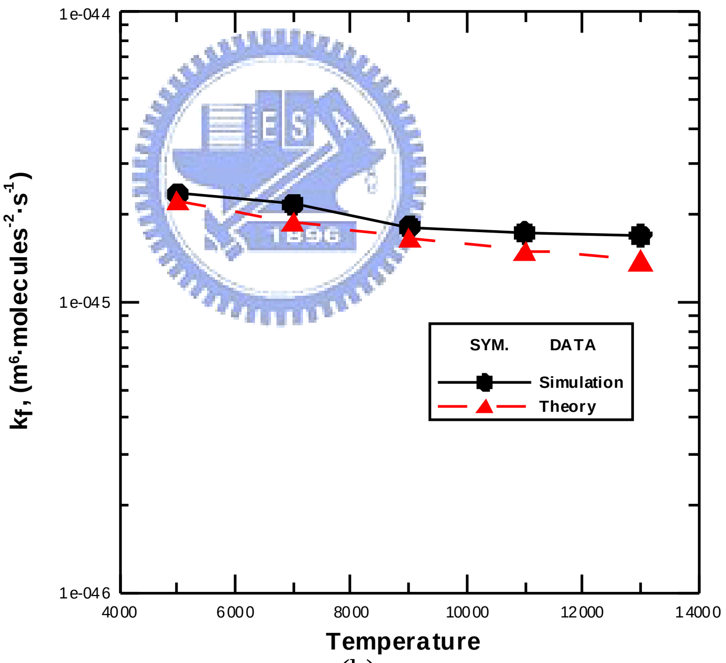


(b)

Figure 6. 15 Comparison of simulation and theoretical data of $N + O + N_2 \rightarrow NO + N_2$ reaction (a) reaction probability; (b) rate constant.

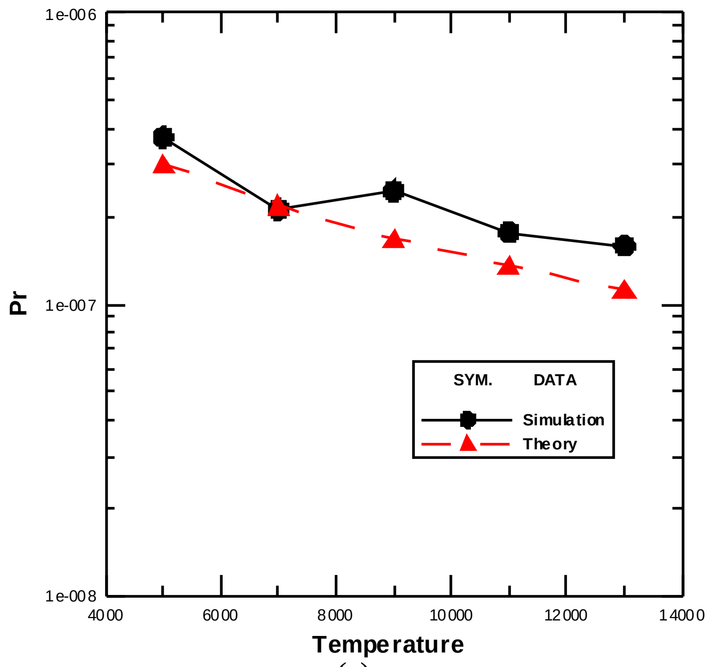


(a)

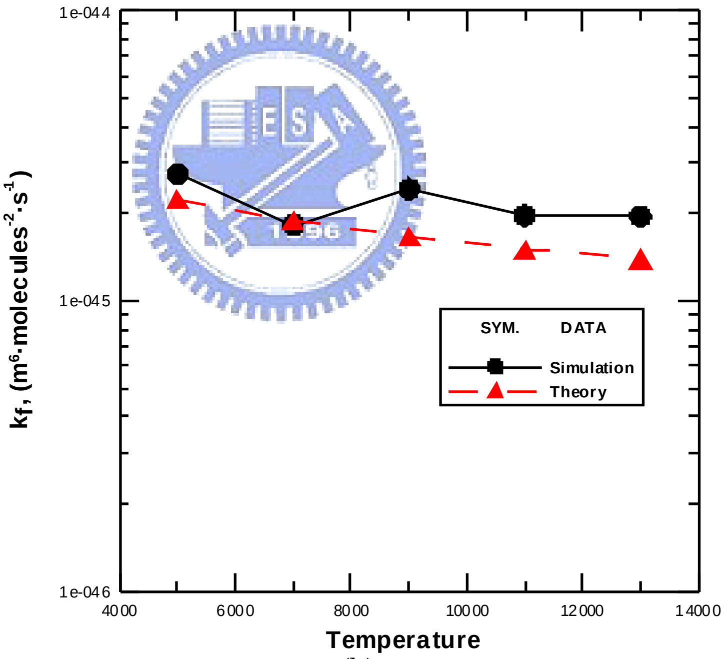


(b)

Figure 6. 16 Comparison of simulation and theoretical data of $N + O + O_2 \rightarrow NO + O_2$ reaction (a) reaction probability; (b) rate constant.

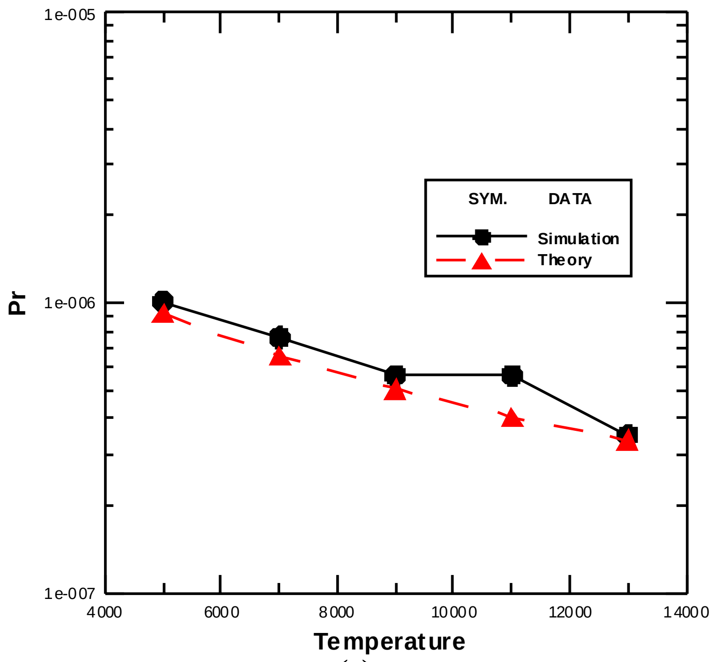


(a)

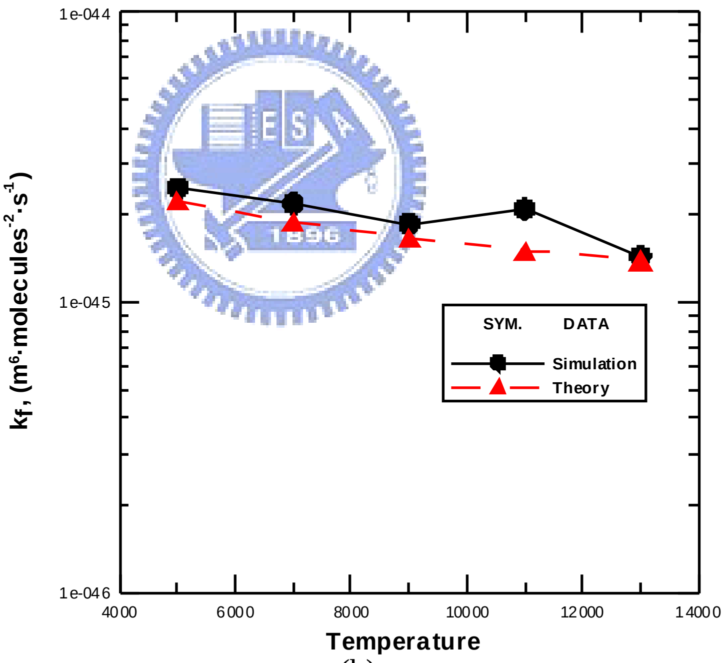


(b)

Figure 6. 17 Comparison of simulation and theoretical data of $N + O + NO \rightarrow NO + NO$ reaction (a) reaction probability; (b) rate constant.

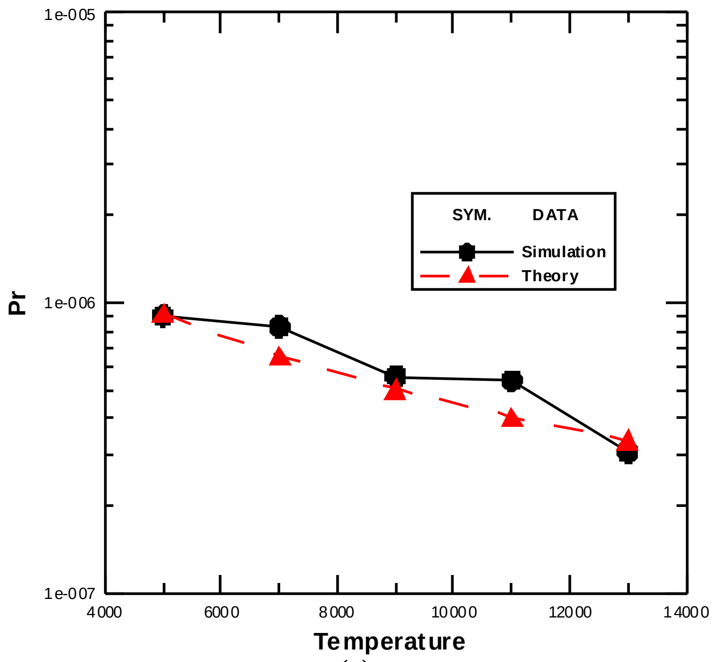


(a)

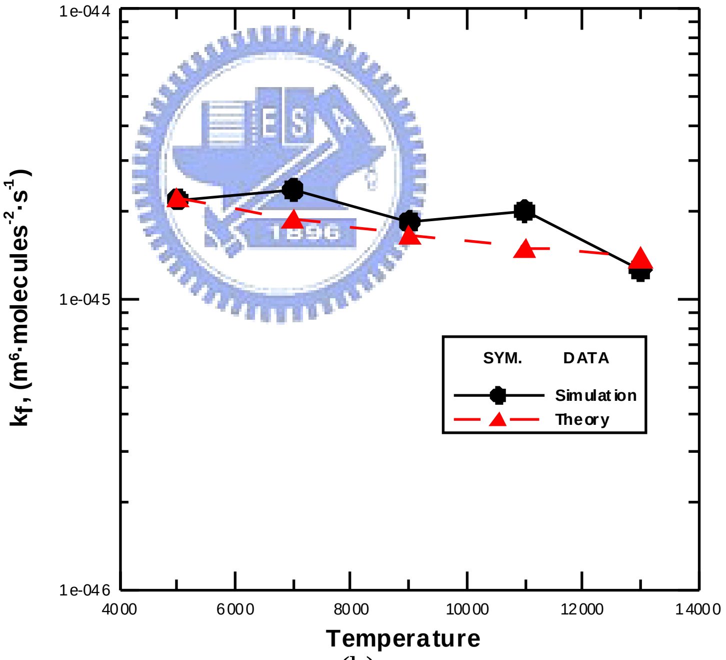


(b)

Figure 6. 18 Comparison of simulation and theoretical data of $N + O + N \rightarrow NO + N$ reaction (a) reaction probability; (b) rate constant.

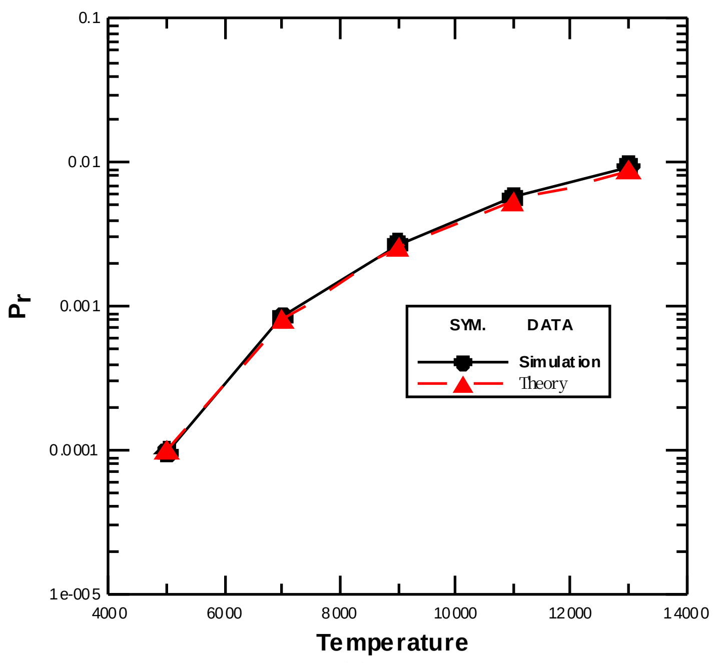


(a)

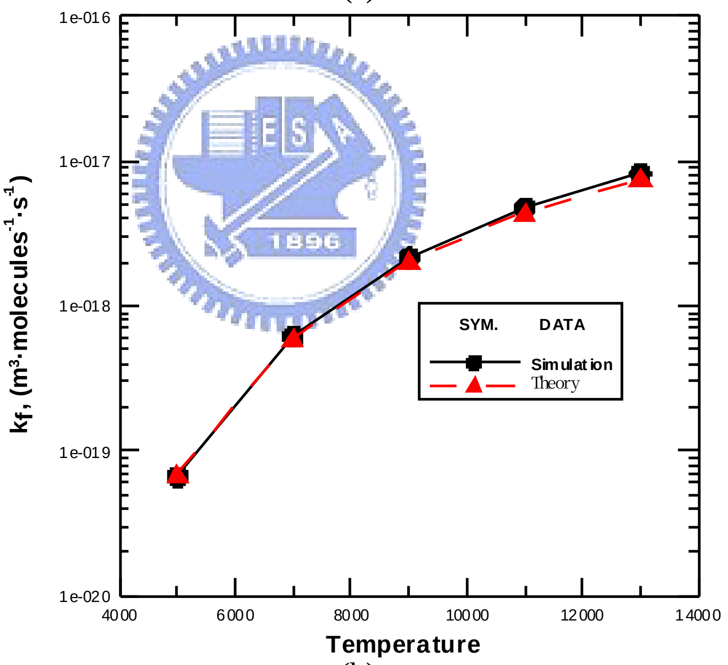


(b)

Figure 6. 19 Comparison of simulation and theoretical data of $N + O + O \rightarrow NO + O$ reaction (a) reaction probability; (b) rate constant.

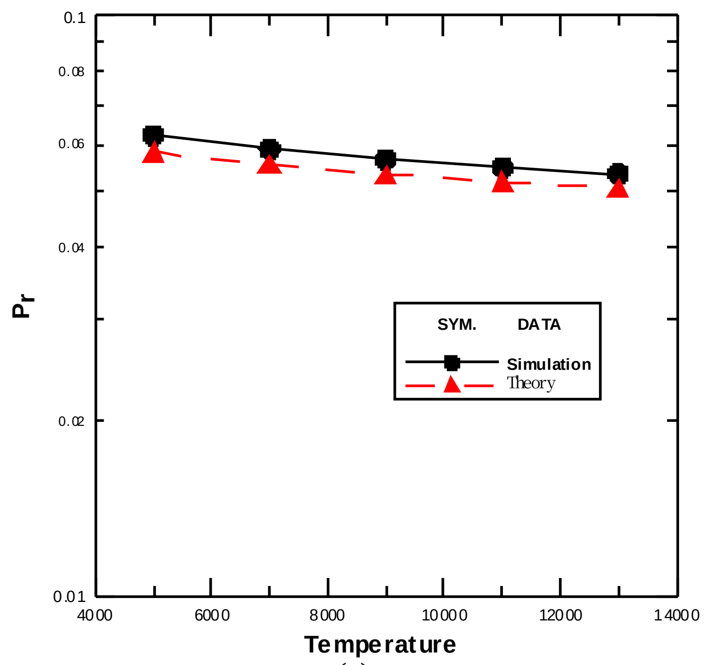


(a)

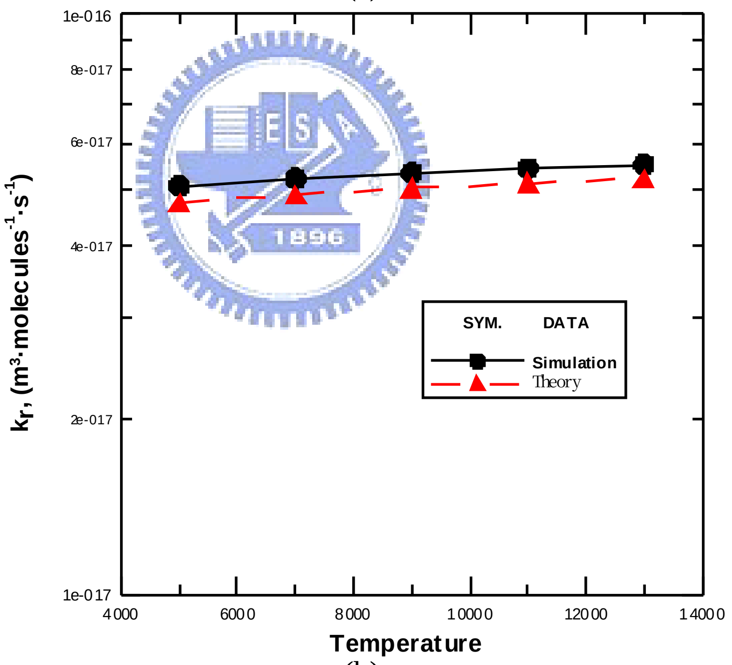


(b)

Figure 6. 20 Comparison of simulation and theoretical data of $N_2 + O \rightarrow NO + N$ reaction (a) reaction probability; (b) rate constant.

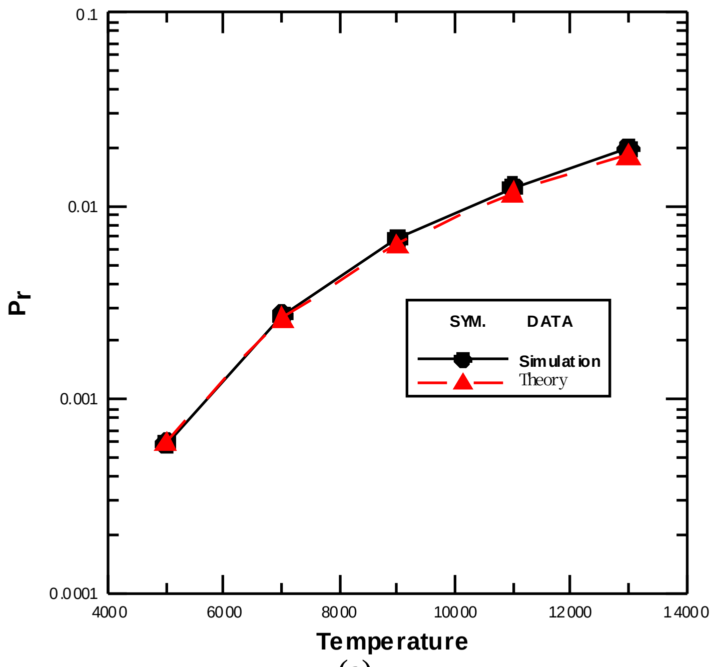


(a)

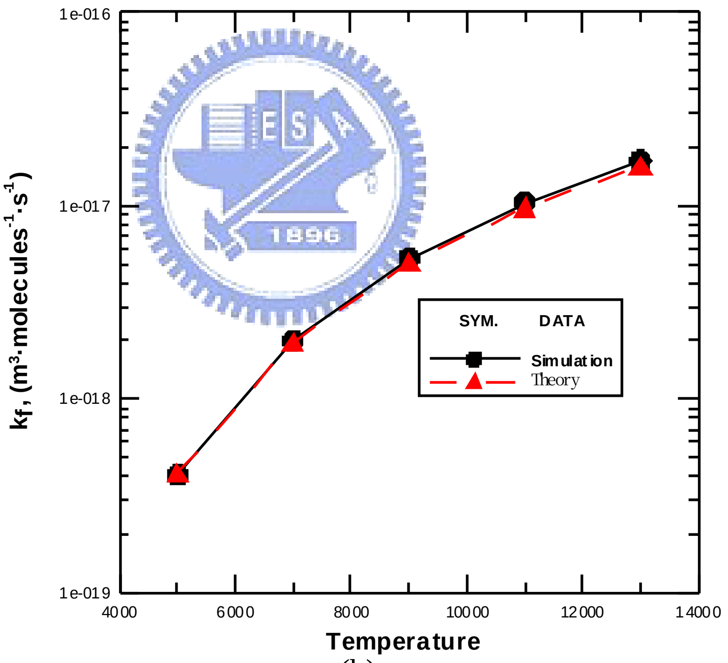


(b)

Figure 6. 21 Comparison of simulation and theoretical data of $NO + N \rightarrow N_2 + O$ reaction (a) reaction probability; (b) rate constant



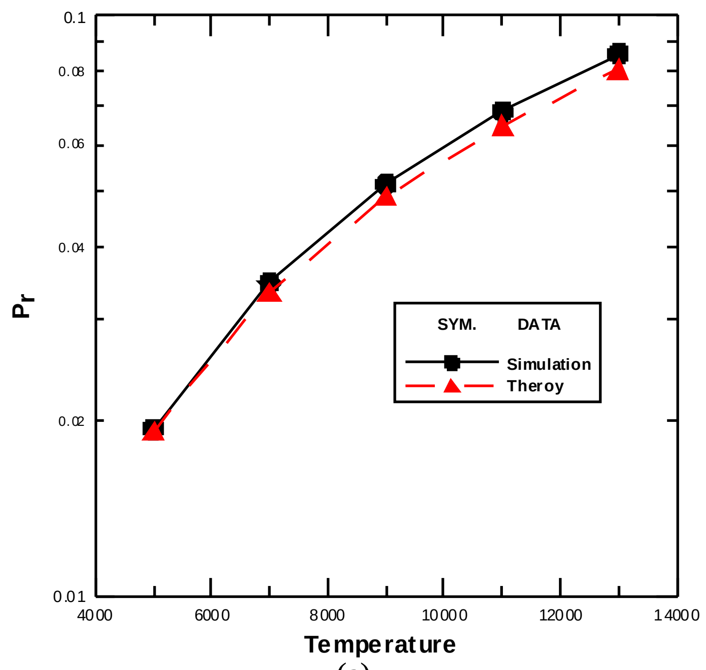
(a)



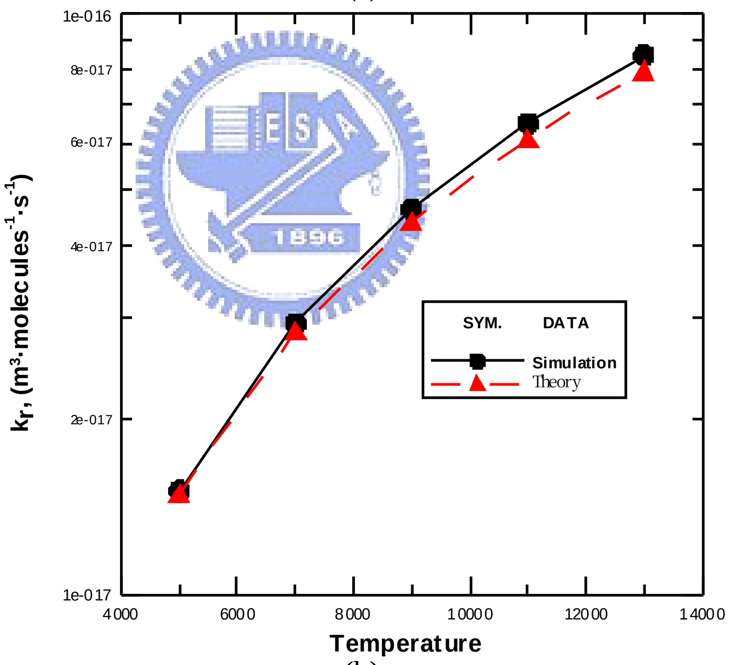
(b)

Figure 6. 22 Comparison of simulation and theoretical data of $NO + O \rightarrow O_2 + N$ reaction (a) reaction probability; (b) rate constant

ES 1896

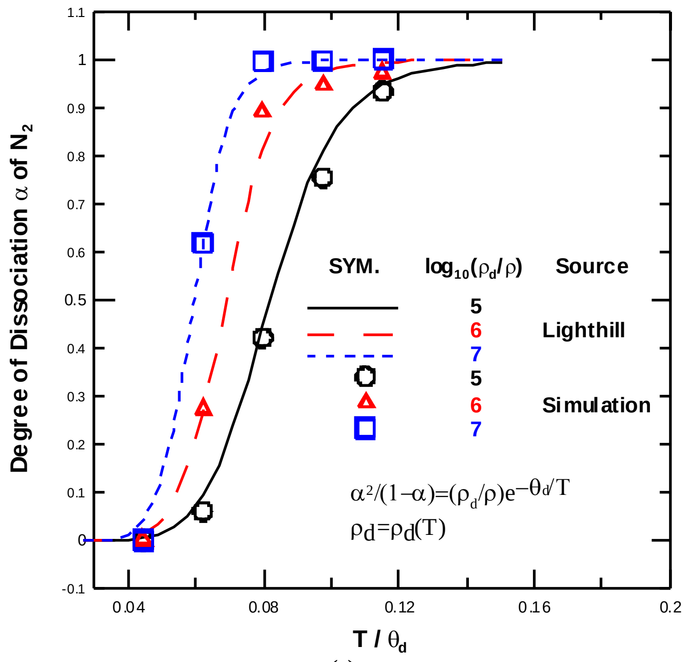


(a)

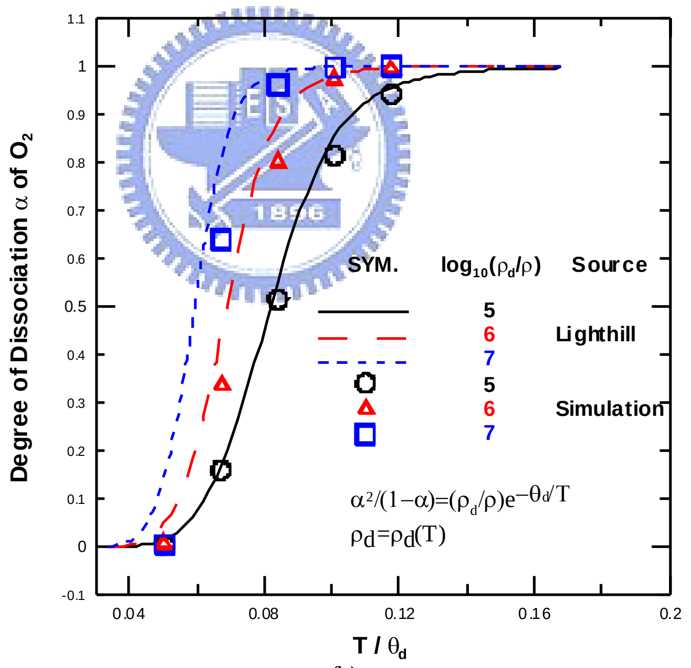


(b)

Figure 6. 23 Comparison of simulation and theoretical data of $O_2 + N \rightarrow NO + O$ reaction (a) reaction probability; (b) rate constant.

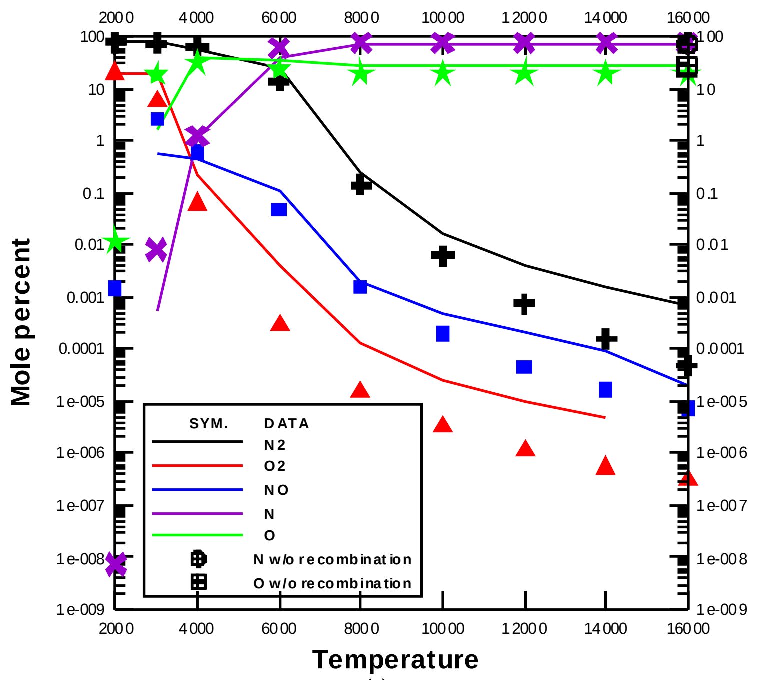


(a)

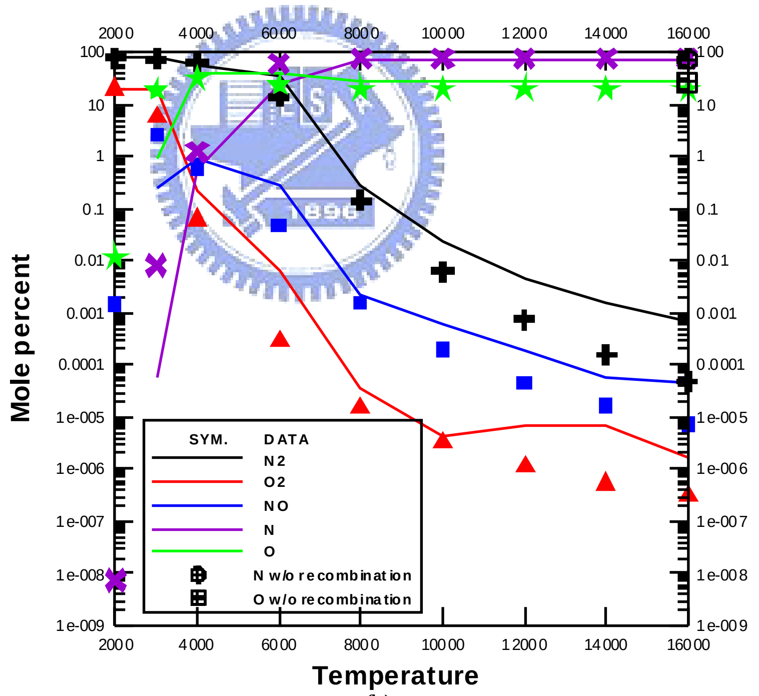


(b)

Figure 6. 24 Degree of dissociation for idea dissociating gas (a) nitrogen; (b) oxygen.

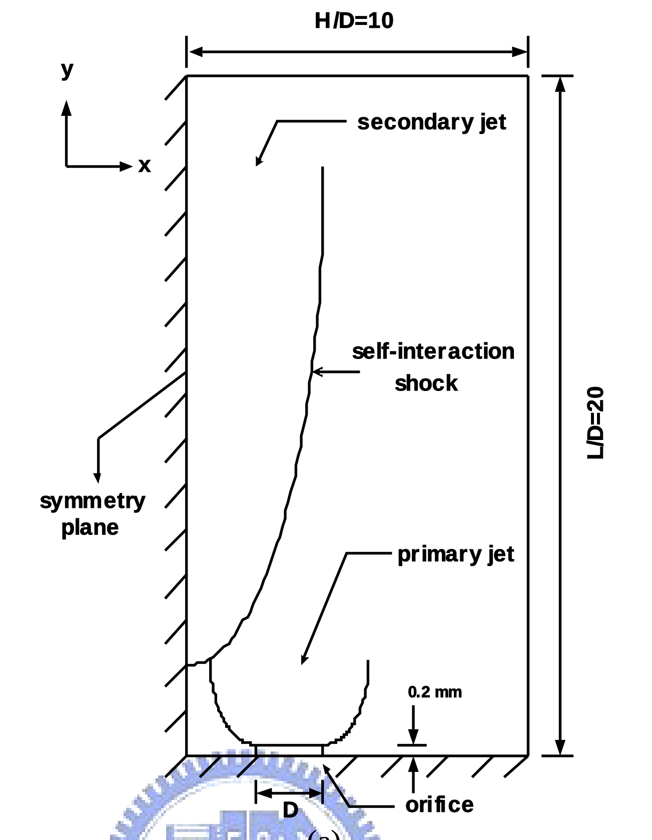


(a)

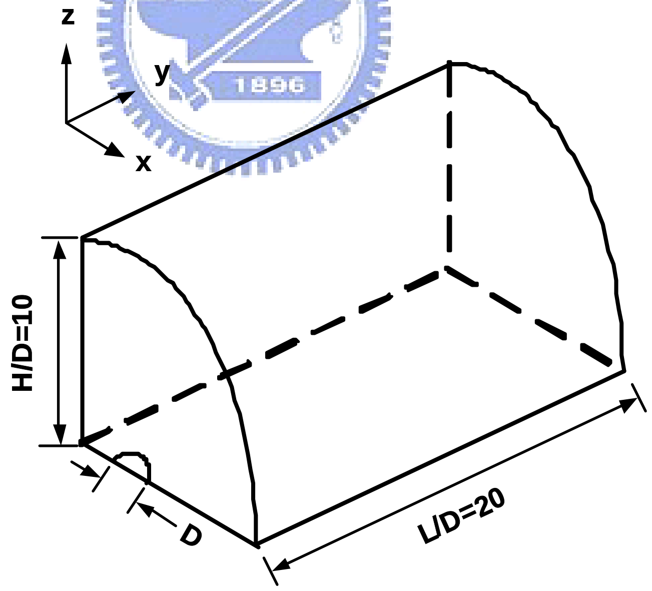


(b)

Figure 6. 25 Equilibrium composition of air at density of 10^{-2} atm (a) original; (b) new fitting chem.dat. (lines= simulation; symbols=rate equation analysis)

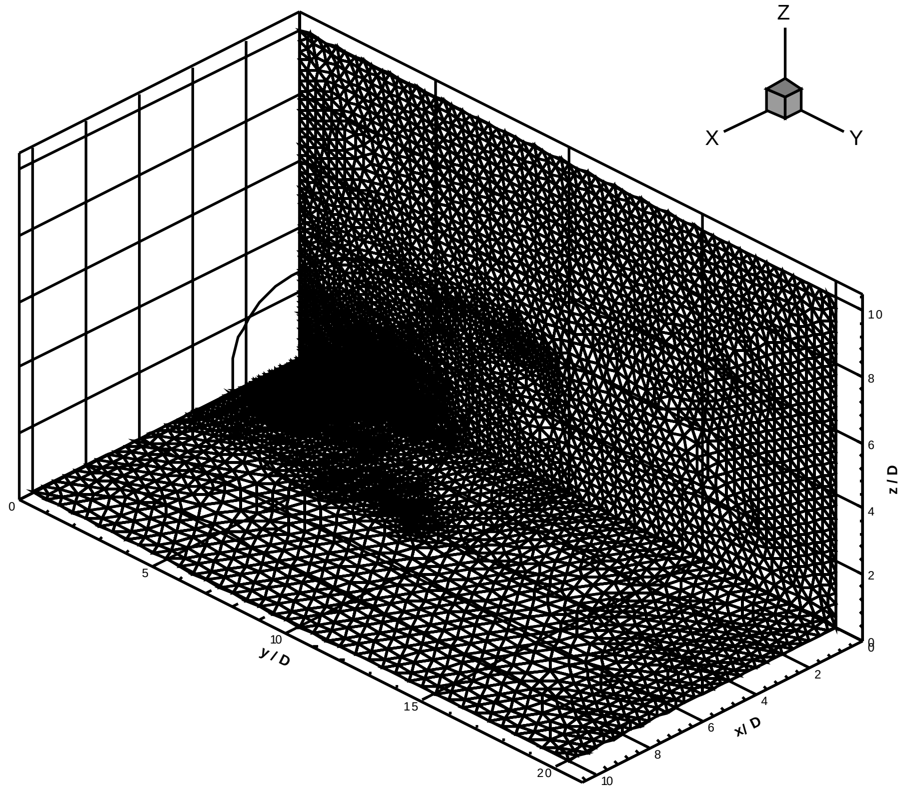


(a)

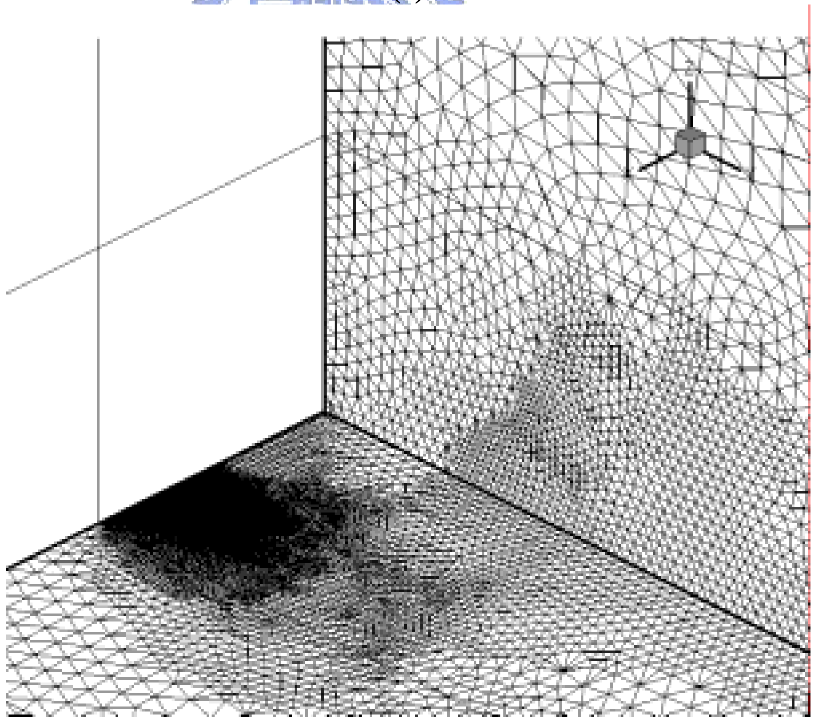


(b)

Figure 7. 1 Sketch of a flow of twin-jet interaction (N_2 gas, *Po*=870 Pa, T_o=285 K, *L*=*W*=10*D*, *L*=20*D*, *D*=3 mm, Kn_{th}=0.00385).



(a)



(b)

Figure 7. 2 The mesh of level-2 adaptation along x-y and y-z planes for the twin-jet interaction (657,624).

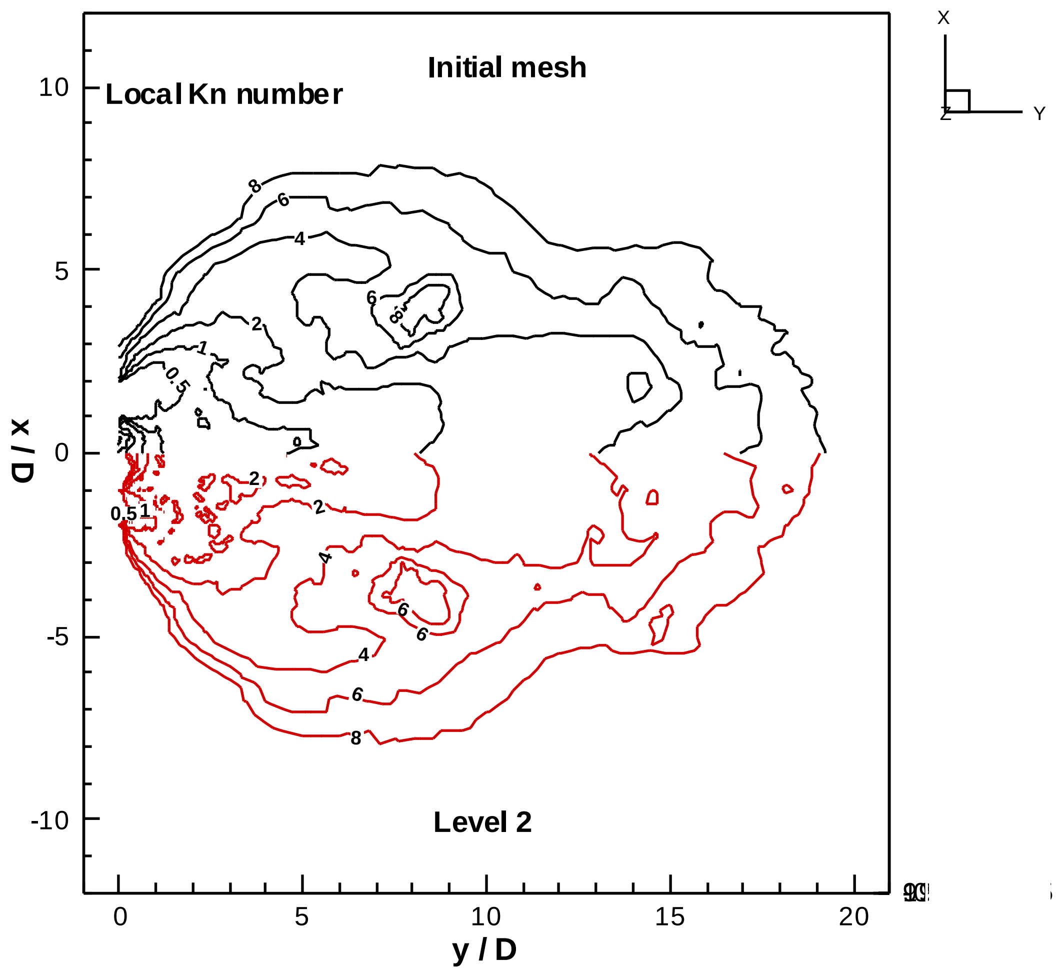
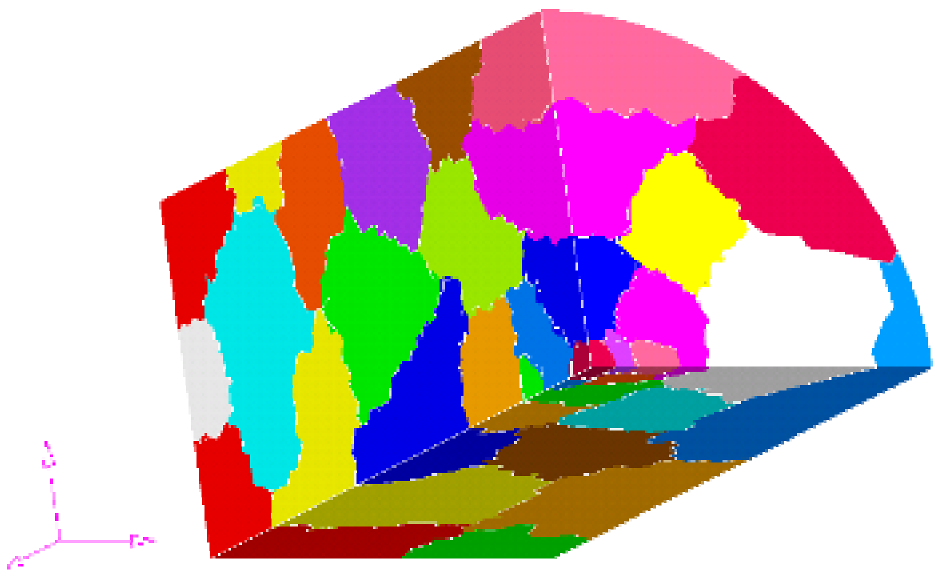
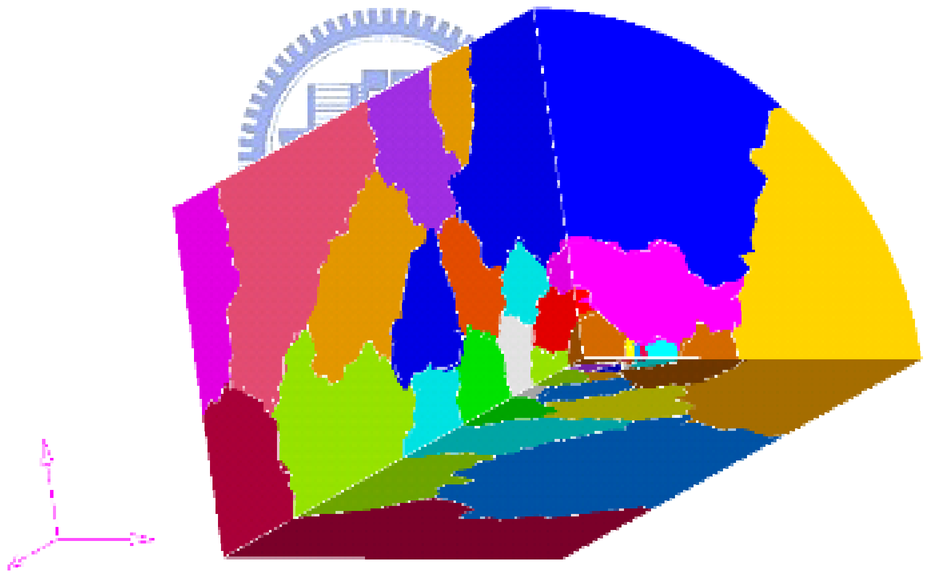


Figure 7. 3 Surface local cell Knudsen number distribution on initial and level-2 adaptive meshes for the twin-jet interaction in a near-vacuum environment.



(a)



(b)

Figure 7. 4 Initial and final domain decomposition for 32 processors for twin-jet interaction (a) initial; (b) final.

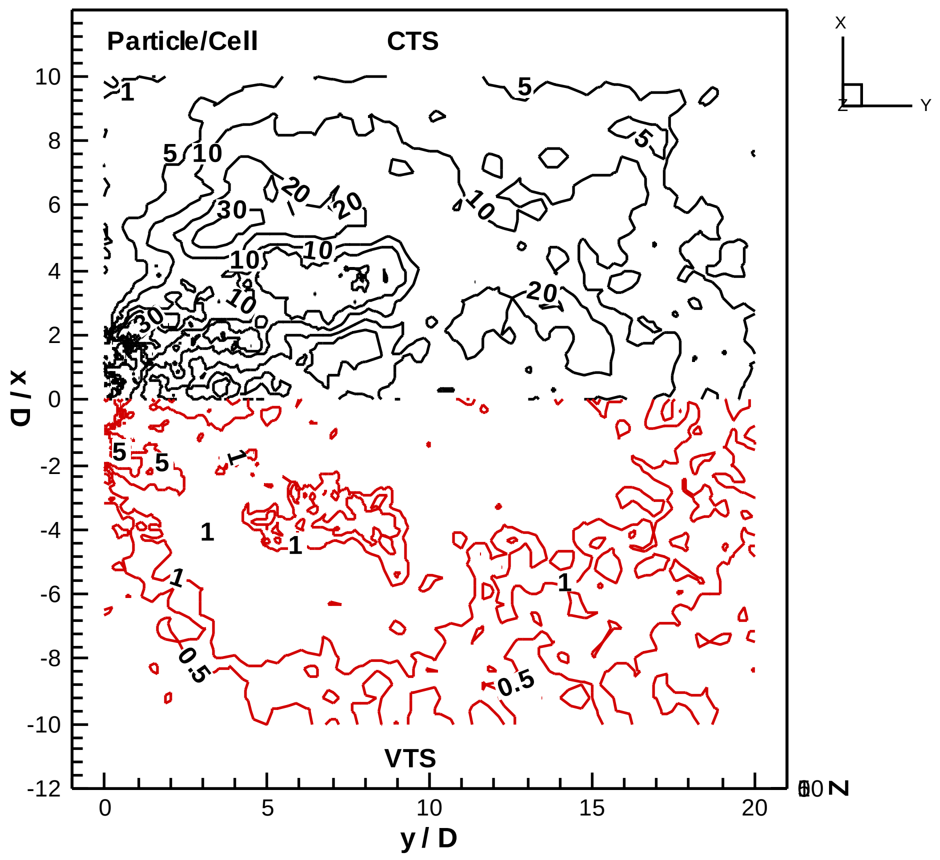


Figure 7. 5 Particle distribution for the twin-jet interaction by using constant and variable time-step schemes.

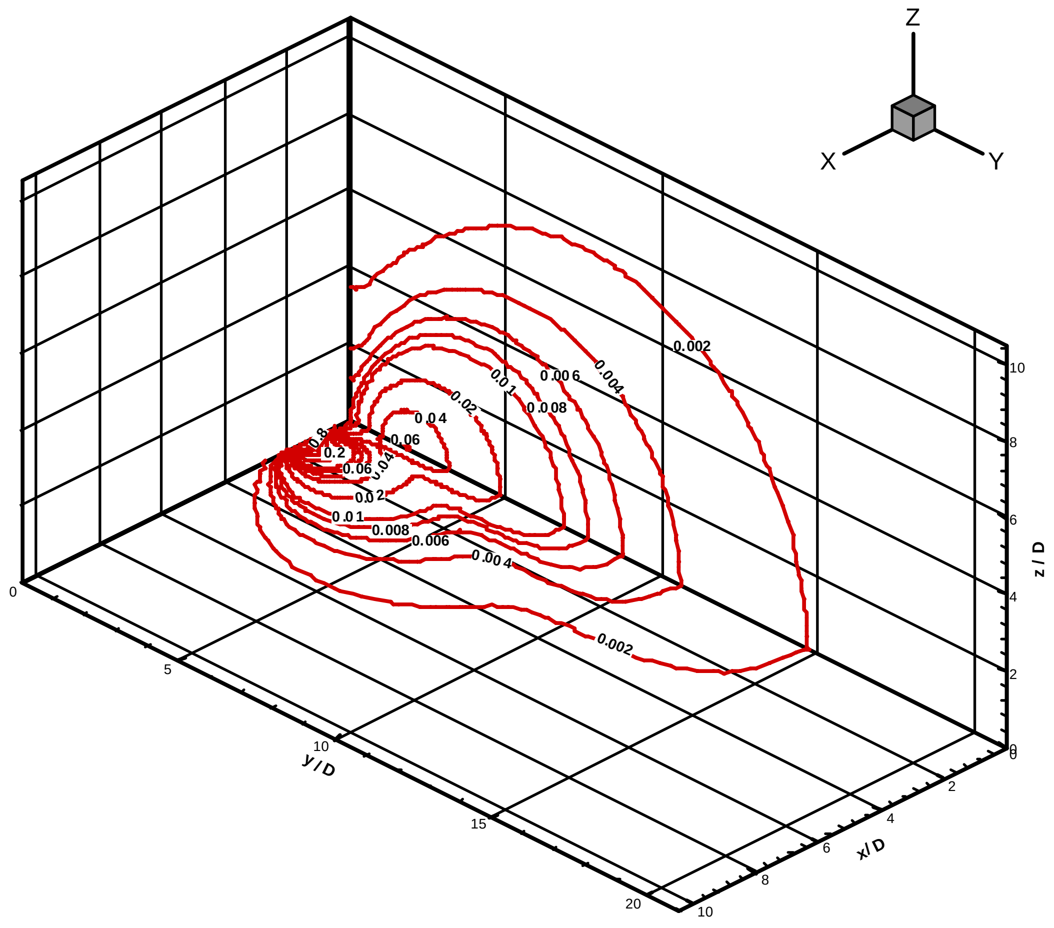
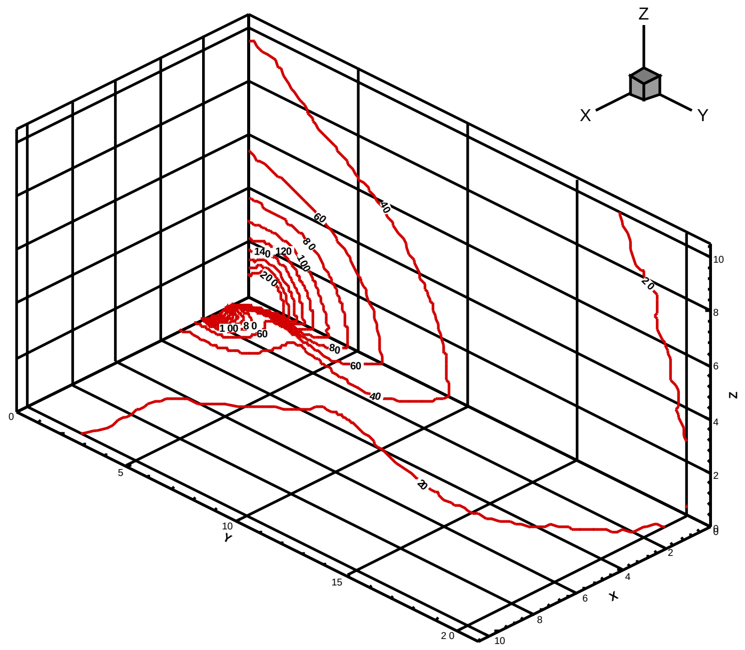
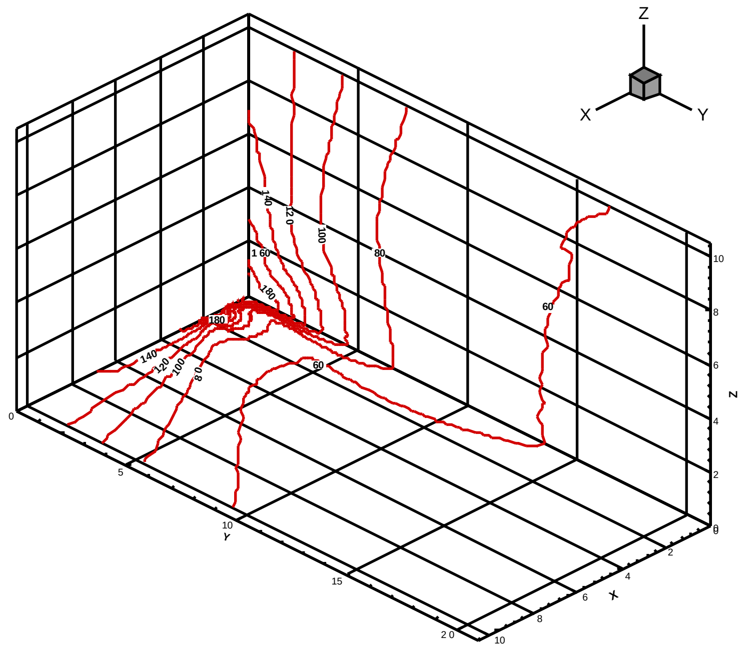


Figure 7. 6 Normalized density contours (with respect to the density at the sonic orifice) on x-y and y-z planes using vacuum outflow boundary for twin-jet interaction.

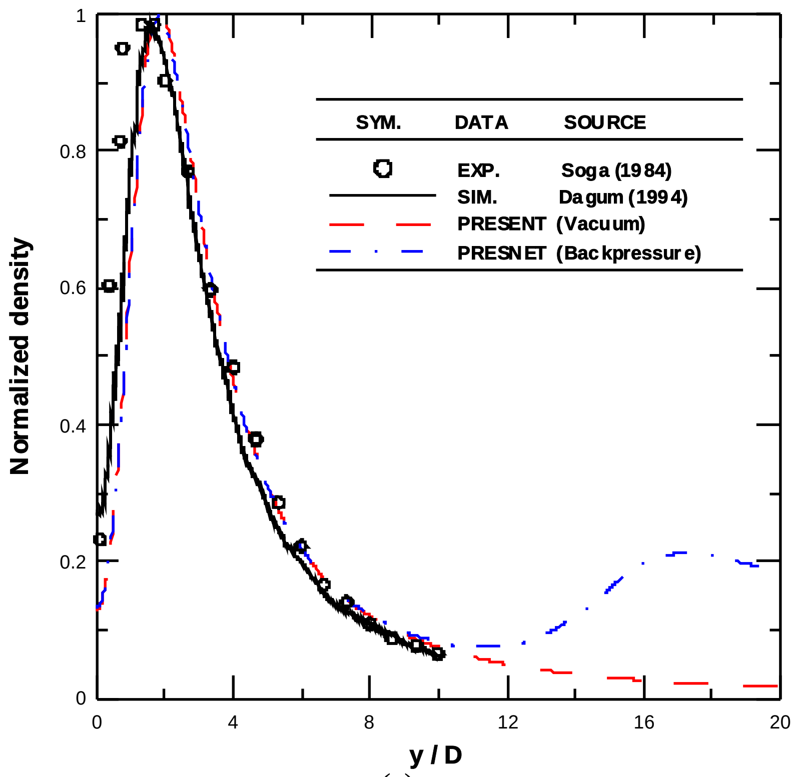


(a)

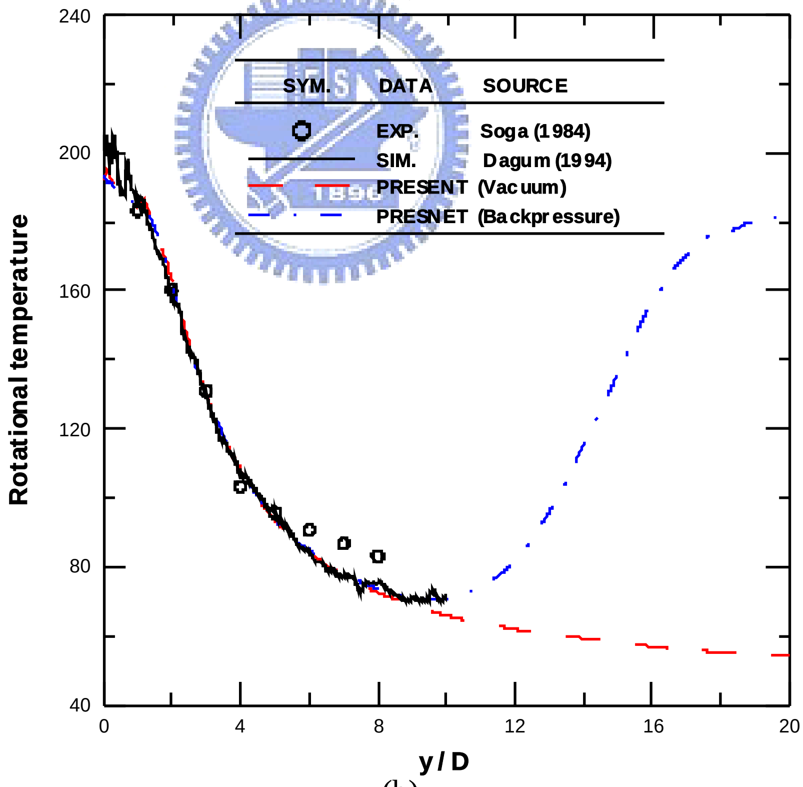


(b)

Figure 7. 7 Normalized density contours (labels in Kelvins) on x-y and y-z planes using vacuum outflow boundary for twin-jet interaction (a) translational; (b) rotational.

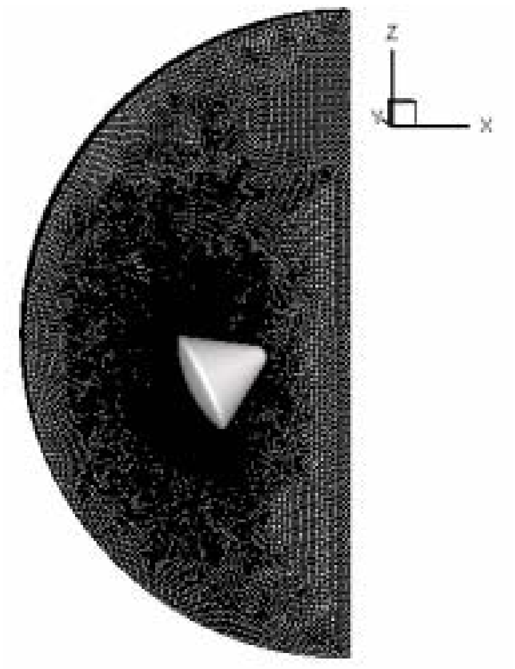


(a)

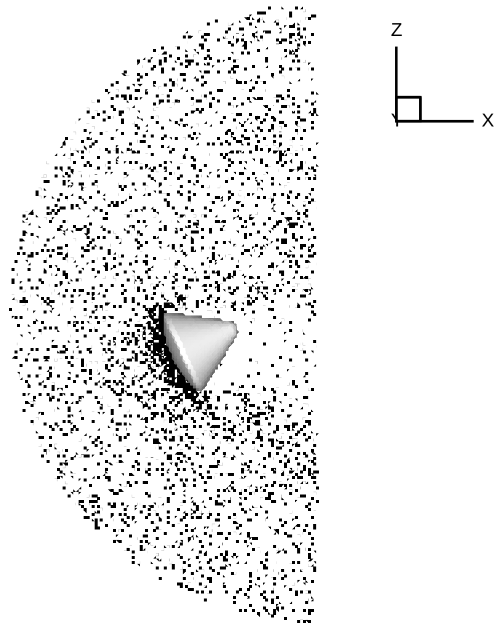


(b)

Figure 7. 8 Density and rotational temperature distribution along the symmetric centerline (y-axis) using vacuum outflow boundary for twin-jet interaction.

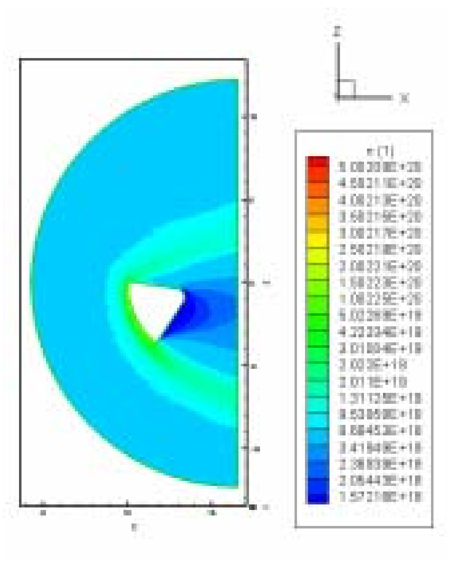


(a)

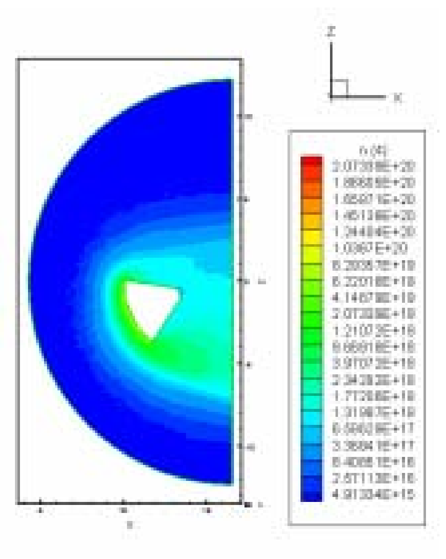


(b)

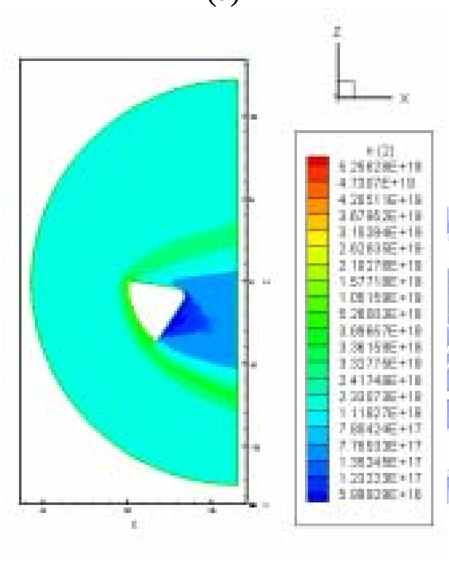
Figure 7. 9 Mesh and particle distribution of the 3-D Apollo case on X-Z plane (a) mesh; (b) particle distribution.



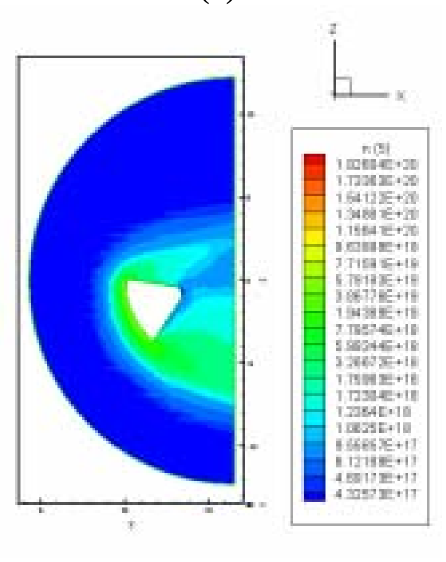
(a)



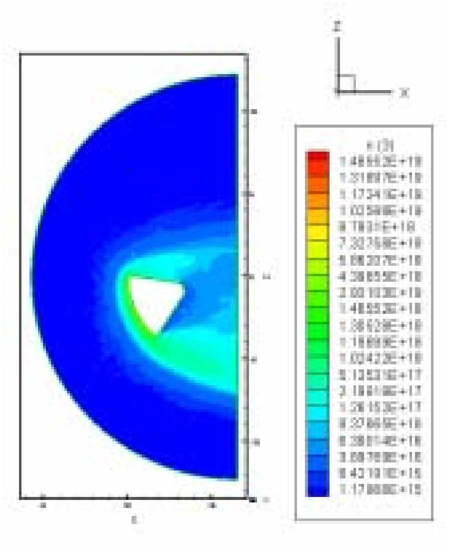
(d)



(b)

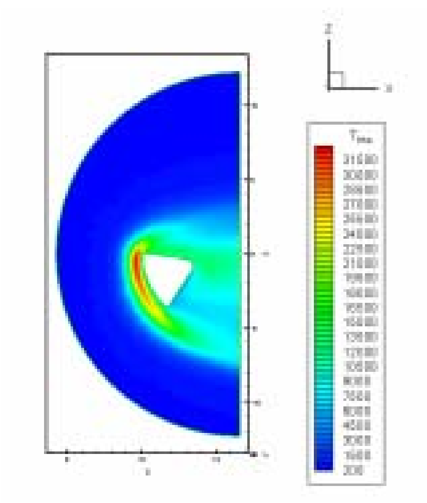


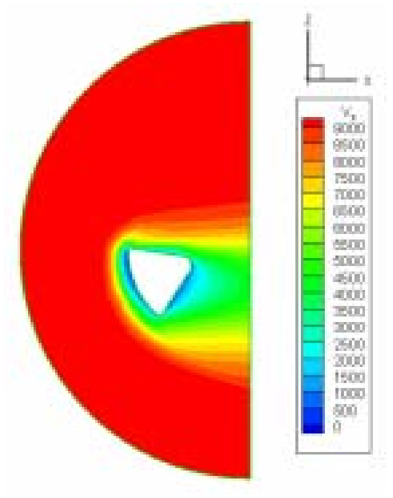
(e)

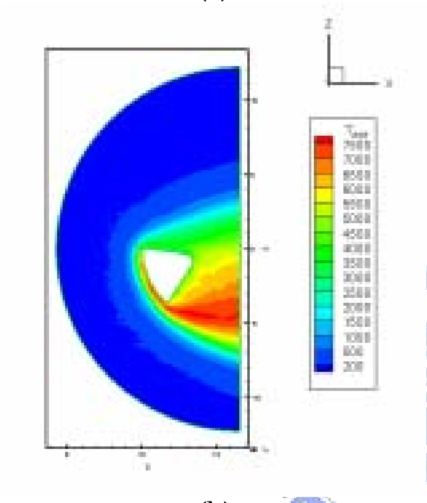


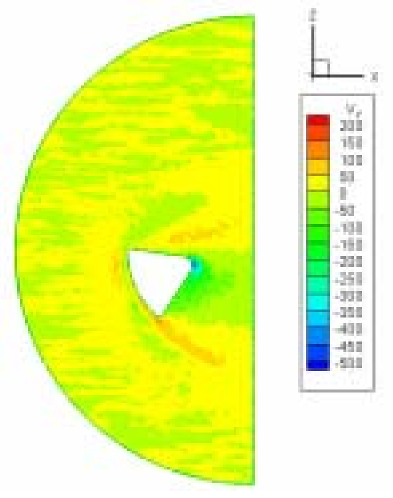
(c)

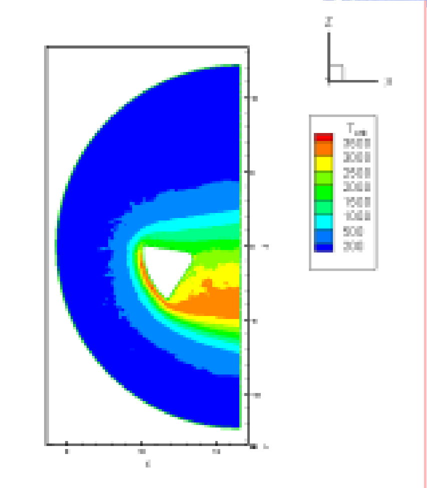
Figure 7. 10 Number density contours of the 3-D Apollo case (a) N_2; (b) O_2; (c) NO; (d) N; (e) O.


(a)


(d)


(b)


(e)


(c)

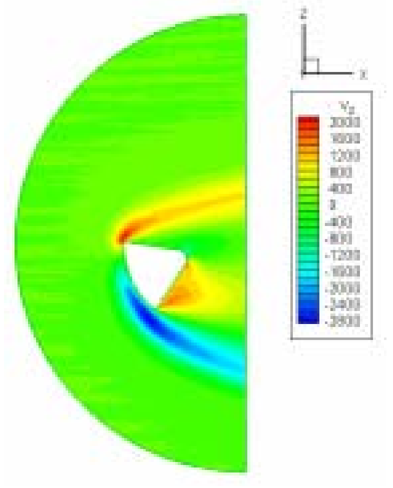
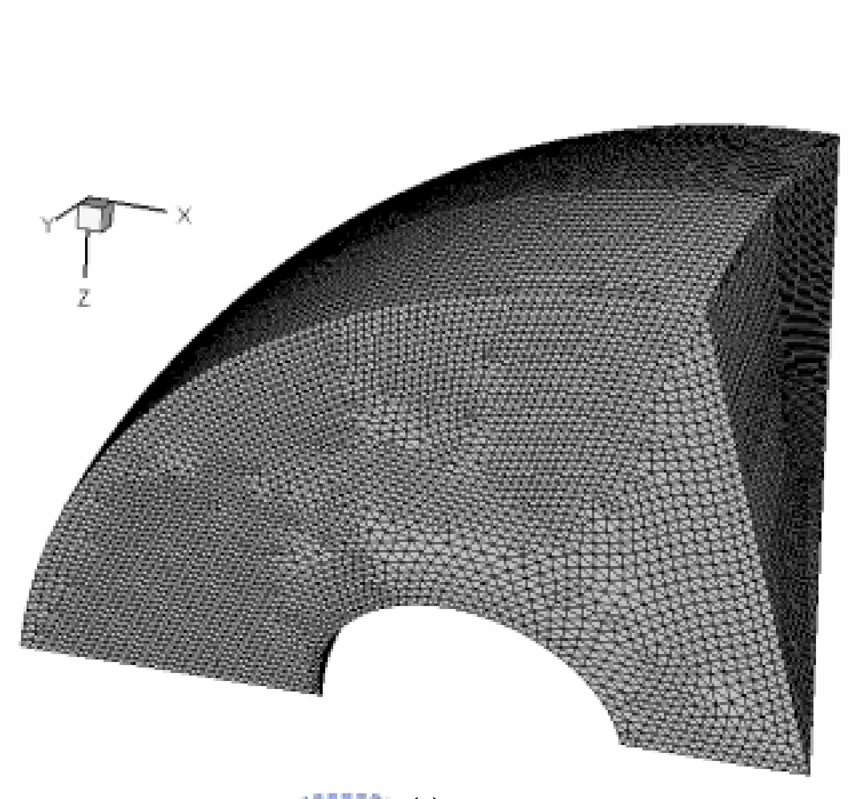
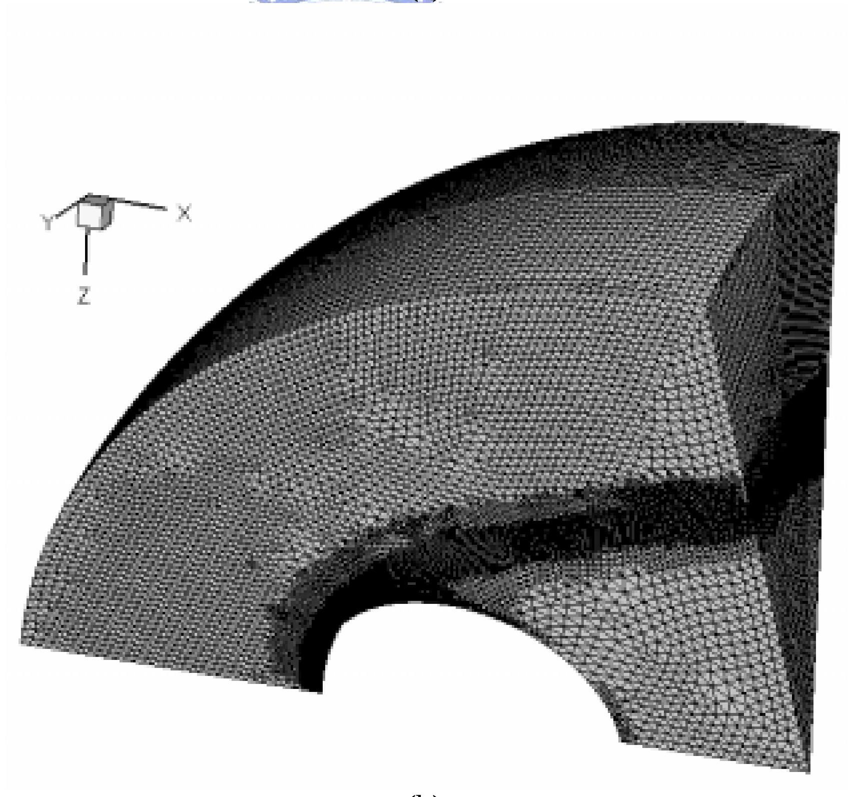

(f)

Figure 7. 11 Temperature and Velocity contours of the 3-D Apollo case (a) translational; (b) rotational; (c) vibrational; (d) U-velocity; (e) V-velocity; (f) W-velocity.

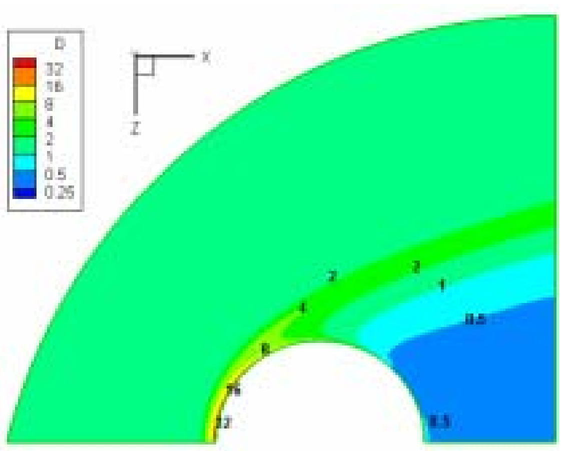


(a)

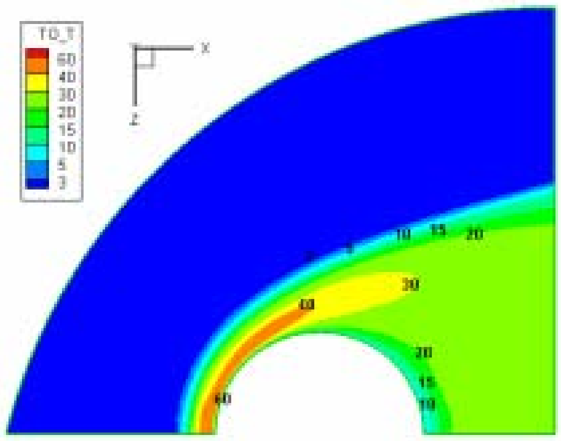


(b)

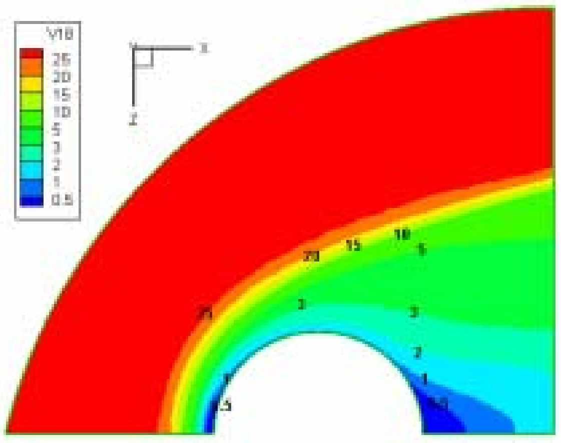
Figure 7. 12 Evolution of adaptive mesh of re-entry sphere (a) initial (100,476); (b) level-2 (669,072).



(a)



(b)



(c)

Figure 7. 13 Normalized property contours of the re-entry sphere (a) density; (b) overall temperature; (c) Mach number.

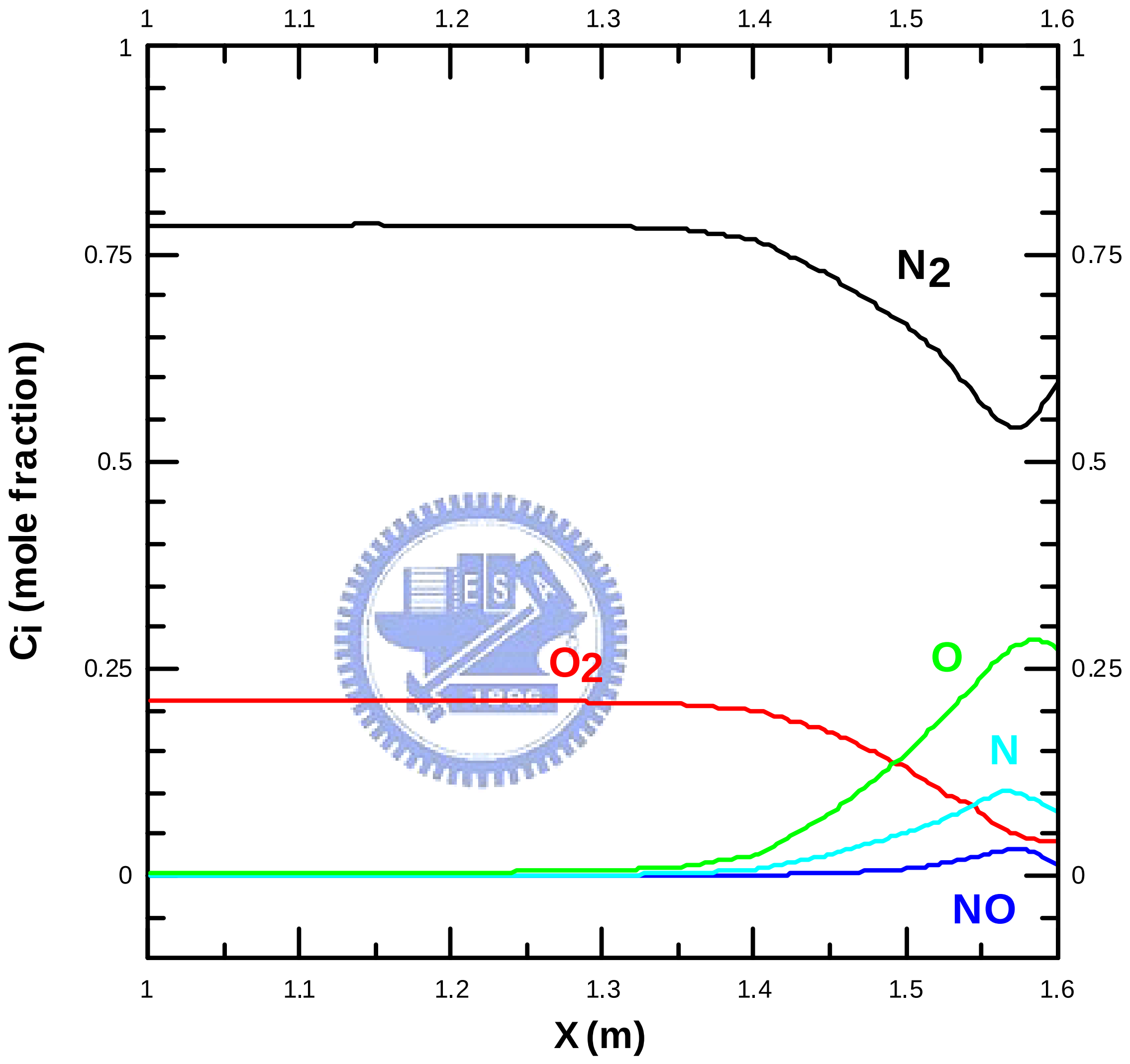
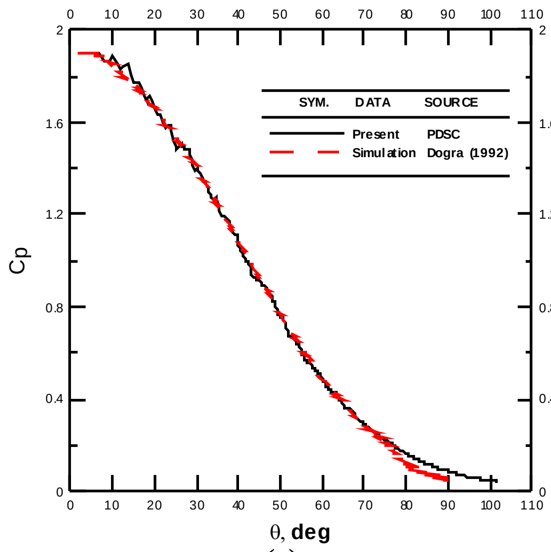
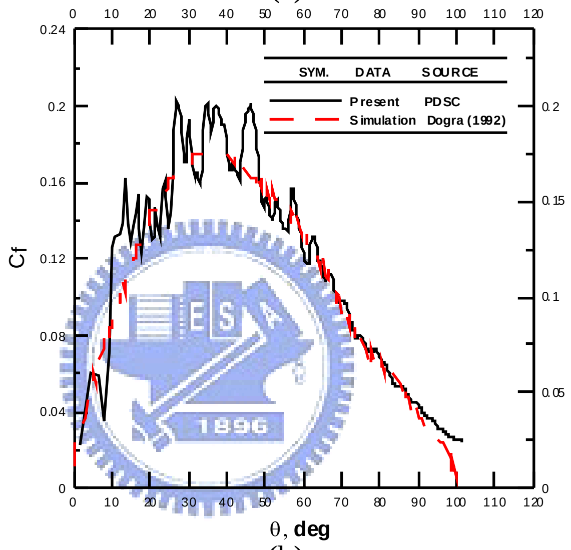


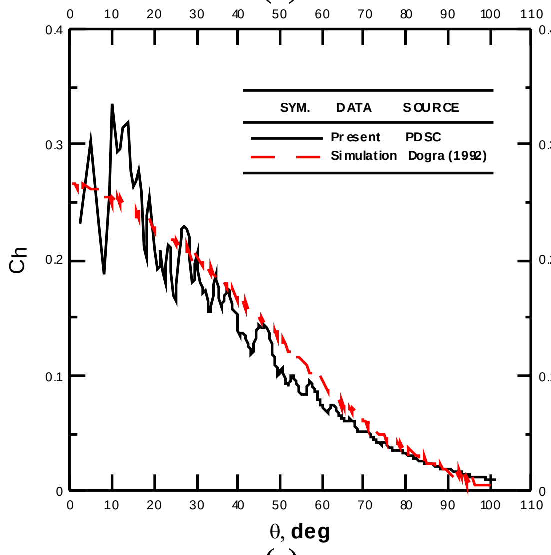
Figure 7. 14 Chemical composition along the stagnation line of the re-entry sphere.



(a)



(b)



(c)

Figure 7. 15 Surface coefficients of the re-entry sphere (a) pressure; (b) skin-friction; (c) heat transfer.

Autobiography

Kun-Chang Tseng (曾坤樟)

Phone: 03-5712121-55175

Email: gato.me89g@nctu.edu.tw

Education

2004-2005	**Visiting Scholar (Graduate Students Study Abroad Program, 國科會千里馬專案)** in Aerospace Engineering, University of Michigan, Ann Arbor, Michigan, USA The research emphasized on the chemical reaction of the Direct Simulation of Monte Carlo (DSMC)
2000-Pressent	**PhD Program** in Mechanical Engineering, National Chiao-Tung University, Hsinchu, Taiwan This research emphasized on numerical methods of the Direct Simulation of Monte Carlo (DSMC) code to increase accuracy of simulation and speed up the computation.
1998-2000	**M. S.** in Mechanical Engineering, National Chiao-Tung University, Hsinchu, Taiwan This research emphasized on the numerical analysis by using the Direct Simulation of Monte Carlo (DSMC) method for studying rarefied gas dynamics.
1994-1998	**B. S.** in Mechanical Engineering, National Chung-Cheng University, Chiayi, Taiwan

Languages

Mandarin Chinese, Taiwanese and English

Publications

Referred Papers:

1. J.-S. Wu and **K.-C. Tseng**, "Analysis of micro-scale gas flows with pressure boundaries using direct simulation Monte Carlo method", Computers & Fluids, Vol. 30, pp. 711-725, 2001 (SCI : 1.479).
2. J.-S. Wu, **K.-C. Tseng** and C.-H. Kuo, "The direct simulation Monte Carlo Method using unstructured adaptive mesh and its application", International Journal for Numerical Methods in Fluids, Vol. 38, Issue 4, pp. 351-375, 2002 (SCI : 0.544).
3. J.-S. Wu and **K.-C. Tseng**, "Parallel Particle Simulation of the Near-Continuum Hypersonic Flows Over Compression Ramps", ASME Journal of Fluids Engineering, Vol. 125, No. 1, pp. 181-188, 2003 (SCI : 0.6569). *
4. J.-S. Wu, **K.-C. Tseng** and T.-J. Yang, "Parallel Implementation of DSMC Using Unstructured Mesh", International Journal of Computational Fluid Dynamics, Vol. 17 (5), pp. 405-422, 2003 (SCI : 0.373).
5. J.-S. Wu, W-J. Hsiao, Y.-Y. Lian and **K.-C. Tseng**, "Assessment of conservative weighting scheme in simulating chemical vapour deposition with trace species", International Journal for numerical Methods in Fluids, Vol. 43, pp. 93-114, 2003 (SCI : 0.544). *
6. J.-S. Wu, **K.-C. Tseng** and F.-Y. Wu, "Parallel three-dimensional DSMC method using mesh refinement and variable time-step scheme", Computer Physics Communications, Vol. 162, No. 3, pp. 166-187, 2004 (SCI : 1.170). *
7. J.-S. Wu and **K.-C. Tseng**, "Parallel DSMC method using dynamic domain decomposition", International Journal for Numerical Methods in Engineering, Vol. 63, Issue 1, pp. 37-76, 2005 (SCI : 1.691). *
8. **K.-C. Tseng**, I. D. Boyd and J.-S. Wu, "Applications of Chemical Reactions in the Parallel DSMC Code (PDSC)", 2005. (Preparing)
9. J.-S. Wu, U.-M. Lee and **K.-C. Tseng**, "Full-scale DSMC Simulation of Plume Impingement on Satellite", 2005. (Preparing)

(* are those presented in this thesis)

Conference Papers:

1. J.-S. Wu, **K.-C. Tseng** and C.-H. Kuo, "Applications of Local Mesh Refinement in The DSMC Method", 22nd International Symposium on Rarefied Gas Dynamics, Sydney, Australia, pp. 417-425, July 9-14, 2000.
2. J.-S. Wu and **K.-C. Tseng,** "Analysis of Internal Micro-Scale Gas Flows with Pressure Boundaries Using The DSMC Method", 22nd International Symposium on Rarefied Gas Dynamics, Sydney, Australia, pp. 486-493, July 9-14, 2000.
3. J.-S. Wu, **K.-C. Tseng** and C.-H. Kuo, "The Direct Simulation Monte Carlo Method Using Unstructured Adaptive Mesh and Its Applications", The 17th National Conference on Mechanical Engineering The Chinese Society of Mechanical Engineers, NKFUST, Kaohsiung, Taiwan, December 8-9, 2000.
4. J.-S. Wu, **K.-C. Tseng**, T.-R. Yang and C.-H. Kuo, "The Parallel Implementation of The Direct Simulation Monte Carlo Method Using Unstructured Mesh and Its Application", 24th National Conference on Applied Theoretical mechanics, CYCU, Chung Li, Taiwan, December 9-10, 2000.
5. J.-S. Wu and **K.-C. Tseng,** "Parallel Direct Simulation of The Near-Continuum Hypersonic Flows Over Compression Ramps", 35th AIAA Thermophysics Conference, AIAA 2001-2894, Anaheim, California, USA, June11-14, 2001.
6. J.-S. Wu and **K.-C. Tseng,** "Concurrent DSMC Method Using Dynamic Domain Decomposition", 23rd International Symposium on Rarefied Gas Dynamics, Whistler Conference Centre Whistler, British Columbia, July 20-25, 2002.
7. J.-S. Wu and **K.-C. Tseng,** "Concurrent DSMC Method Combining Variable Time-Step and Adaptive Unstructured Mesh", Parallel CFD 2003, Moscow, Russia, May 13-15, 2003.
8. J.-S. Wu, **K.-C. Tseng**, U.-M. Lee and Y.-Y. Lian, "Development of a General Parallel Three-Dimensional Direct Simulation Monte Carlo Code (PDSC)", 24th International Symposium on Rarefied Gas Dynamics, Bari, Italy, July 10-16, 2004.
9. Jose F. Padilla, **K.-C. Tseng** and Ian D. Boyd, "Analysis of Entry Vehicle Aerothermodynamics Using the Direct Simulation Monte Carlo Method", 38th Thermophysics Conference, Toronto, Canada, AIAA Paper 2005-4681, 2005.
10. **K.-C. Tseng**, J.-S. Wu and I. D. Boyd, "Validation and Applications of Chemical-Reactions Simulation in the Parallel DSMC Code (PDSC)", 6th

Asia Computational Fluid Dynamics, Taipei, Taiwan, 2005 (Preparing).