PERGAMON

Computers & Fluids 32 (2003) 1133–1160

computers & fluids

www.elsevier.com/locate/compfluid

Parallel three-dimensional direct simulation Monte Carlo method and its applications

J.-S. Wu *, Y.-Y. Lian

Department of Mechanical Engineering, National Chiao-Tung University, 1001 Ta-Hsueh Road, Hsinchu 30050, Taiwan

Accepted 2 September 2002

Abstract

The development of a parallel three-dimensional direct simulation Monte Carlo (DSMC) method using unstructured cells is reported. Variable hard sphere molecular model and no time counter method are used for the molecular collision kinetics, while the cell-by-cell ray-tracing technique is implemented for particle movement. Developed serial code has been verified by comparing the results of a supersonic corner flow with those of Bird's three-dimensional structured DSMC code. In addition, a benchmark test is performed for an orifice expanding flow to verify the parallel implementation of DSMC method by comparing with available experimental data. Static physical domain decomposition is used to distribute the workload among multiple processors by considering the estimated particle weighting distribution. Two-step multi-level graph partitioning technique is used to perform the required domain decomposition. Completed code is then applied to compute a hypersonic flow over a sphere (external flow) and the flow field of a spiral drag pump (internal flow), respectively. Results of the former are in good agreement with previous numerical results using axisymmetric DSMC method and experimental data. Results of the latter also agree well with previous numerical results.

1. Introduction

The understanding of rarefied gas dynamics (high-Knudsen number flows) has begun to play an important role in several research disciplines. The examples include the pumping characteristics of turbo-molecular drag vacuum pump [1,2], the low-pressure plasma etching and chemical vapor deposition (LPCVD) [3], the micro-filter [4], and the micro-electro-mechanical-system (MEMS)

* Corresponding author. Tel.: +886-3-573-1693; fax: +886-3-572-0634.
E-mail address: chongsin@cc.nctu.edu.tw (J.-S. Wu).

0045-7930/03/$ - see front matter
doi:10.1016/S0045-7930(02)00083-X

[5–7]. It is well recognized that Navier–Stokes equations fail to approximate the flow behavior in the regime of rarefied gas due to the breakdown of the continuum assumption. Thus, the particle nature of the flow must be carefully taken into account to correctly model the flow.

It is well known that Boltzmann equation is more appropriate for approximating flow with high-Knudsen number; 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 modeling the integral collision term. Bird [8,9] proposed an alternative but efficient particle method known as direct simulation Monte Carol (DSMC). Later on, both Nanbu [10] and Wagner [11] were able to demonstrate mathematically that the DSMC method is equivalent to solving the Boltzmann equation as the simulated particle numbers become large.

In the past, most applications of DSMC applied structured grids [8,9; and the references cited therein]. It is much easier to program using structured grids; however, it requires tremendous problem specific modification. To alleviate such restriction, an unstructured grid system is one of the best choices, although it may be computationally relatively expensive. In addition, it is more flexible to handle objects with complicated geometry [7] and is easier for the implementation of mesh adaptation [12]. Considering the advantages using unstructured grids, its use in DSMC is highly justified. For example, Boyd's group [13,14] has applied such technique to compute the thruster plume produced by a spacecraft and found that the results are very satisfactory as compared with experimental data. In addition, Piekos and Breuer [5], Nance and Hash [6], and Wu and Tseng [7] have also applied the same grid technology to compute several typical micro-scale flows with acceptable results.

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 implemented directly to the calculation of particle collisions. However, high computational cost has hindered further applications of DSMC to the practical three-dimensional problem. Hence, it is important to increase the computational speed to extend the applicability of the DSMC method.

Several remedies have been suggested to improve the computational speed of the DSMC method. One of them is to vectorize the DSMC code. However, further development in the vectorization of DSMC seems more or less ceased due to the following three reasons. First, the very limited resources are available for supercomputer computation. Second, the performance improvement and mass production of the superscalar computers (workstations or personal computers) drives the simulation cost down further on these superscalar computers. Third, it is difficult or impossible to fully vectorize the DSMC algorithm; for example, too many logic statements (e.g., if ... then) exist in the code. Thus, the recent (since 1990s) research interest in speeding up the DSMC computation has evolved from vectorized DSMC to parallel DSMC.

Since the DSMC method is a particle-based numerical method, the movement of each particle is inherently independent of each other. The coupling between particles is made through collision in the cells. Hence, as compared with other N–S based numerical schemes, the DSMC method is nearly 100% in parallelism. In addition, as mentioned previously, it becomes more affordable to execute the DSMC method using the parallel processors rather than the vectorized supercomputer. Therefore, the implementation of DSMC method on the parallel processors is highly justified.

In the past, several studies on parallel implementation of DSMC have been published [15–19] using static domain decomposition and structured mesh system. Message passing was used to transfer molecular data 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.

Boyd's group [14,20] designed a parallel DSMC software named MONACO. In this code, unstructured grids were used to take the advantage of flexibility of handling complex object geometry. In addition, a new data structured was proposed to meet the specific requirement related to workstation hardware architecture while maintaining high efficiency on supercomputers. Timing results showed the performance improvement on workstations and the necessity of load balancing for achieving high performance on parallel computers. Coordinate partitioning technique was used for domain decomposition. The authors also suggested that only decomposition along streamlines should be chosen to keep communication at a minimum. However, it may not be always possible for realistic flow problems. Scalability issue has not been discussed in their reports. Maximum 400 processors of IBM SP-2 were used throughout the research.

Recently, Wu et al. [34] has successfully applied the graph partitioning technique (METIS [25] and JOSTLE [27–29]) to physically decompose the computational domain using unstructured mesh and then compute several external hypersonic flows using parallel 2-D DSMC method. It was shown that results using JOSTLE [27–29] is better than those using METIS [25] in terms of parallel speedup and efficiency. In their work, a special cell numbering scheme, called *globally sequential but locally unstructured* scheme, has been developed to take advantage of the unstructured data format.

Until very recently, dynamic load balancing technique, using stop at rise (SAR) based on a degradation function, was used in conjunction with the parallel implementation of DSMC method [21,22]. Parallel efficiency up to 90% was reported for 128 processors for flow over a sphere [21], although it is not clear how they implemented the dynamic load balancing. However, a data mapping between the local (in each sub-domain) and the global (whole domain) cell numbers was required, which became very costly for a large simulation otherwise [22].

In the current study, we will concentrate on extending previous implementation of static domain decomposition on two-dimensional DSMC to three-dimensional DSMC. However, the design of the current parallel implementation will take into account the future implementation of dynamic domain decomposition. Static decomposition methods fall into two categories as pointed out by Robinson [22]. These are geometry-based and graph-based. 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 (edge cut) and poor load balancing. On the other hand, graph-based methods provide better E_c and better load balancing if some careful measures are imposed. 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 centers) can be weighted with the particle numbers with all edges having unitary weight. Thus, domain decomposition in DSMC can become very efficient by taking the advantage of success in graph partitioning research. Related description and reviews of graph partitioning can be found in the Refs. [23–29] and are not repeated here for brevity. In the current study, we use JOSTLE [27–29] as the partitioning tool for three-dimensional domain decomposition and is briefly introduced in the following.

JOSTLE [27–29] initiates the domain decomposition 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 unbalancing across the processor array.

As a first step to reduce the computational time of practical three-dimensional problems, we decompose the computational domain using JOSTLE [27–29], by considering the total particle weighting in each sub-domain obtained from the serial-code simulation with very few simulated particles (e.g., less than 3%).

From previous review, the objectives of the current study are summarized as follows.

1. To complete a serial three-dimensional DSMC code using unstructured grids and verified by comparing the results of a supersonic corner flow with those of Bird's three-dimensional structured DSMC code [8].
2. To complete a parallel three-dimensional DSMC code, applying graph partitioning technique for domain decomposition, and to be verified by comparing the results of an orifice expanding flow with the experimental data of Weaver et al. [31].
3. To apply the parallel DSMC code to compute the flow fields of the hypersonic flow over a sphere (external flow) and a spiral groove vacuum pump (internal flow), respectively.

This paper is organized as follows: the DSMC method is described next, then the method is verified by two benchmark tests, and is applied to compute external and internal flow, respectively. Finally, the paper is summarized with some important conclusions.

2. Numerical method

2.1. The DSMC method

The DSMC method [8,9] is a particle method for the simulation of 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 are considered at the particle level. The method is statistical in nature. 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 use VHS molecular model [9] to reproduce real fluid behavior as well as no time counter (NTC) [9] for the collision mechanics. Details of the procedures and the consequences of the computational approximations regarding DSMC can be found in [9]. In addition, cells in DSMC are mainly used for particle collision and sampling. Hence, adoption of different grid system will definitely affect the practical DSMC implementation, especially the method of particle tracking. A special ray-tracing technique has to be designed to efficiently

track the particle movement for the special grid system (unstructured) we use in the current study.

2.2. 3-D particle ray-tracing in unstructured grids

Particle ray-tracing is performed cell-by-cell in unstructured grids by taking the advantage of cell connectivity provided by the unstructured mesh data. Briefly speaking, the first step of the particle tracking is to determine whether the particle will stay in or leave the current cell (where the particle originally exists). If the particle leaves, then the second step is to determine the intersecting position on the corresponding face. Further journey of the particle depends on the face condition. If it is a normal face between cells, then it will continue its movement until the time step ends. 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. Note that the method of particle ray-tracing technique in the current study is conceptually similar to previous work, which was developed by Kannenberg [14]. Detailed procedures of the particle ray-tracing technique are described next in a more concise mathematical format.

Without considering the external force effects, free-flying position of traced particle at $t + \Delta t$ can be written as, on one hand,

$$\overline{P}_{\mathrm{f}}(t) = \overline{P}_i + \overline{V} \cdot \Delta t \tag{1}$$

where

$$\overline{P}_{\mathrm{f}}(t) = \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \text{final particle position column vector} \tag{1a}$$

$$\overline{P}_i(t) = \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} = \text{initial particle position column vector} \tag{1b}$$

$$\overline{V} = \begin{pmatrix} u \\ v \\ w \end{pmatrix} = \text{particle velocity column vector} \tag{1c}$$

On the other hand, cell face can simply be represented as a planar equation as

$$\bar{n} \cdot \bar{p} + d = 0 \tag{2}$$

where $\bar{n} = (a, b, c)$ is the normal unit vector of the face and $\bar{p} = (x, y, z)$ is the position vector on the plane.

By solving Eqs. (1) and (2), we have

$$\Delta t' = \frac{-(ax_i + by_i + cz_i + d)}{(au + bv + cz)} \tag{3}$$

By computing in turn Eq. (3) of each face in the current cell, the correct intersecting face number can be identified by finding the minimum positive $\Delta t'$. The intersecting coordinate can be found by

substituting $\Delta t'$ into Eq. (1). If the intersecting face is a normal face between cells, then the particle will continue its trajectory until it stops. If the intersecting face is a solid face, the particle will be reflected in a special way (e.g., diffusively or specularly) according to the specified boundary condition. These are related by the coordinate transformation matrix between the local coordinate system (on the face) and the absolute coordinate system for both types of conditions. First, a unit vector $\bar{x}'$ along the face is chosen, then $\bar{y}'$ is the cross product of $\bar{x}'$ and $\bar{z}'$ (the normal unit vector of the face)

$$\bar{y}' = \bar{z}' \times \bar{x}' \tag{4}$$

The coordination transformation matrix $\overline{\overline{H}}$, is

$$\overline{\overline{H}} = \begin{bmatrix} \bar{x}' \\ \bar{y}' \\ \bar{z}' \end{bmatrix} \tag{5}$$

Furthermore, due to the orthonormal set of $\bar{x}'$, $\bar{y}'$ and $\bar{z}'$, so the inverse transformation $\overline{\overline{H}}^{-1}$ can be easily written as

$$\overline{\overline{H}}^{-1} = \overline{\overline{H}}^{\mathrm{T}} \tag{6}$$

where $\overline{\overline{H}}^{\mathrm{T}}$ is the transpose matrix of $\overline{\overline{H}}$.

Now, the particle velocity can be transformed from the velocity in absolute coordinate system ($\overline{V}_{\mathrm{abs}}$) to the velocity in local coordinate system ($\overline{V}_{\mathrm{loc}}$) before the reflection by using $\overline{\overline{H}}$.

$$\overline{V}_{\mathrm{loc}} = \overline{\overline{H}}\,\overline{V}_{\mathrm{abs}} \tag{7}$$

After the reflection of the particle, the new local coordinate system velocity ($\overline{V}'_{\mathrm{loc}}$) can be written as

$$\overline{V}'_{\mathrm{loc}} = F(\overline{V}_{\mathrm{loc}}, \text{wall condition}) \tag{8}$$

where $F(\overline{V}_{\mathrm{loc}},$ wall condition) is a kernel function, depending upon the wall condition and velocity before reflection.

Finally, the absolute velocity after the reflection ($\overline{V}'_{\mathrm{loc}}$) will be obtained by using the inverse transformer $\overline{\overline{H}}^{-1}$ as

$$\overline{V}'_{\mathrm{abs}} = \overline{\overline{H}}^{-1}\overline{V}'_{\mathrm{loc}} = \overline{\overline{H}}^{\mathrm{T}}\overline{V}'_{\mathrm{loc}} \tag{9}$$

Then, the particle continues its journey with its new absolute velocity until it stops.

2.3. Parallel DSMC method

The DSMC algorithm is readily parallelized through 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 domain. Parallel communication occurs when particles cross the domain (processor) boundaries and are then transferred between processors. Fig. 1 illustrates a simplified flow chart of the 3-D parallel DSMC method proposed in the current study.

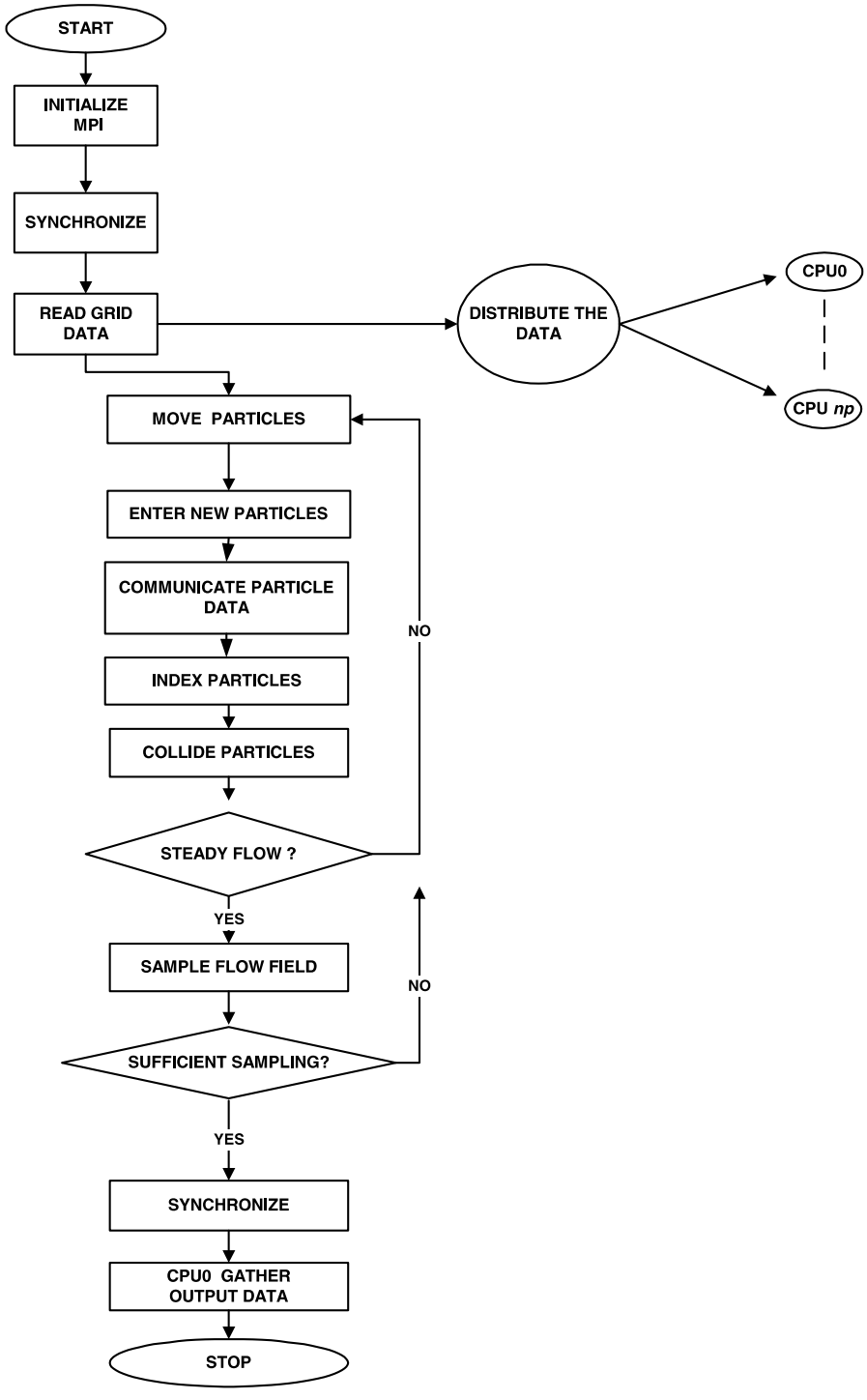

Fig. 1. Simplified flow chart of the parallel DSMC method for $np + 1$ processors.

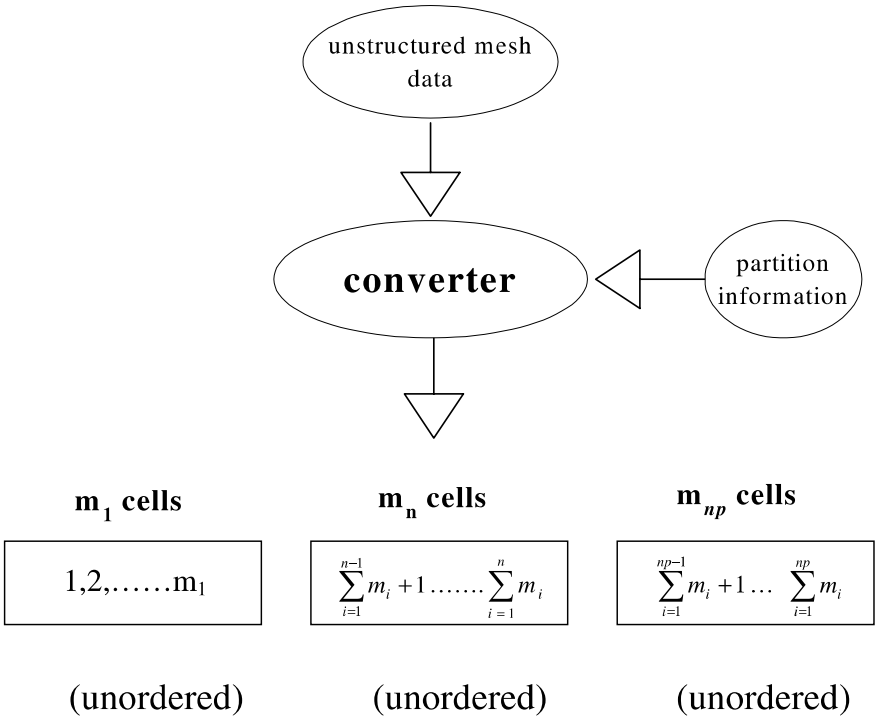

Fig. 2. Schematic diagram of the proposed cell numbering scheme.

In this method, an approach of handling the cell related data is proposed. We use *HyperMesh*™ [30] to construct an unstructured tetrahedral (or hexahedral) mesh by the advancing front method. Then, a preprocessor (or "converter") is designed to convert the unstructured mesh data into the *globally sequential but locally unstructured* mesh data for each processor in conformation with the partitioning information, as schematically presented in Fig. 2. Memory in each processor only stores the starting and ending cell numbers, rather than the global cell related data. 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. This simple conversion reduces dramatically the memory cost otherwise required for storing the mapping between the local cell number on each processor and the global cell number in the computational domain. In addition, a processor neighbor-identifying array is created for each processor, which is used to identify the surrounding processors due to the unstructured format of the processor distribution in the domain. From our practical experience, the maximum numbers of processor neighbor are less than 12; therefore, the increase of memory cost is negligible. The resulting globally sequential but locally unstructured mesh data is then imported into the parallel DSMC code.

After reading the mesh data on a master processor (cpu0), the mesh data are then distributed to all other processors according to the designated domain decomposition. All the particles on each processor then start to move as in sequential algorithm. The particle data are sent to a buffer and is numbered sequentially when hitting the inter-processor boundary (IPB) during its journey within a simulation time step. After all the particles on a processor are moved, the destination

processor for each particle in the buffer is identified via a simple arithmetic computation, owing to the approach adopted for the cell numbering, and are then packed into arrays. Considering communication efficiency the packed arrays are then sent as a whole to its surrounding processors in turn based on the tagged numbers. 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.

After all particles on each processors have come to their final locations 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 on each processor as usual in a sequential DSMC code. The particles in each cell are then sampled at the preset appropriate time.

High parallel efficiency can only be achieved if communication is minimized and the computational load is evenly distributed among processors. To minimize communication for static domain decomposition, the boundaries between sub-domains should lie along the streamlines of the flow field [20] as mentioned previously; however, it is nearly impossible to achieve this partition for most of the practical flows. Fortunately, the advancement of networking speed has reduced the communication time between processors to a tolerable quantity. For the DSMC algorithm, the workload (or equivalently particle numbers) on each processor changes frequently, especially during the transient period of a simulation, while the workload attains a roughly constant value during steady-state sampling. Thus, a truly dynamic (or adaptive) domain decomposition technique is required to perfectly balance the workload among the processors. However, as a first step towards this objective, we have instead adopted the static domain decomposition method to provide the partitioned sub-domains. We use a state-of-the-art graph partitioning tool, JOSTLE [28], by considering the estimated particle weight at each graph vertex (cell center). The weight at each graph vertex is estimated by running the sequential (serial) DSMC code with about 3% (or less) of the total particle numbers actually used for a parallel simulation. The variations of weight at vertices obtained this way can be shown later that they are roughly equivalent to those obtained by running all particles (100%) for a large granularity problem.

Although DSMC possesses nearly 100% parallelism (except for initialization and final output), both the values of speedup and efficiency are expected to be lower than the ideal values due to the load unbalancing and communication as mentioned previously.

3. Results and discussion

3.1. Verifications

3.1.1. Supersonic corner flow

We have used a supersonic corner flow [8] as the first test problem for the verification of the proposed particle ray-tracing algorithm in three-dimensional unstructured sequential DSMC method. This test problem is a supersonic flow in the 'corner' between two flat plates that are

perpendicular to each another and parallel to the stream. The free stream is argon gas at temperature of 300 K, number density of 10^{20} m^{-3} and Mach number of 6. The length (x-axis), height (y-axis), and the depth (z-axis) of the computational domain are 0.3, 0.18, and 0.18 m, respectively. The leading edge of the plates is 0.05 m from the upstream face ($x = 0$) of the computational domain and the plates are located at $y = 0$ and $z = 0$, respectively, and are assumed as specular [8]. The plate-wall temperatures are all set to 1000 K and diffusive (100% full thermal accommodation). The corresponding Knudsen number, based on the free-stream mean free path and the length of the plate, is 0.043.

Uniform hexahedral cells (9720) and approximately 100,000 particles are used in the current simulation and the time-step (2.0×10^{-6} s) is chosen as smaller than free-stream mean collision time. Corresponding ratio of the cell size to free-stream mean free path is 0.775. 200,000 time steps are used for data samplings. Results of pressure coefficient (C_p) on the plate walls are shown in Fig. 3 along with the results using Bird's code [8]. Similar trends are found for skin frictions (C_f)

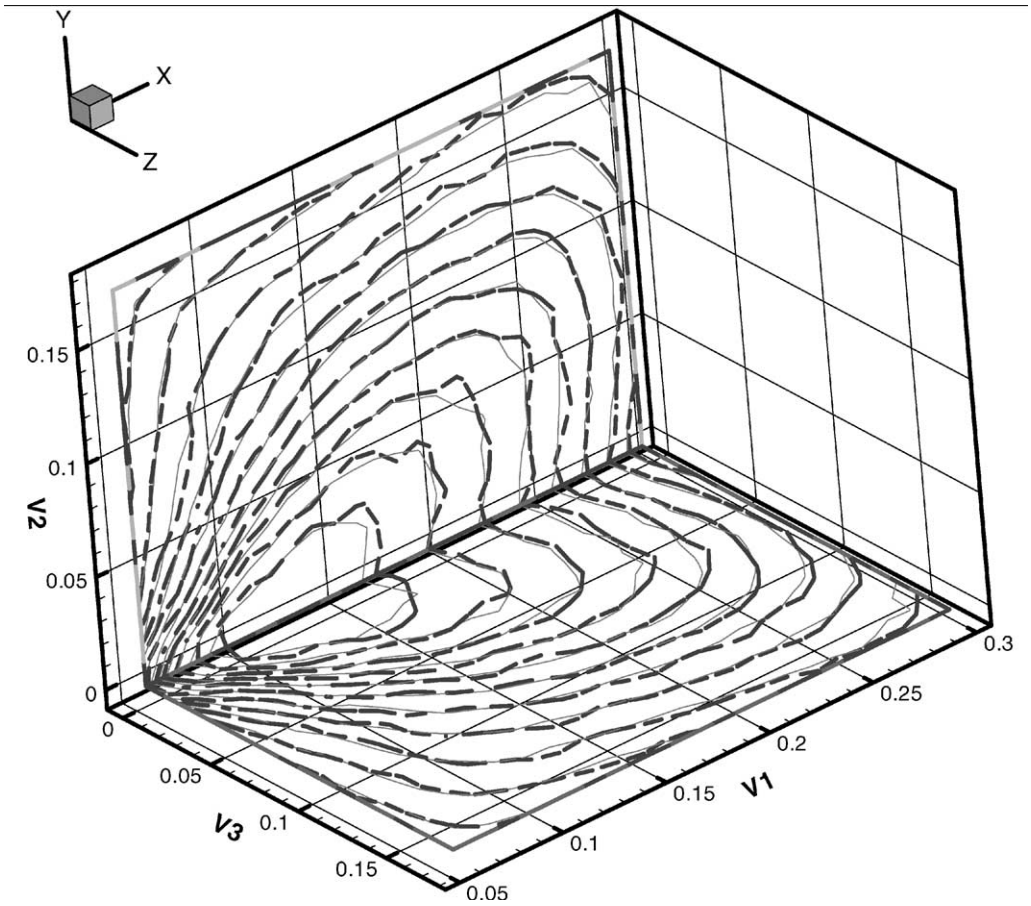

Fig. 3. Contour of pressure coefficient over the surface. The dashed lines are the result of the present data while the solid lines are those of Bird's 3-D unstructured code (Ar gas, $T_\infty = 300$ K, $n_\infty = 1 \times 10^{20}$ m^{-3}, $Ma = 6$, $L = 0.3$ m, $Kn_\infty = 0.043$).

and heat transfer coefficient (C_h). The flow is symmetrical about a plane that includes the x-axis and bisects the angle between the plates. The agreement between the present results and those of Bird's three-dimensional structured code is excellent, which verifies the correct implementation of the proposed three-dimensional particle ray-tracing technique in unstructured mesh.

3.1.2. Sonic orifice expanding flow

The test problem to verify the three-dimensional parallel DSMC implementation is a sonic orifice expanding flow [31]. The inlet condition at the orifice throat is the nitrogen gas with temperature of 261.4 K and number density of 4×10^{22} m^{-3} at a Mach number of 1 (sonic). Orifice diameter is 1.984 mm. By taking into account the flow symmetry, only one-fourth of the flow-field is used as the computational domain. The length (x-axis), height (y-axis), and the depth (z-axis) of the system grid are 39.68, 39.68 and 39.68 mm, respectively. The wall temperature is 296 K and the wall surface is assumed to be diffusive. The corresponding Knudsen number, based on the mean free path at stagnation state and the diameter of the orifice, is 0.0163.

Total 6414 non-uniform tetrahedral cells as illustrated in Fig. 4 (only surface mesh is shown), and approximately 440,000 particles are used in the current simulation. The time-step (5.0×10^{-7} s) is chosen larger than mean collision time at the throat due to the savings of computational time. Related cell sizes are about five times as compared with those at throat conditions. This might incur some concerns of accuracy problem. But it should be acceptable since the current test problem is mainly used for testing the parallel implementation. Fig. 5 illustrates a typical domain

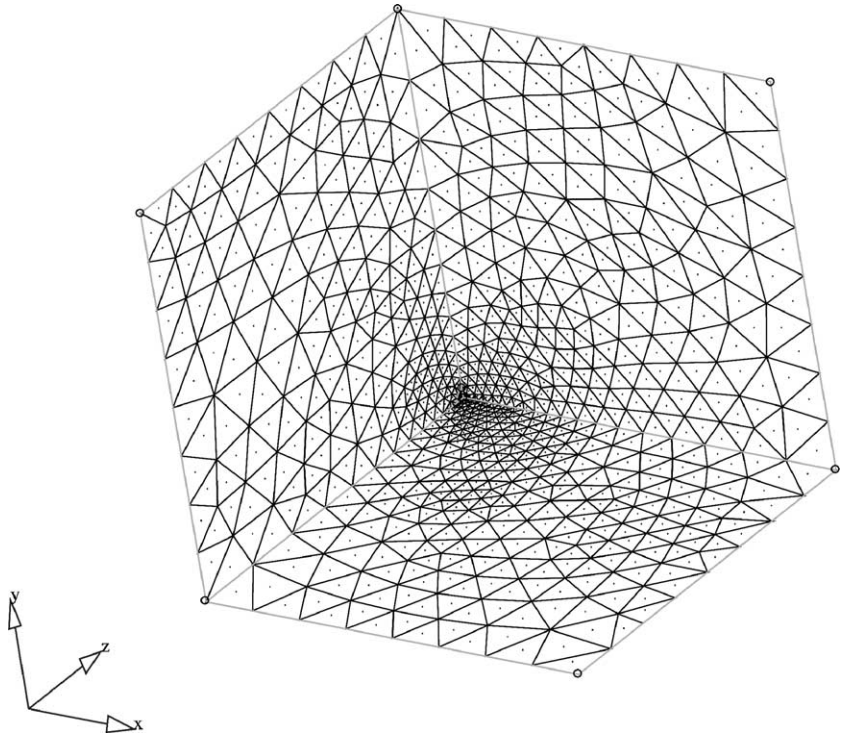

Fig. 4. Surface mesh distribution of orifice expanding flow.

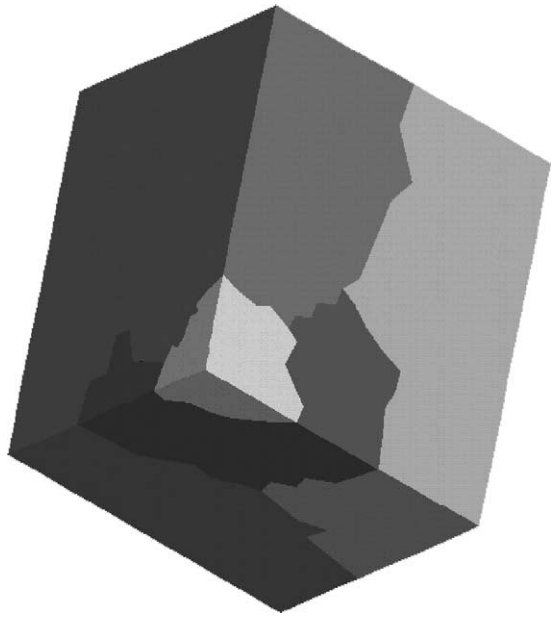

Fig. 5. Surface domain decomposition of orifice expanding flow.

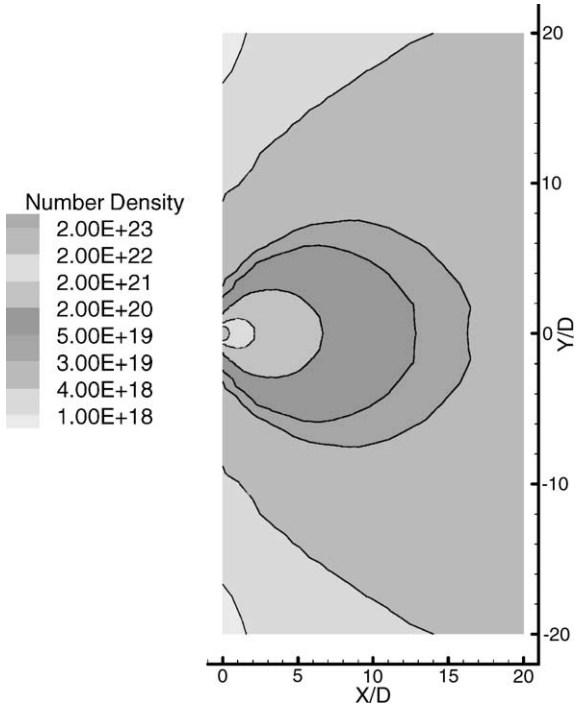

Fig. 6. Number density contour of orifice expanding flow at $Z/D = 0$ (N_2 gas, $Ma_i = 1$, $n_i = 4 \times 10^{22}$ m^{-3}, $T_i = 261.368$ K, $Kn_i = 0.0164$).

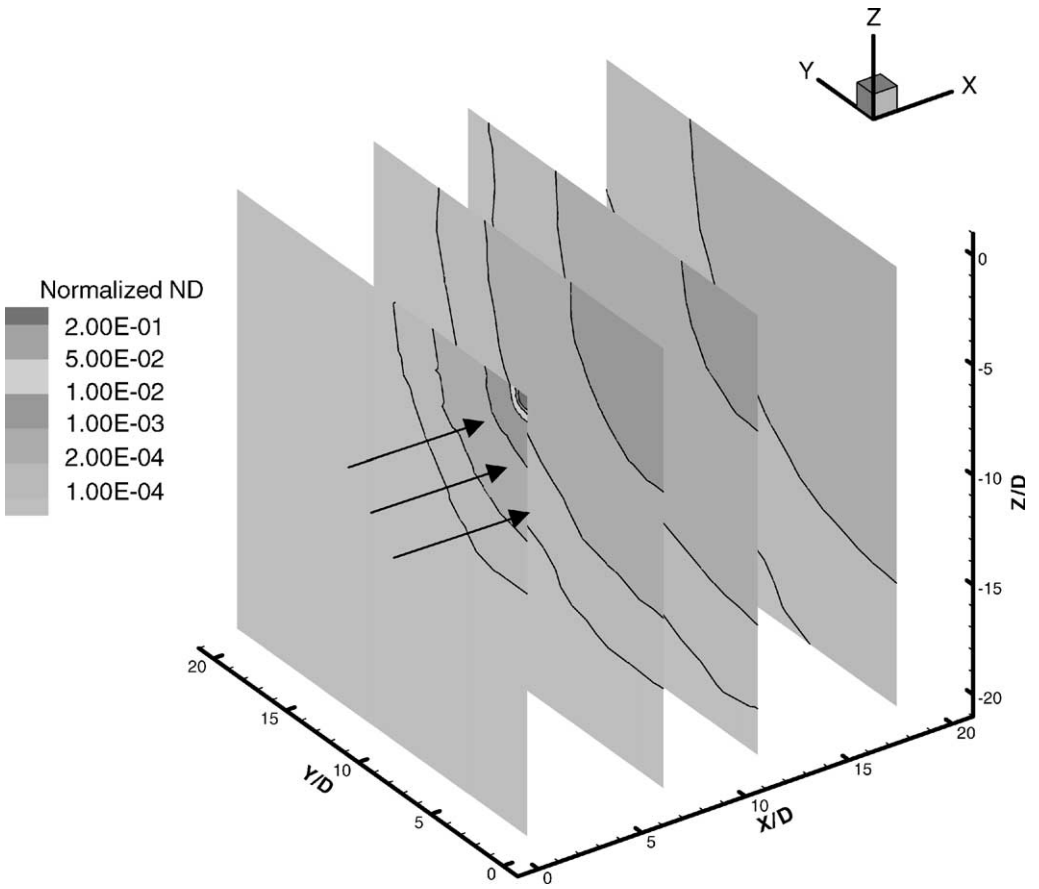

Fig. 7. Normalized number density contours of orifice expanding flow at $X/D = 0$, 6.74, 11.3 and 18, respectively (N_2 gas, $Ma_i = 1$, $n_i = 4 \times 10^{22}$ m^{-3}, $T_i = 261.368$ K, $Kn_i = 0.0164$).

decomposition using JOSTLE for four processors (only surface decomposition is shown), which is estimated by the results of approximately 12,000 particles. Parallel efficiency using four processors in this test problem is about 105%, which is superlinear caused by "cache" effects of super-scalar machine. Computed normalized number density is shown in both Figs. 6 and 7. From the results of Fig. 6, gas flow expands rapidly in the axial direction, which reduces the density about three order of magnitude within an axial distance of 20 times of orifice diameter. Sectioned normalized number density contour, as shown in Fig. 7, clearly demonstrates that the flow field is symmetric in azimuth direction at each X/D. By plotting the velocity vectors along with the streamlines (Fig. 8), we can see the flow expands rapidly not only in the axial direction but also in the radial direction, due to the vacuum boundary condition imposed. Fig. 9 illustrates the computed centerline number density along with the experimental data of Weaver et al. [31]. The agreement between the present computed results with the experimental data is excellent considering the experimental uncertainties. Similar agreement is found as well in Fig. 10, which illustrates the computed radial distribution of normalized number density along with the experimental data [31] at different axial positions.

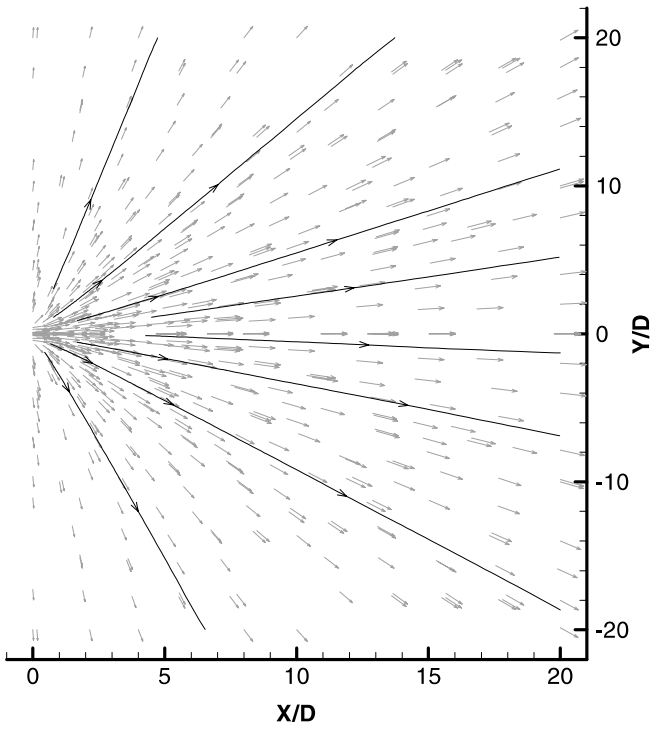

Fig. 8. Streamlines of orifice expanding flow at $Z/D = 0$ (N_2 gas, $Ma_i = 1$, $n_i = 4 \times 10^{22}$ m^{-3}, $T_i = 261.368$ K, $Kn_i = 0.0164$).

Results of the above test problems verify both the proposed three-dimensional particle ray tracing in unstructured mesh as well as the parallel implementation of three-dimensional DSMC method. Next, we will apply this method to compute typical rarefied gas flows involving both an external and an internal flows.

3.2. Applications

To demonstrate the powerful computational capability of the current parallel implementation, we have applied it to compute two applications, including a hypersonic flow over a sphere (external flow) and the flow field of the spiral-groove drag pump (internal flow). The results are then compared with previous simulation and experimental studies whenever available.

3.2.1. A hypersonic flow over a sphere

3.2.1.1. Flow and simulation conditions. Considering a hypersonic flow over a sphere, related flow conditions are listed as follows: VHS nitrogen gas, free-stream Mach number $M_\infty = 11.25$, free-stream number density $n_\infty = 8.61 \times 10^{20}$ particles/m^3, free-stream temperature $T_\infty = 22.8$ K, stagnation temperature $T_o = 600$ 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 0.078, based on the free-stream mean free path and diameter of the sphere. These flow conditions

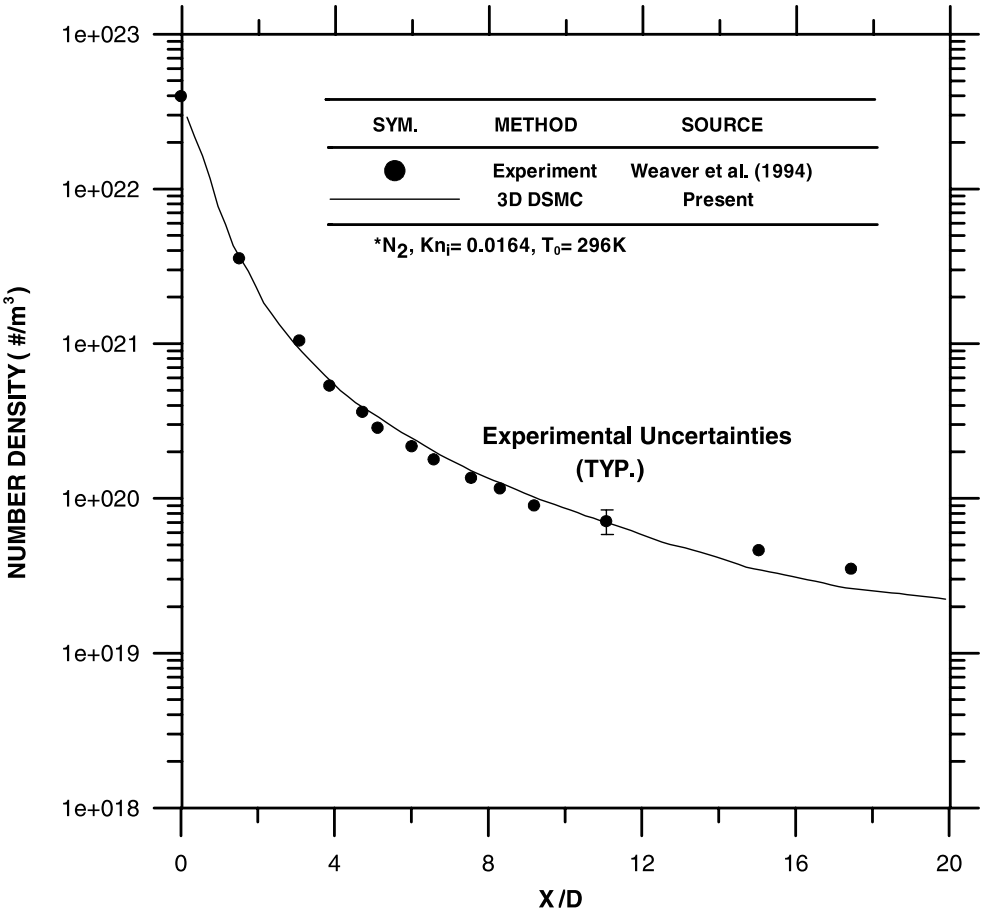

Fig. 9. Number density distribution along centerline of orifice expanding flow.

represent the numerical simulation and experiments conducted by Dogra et al. [32]. Total 121,539 tetrahedral cells, as illustrated in Fig. 11, and two million particles are used for the simulation. The simulation time-step is 1.0×10^{-7} s. Ratio of cell size to the local mean free path in the free stream is roughly unity, where the flow properties varies very little. The domain decomposition using JOSTLE for 10 processors, estimated by the results of approximately 25,000 particles, is shown in Fig. 12. Parallel efficiency using 10 processors is 71%.

3.2.1.2. Normalized number density contour. Results of the normalized number density (n/n_∞) are presented in Fig. 13. As shown in Fig. 13, a bow shock stands off at some distance away from the sphere. On one hand, the flow is highly compressed across the nearly normal shock along the stagnation line, where density increases tremendously. On the other hand, the flow is slightly compressed across the oblique shock away from the sphere and then is slightly expanded further downstream. A relatively rarefied region (wake) is formed behind the sphere. A maximum value of 4.81 of the number density ratio is observed at the stagnation point while a minimum value of 0.05

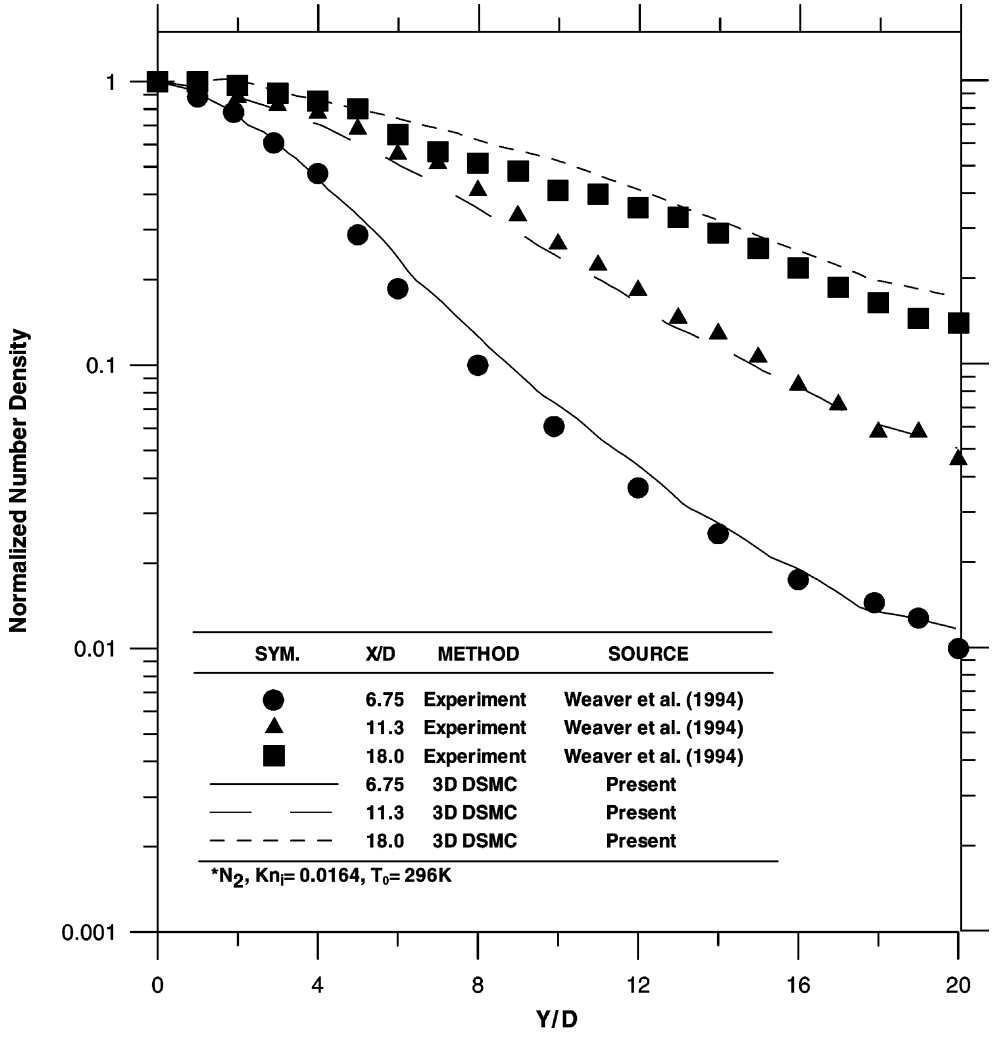

Fig. 10. Normalized radial number density distribution of orifice expanding flow at $X/D = 6.74$, 11.3 and 18, respectively.

is observed behind the sphere. Similar physical trends can also be identified from the temperature contour and Mach number contours, although they are not presented in the paper.

3.2.1.3. Properties along the stagnation line. Fig. 14 shows the results of normalized density, and normalized translational (T_t/T_∞), rotational temperature (T_r/T_∞) from $X/D = -0.6$ to stagnation point along the stagnation line. Previous data of Dogra et al. [32] using axisymmetric DSMC code are also included in this figure. First, the density profile shows that the shock wave is very diffuse, and for the density ratio, the present data agree well with the simulated data of Dogra et al. [32] in front of sphere. The maximum density ratio (4.8) of the present study is, however, smaller than

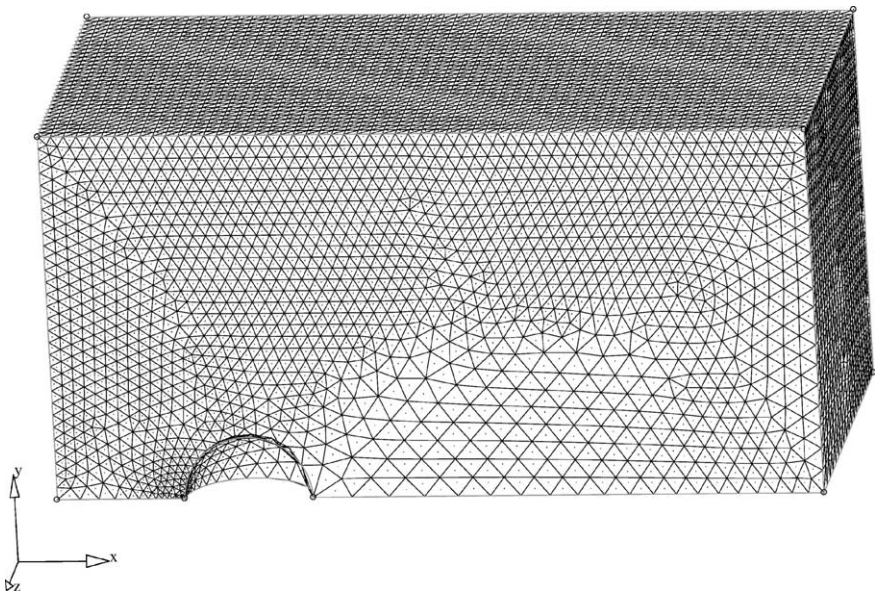

Fig. 11. Surface mesh distribution of hypersonic flow over a sphere.

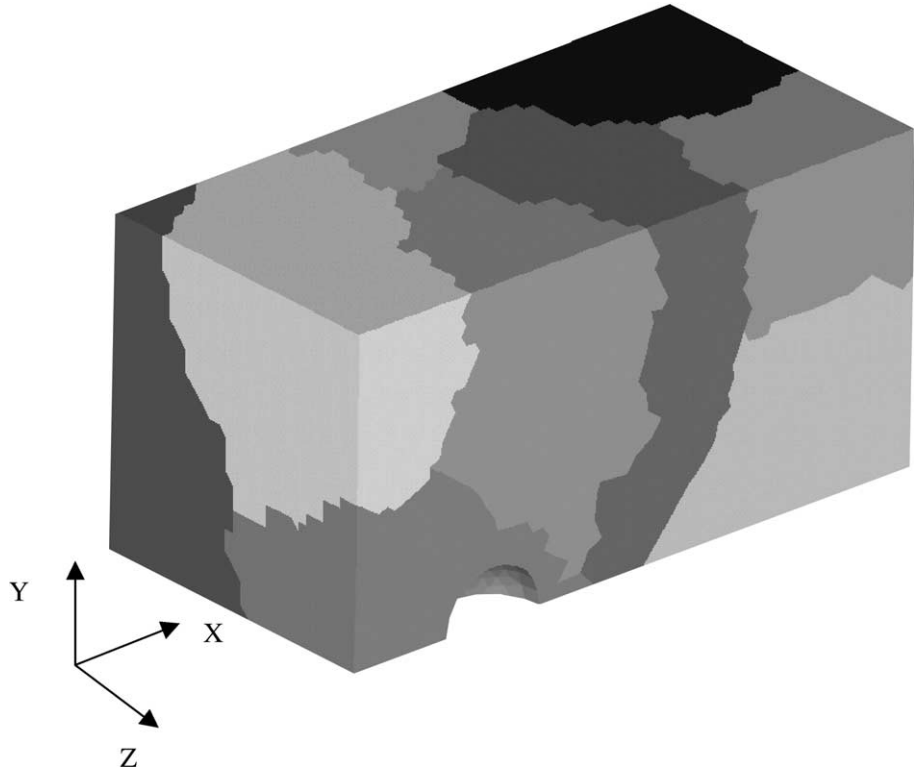

Fig. 12. Surface domain decomposition of hypersonic flow over a sphere for 10 CPUs.

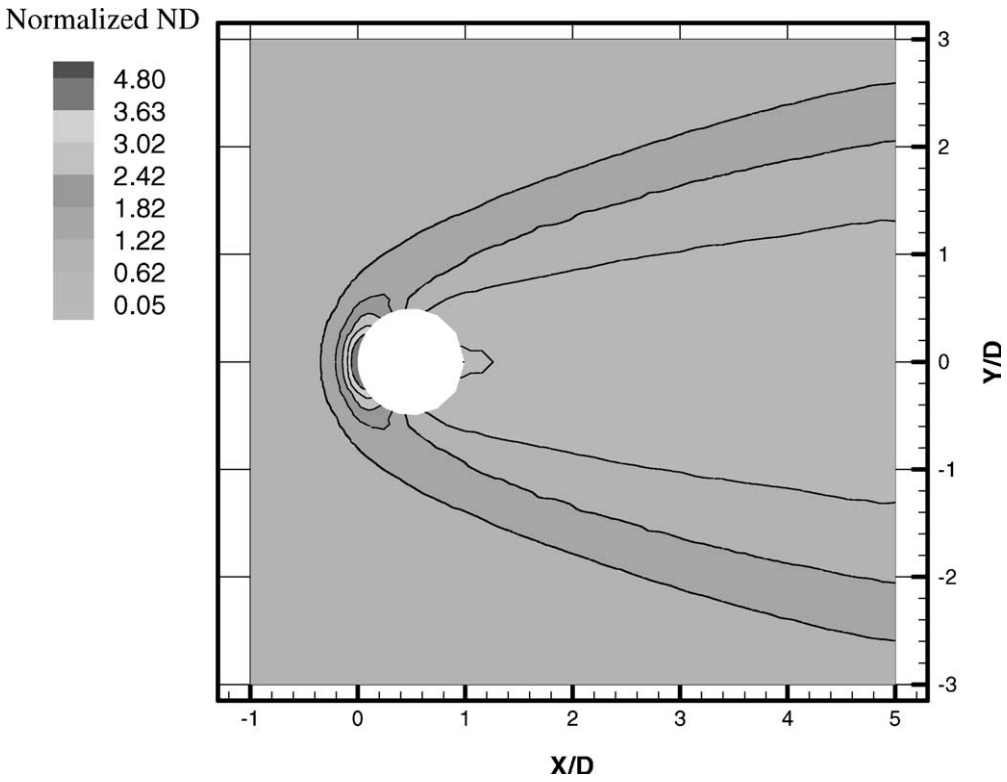

Fig. 13. Normalized number density contour of hypersonic flow over sphere on $Z/D = 0$ (air, $Kn = 0.078$, $Ma_\infty = 11.25$, $T_\infty = 22.8$ K, $T_w = 600$ K).

that (5.4) of Dogra et al. [32]. The reason for this discrepancy is not clear, although the data near axis using axisymmetric DSMC method is often questionable. Second, there is a large amount of thermal non-equilibrium in the shock-wave and shock-layer regions, as indicated by the fact that the rotational temperature deviates dramatically from the translational temperature. In brief summary, the current simulation data along the centerline compare reasonably well with those of Dogra et al. [32].

3.2.1.4. Surface properties and sphere drag. Pressure and skin-friction coefficients along the sphere surface along with those simulation data using axisymmetric DSMC of Dogra et al. [32] are presented in Fig. 15(a) and (b), respectively. The results are shown as a function of the circumferential angle θ, which is measured counterclockwise from the front stagnation point. The pressure and skin-friction coefficients of the present simulation and the axisymmetric DSMC data of Dogra et al. [32] are both well below the free-molecular limit. The results of the surface-pressure and skin-friction coefficients are in good agreement with those of Dogra et al. [32]. Furthermore, the sphere drag C_D of the present simulation (1.465) by integrating the surface drag is within 1% of the experimental data (1.481) of Dogra et al. [32], which is surprisingly excellent.

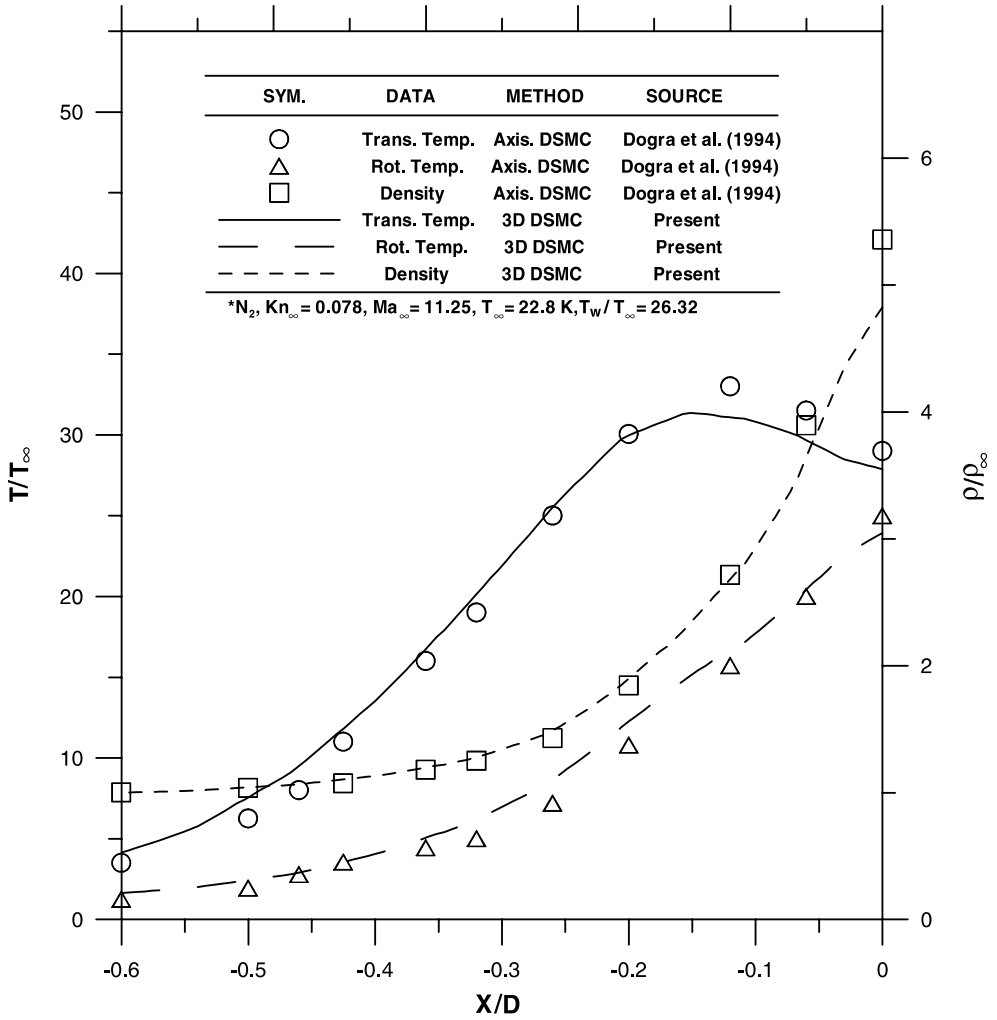

Fig. 14. Flowfield properties along stagnation streamline.

3.2.2. Flow field of the spiral-groove drag pump

Spiral-groove drag pumps, which have been extensively used in industries requiring high and clean vacuum, have a relatively excellent performance for a high inlet pressure. Thus, it is important to fully understand the flow field to improve the pumping performance. In the current study, flow field in the transitional regime in the grooves is simulated with using the three-dimensional parallel DSMC method using unstructured mesh. The results are then compared with the experimental and the DSMC data of Nanbu et al. [33].

3.2.2.1. Flow and simulation conditions. The spiral grooves on a rotor are cut in the axial direction and expanded in the circumferential direction. The groove and ridge are on x–y plane surface as

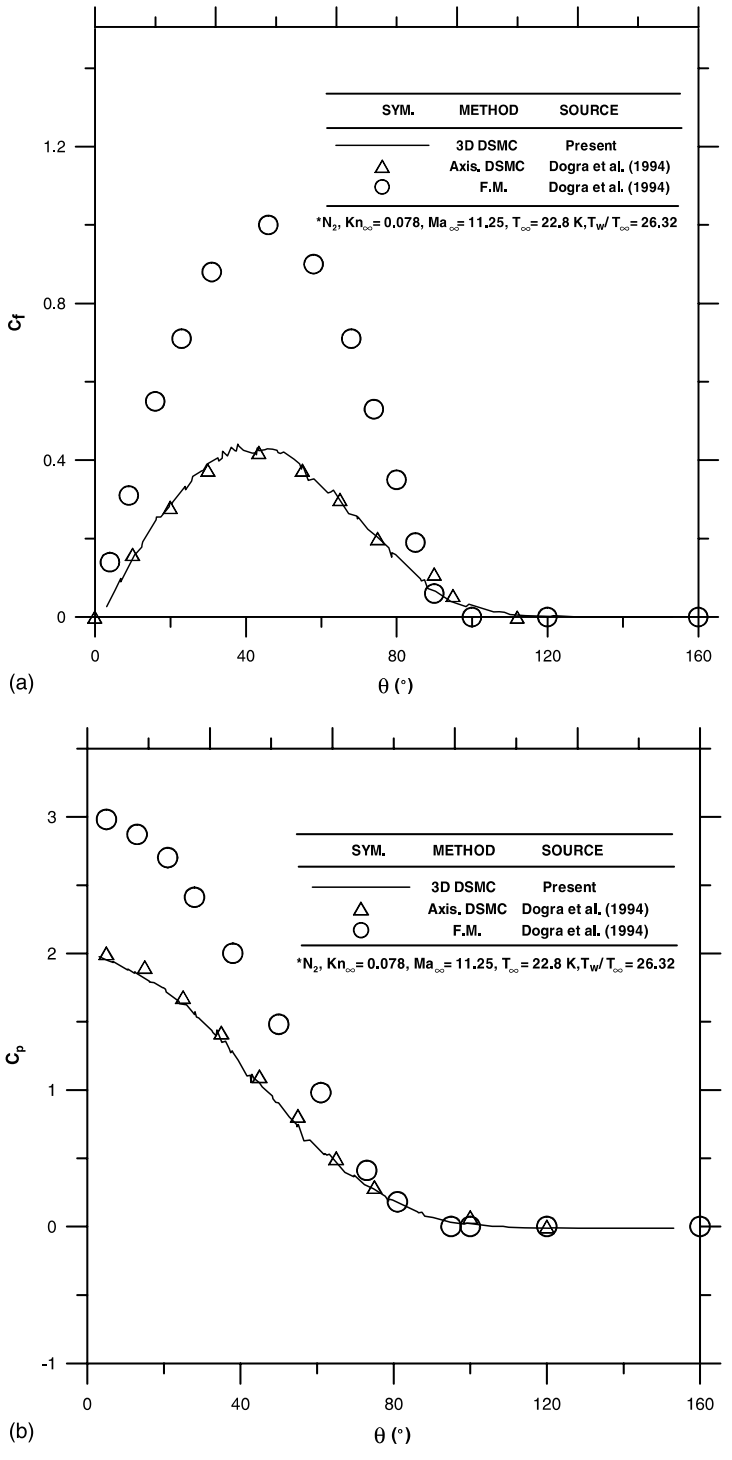

Fig. 15. Surface quantities: (a) pressure coefficient; (b) skin-fiction coefficient.

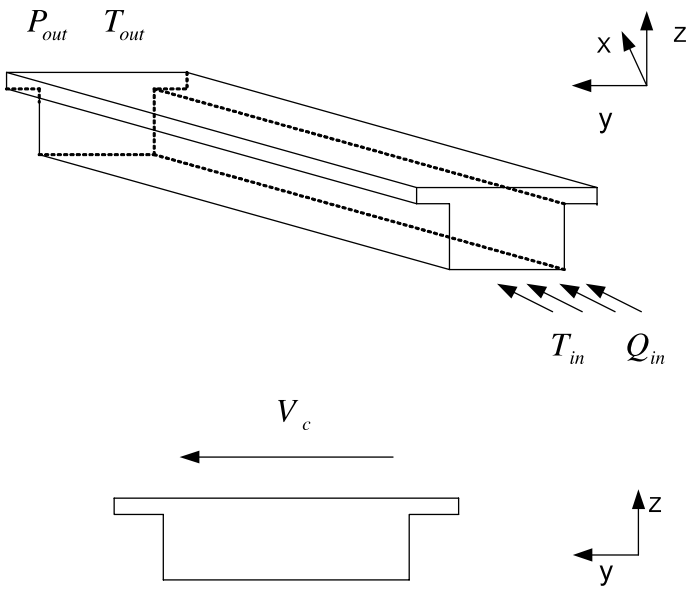

Fig. 16. Sketch of expanded spiral groove (N_2, $P_e = 13$ Pa, $T_i = T_e = T_w = 296$ K).

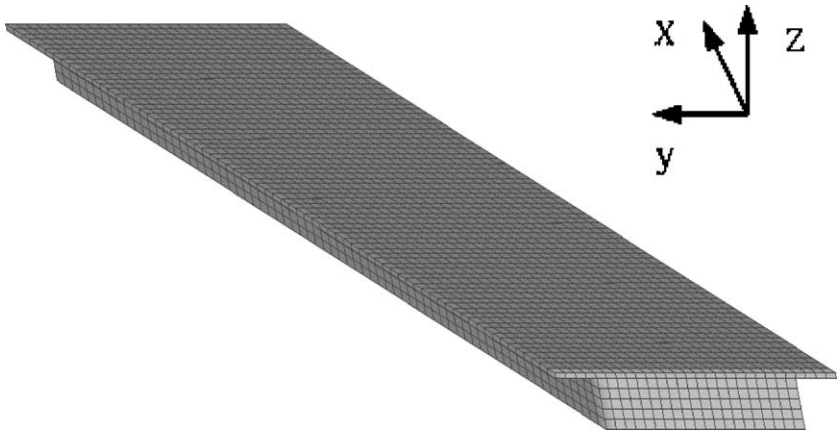

Fig. 17. Surface mesh distribution for an expanded spiral drag pump channel.

shown Fig. 16. By fixing the coordinate system on the rotor, the casing runs along the y-direction with a velocity, V_c. Related flow conditions are listed as follows: VHS nitrogen gas, outlet pressure $P_e = 13$ Pa, temperatures $T_i = T_e = T_w = 296$ K (subscript i, e, w represent inlet, outlet and wall, respectively) and inlet pressure is obtained through iteration. The corresponding outlet Knudsen number Kn_e is 0.112. These flow conditions represent the experiments conducted by Nanbu et al. [33]. Total 11,520 hexahedral cells, as illustrated in Fig. 17. 33,000 particles in total are used for

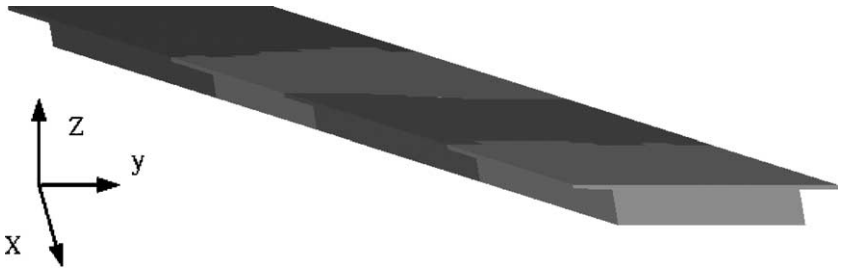

Fig. 18. Typical surface domain decomposition of an expanded spiral drag pump channel ($P_i = 10.4$ Pa, $K_i = 0.166$, $P_e = 13$ Pa, $K_e = 0.112$).

the simulation. Simulation time-step is 2.5×10^{-5} s. Ratio of cell the size to the local mean free path near the outlet port is roughly unity, which is definitely less than unity for the cells near inlet port for the uniform cell used in the current study. The domain decomposition using JOSTLE for four processors, estimated by the results of approximately 3300 particles, is shown in Fig. 18. Resulting parallel efficiency using four processors is about 90% in this case due to the iterative nature of the boundary conditions in the internal flows.

3.2.2.2. Iteration of inlet pressure. To be consistent with the experimental conditions, throughput (or equivalently mass flux) and outlet pressure are specified as well as the inlet and outlet temperatures. Throughput is defined as $Q = P_i S_i = P_e S_e$, where S_i and S_e are the inlet and outlet pumping speed, respectively. Assuming that the flow is in thermal equilibrium outside the inlet and outlet ports, the flux from outlet can be determined based on the kinetic theory. But inlet flux cannot be directly found because of unknown P_i and S_i. Therefore, iteration of inlet pressure, which is similar to, but different in the details from, that of Chang et al. [2], is implemented for a constant throughput Q. The steps of inlet pressure iteration are listed as follows: (1) Guess initial value for P_i, so the inlet flux can be found by using the definition of throughput. (2) Perform the DSMC computation as usual, and count the sums of incoming and outgoing particle number at the inlet (denoted as $N_{i,in}$ and $N_{i,out}$, respectively) during the selected number of time steps (e.g., the last 100 out of 2000 time steps). (3) Calculate the inlet pressure of the next iteration, and the influx at the inlet can be determined by using:

$$U_i = \frac{N_{i,in} - N_{i,out}}{A_i \times n_i \times \text{time}} \tag{10a}$$

$$S_i = U_i A_i \tag{10b}$$

$$P_i = \frac{Q}{S_i} \tag{10c}$$

where U_i is the uniform velocity from inlet and A_i is inlet area and n_i is the inlet number density in the current time step.

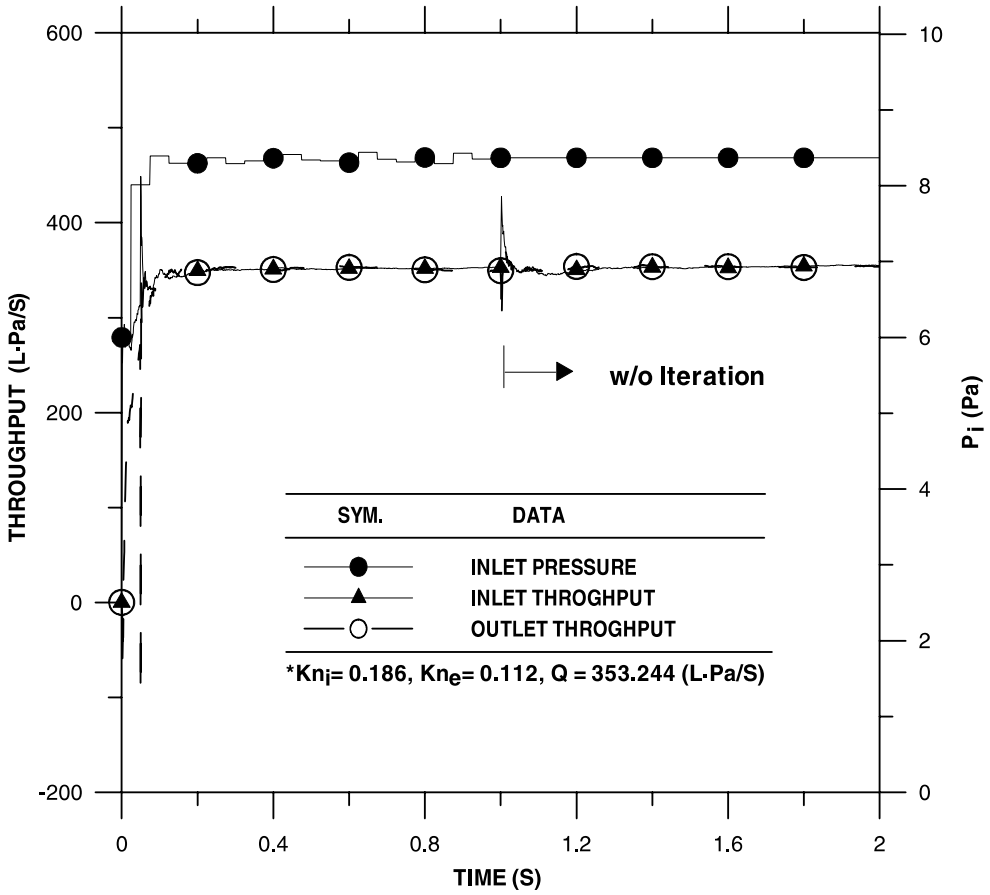

Fig. 19. Time history of inlet pressure and inlet/exit throughput during simulation.

Repeat steps (2) and (3) until the inlet pressure reaches a steady-state value. At the beginning of the simulation, the flow is developed by the initial guess inlet pressure for 1000 time steps. After that, the flow is re-developed with the iterated inlet pressure for several thousands of time steps. When inlet pressure and throughput (inlet and outlet) reaches a stable value, the iteration of inlet boundary conditions stops and further DSMC computation goes on using the final boundary condition at the inlet.

3.2.2.3. Time history of inlet pressure and inlet/outlet throughput. Fig. 19 illustrates the time history of inlet pressure and inlet/outlet throughput for a typical case ($P_e = 13$ Pa, $Q = 353.3$ L Pa/S). Typical initial inlet pressure is guessed as 1 Pa at first. Using such inlet condition, the simulation is running for 1000 time steps, assuming that the flow is fully developed. Inlet pressure iteration then starts each 2000 time steps. Through four or five iterations, the values the inlet/outlet throughputs computed from the sampled data are very close to the specified throughput (353.3 L Pa/S). After

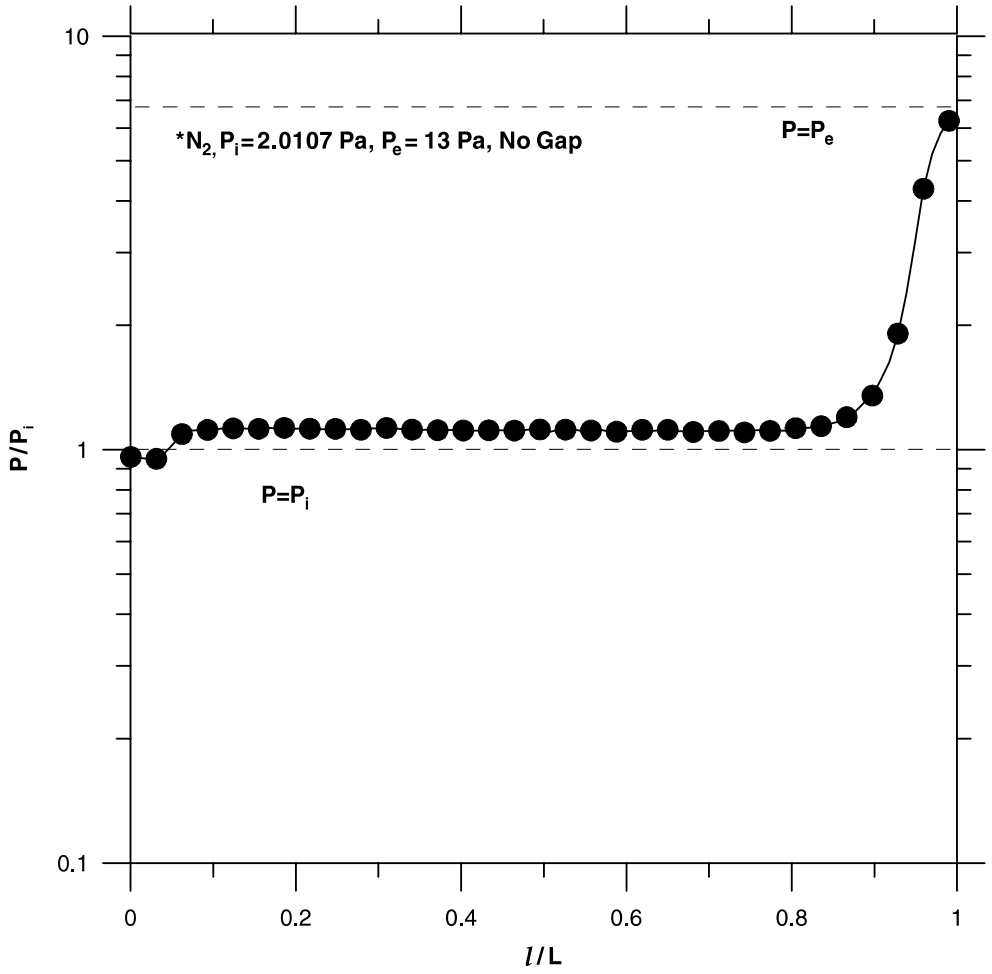

Fig. 20. Pressure distribution along centerline of expanded spiral-groove ($Kn_i = 0.774$, $Kn_e = 0.112$, $Q = 85.6548$ L Pa/S).

40,000 time steps, the inlet pressure is assumed to be steady, and the simulation stops iterating the inlet conditions. At that time, the cell properties are all reset. The inlet pressure remains fixed, but the inlet/outlet throughputs start to vary, but soon approach to the assumed throughput again. The proposed method of updating the inlet conditions has been proved to be very efficient from the above results.

3.2.2.4. Pressure distribution along centerline. Fig. 20 presents the typical result of pressure distribution along the centerline ($P_e = 13$ Pa, $Q = 85.7$ L Pa/S) for $Kn_i = 0.774$. Along the centerline, the normalized cell pressure increases from 0.95 ($x/L = 0$) to 6.3 ($x/L = 1.0$). Centerline pressure varies slightly along the channel till $x/L = 0.85$, while it increases rapidly to the specified outlet

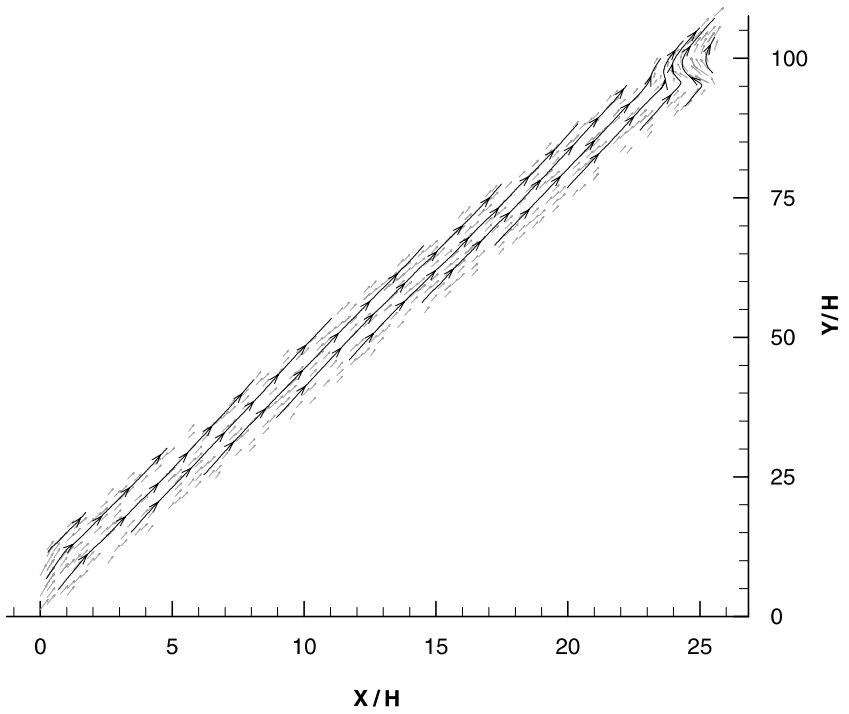

Fig. 21. Velocity vector plot (streamline) of an expanded spiral-groove at $Z/H = 1$ (N_2, $P_e = 13$ Pa, $Kn_i = 0.774$, $Kn_e = 0.112$, $Q = 85.6548$ L Pa/S).

pressure within a very short distance near the end of the channel. Because of the high-pressure gradient in this regime, backflow appears (Fig. 21) and thus molecular dragging effect caused by the moving case becomes weaker in the outlet regime.

3.2.2.5. Pumping performance. Pumping performance for the spiral-groove channel along with the experimental data and previous DSMC data [33] is shown in Fig. 22. Results show that the higher throughput is specified, the smaller pressure difference (i.e., larger inlet pressure) is obtained. Current simulation predicts lower pressure difference (the same throughput) with the gap opened as compared with that of Nanbu et al. [33]. The discrepancy is unknown and required further study. It, however, predicts much higher pressure difference with the gap closed. This is reasonable since the gap between stationary always acts as a leaking source for pumping motion. In addition, the experimental throughput is larger than the simulation data for both the present study and Nanbu et al. [33] with higher inlet pressure (lower pressure difference). Possible reasons for this deviation could be: (1) the simplified computational domain (rectangular duct, rather than real spiral-groove), (2) neglected Coriolis force due to rotating the casing of the pump and (3) the surface properties of the channel wall and the rotating case. Further consideration of the above three should help to resolve the deviation between the simulation and experiments and work in this direction is currently in progress.

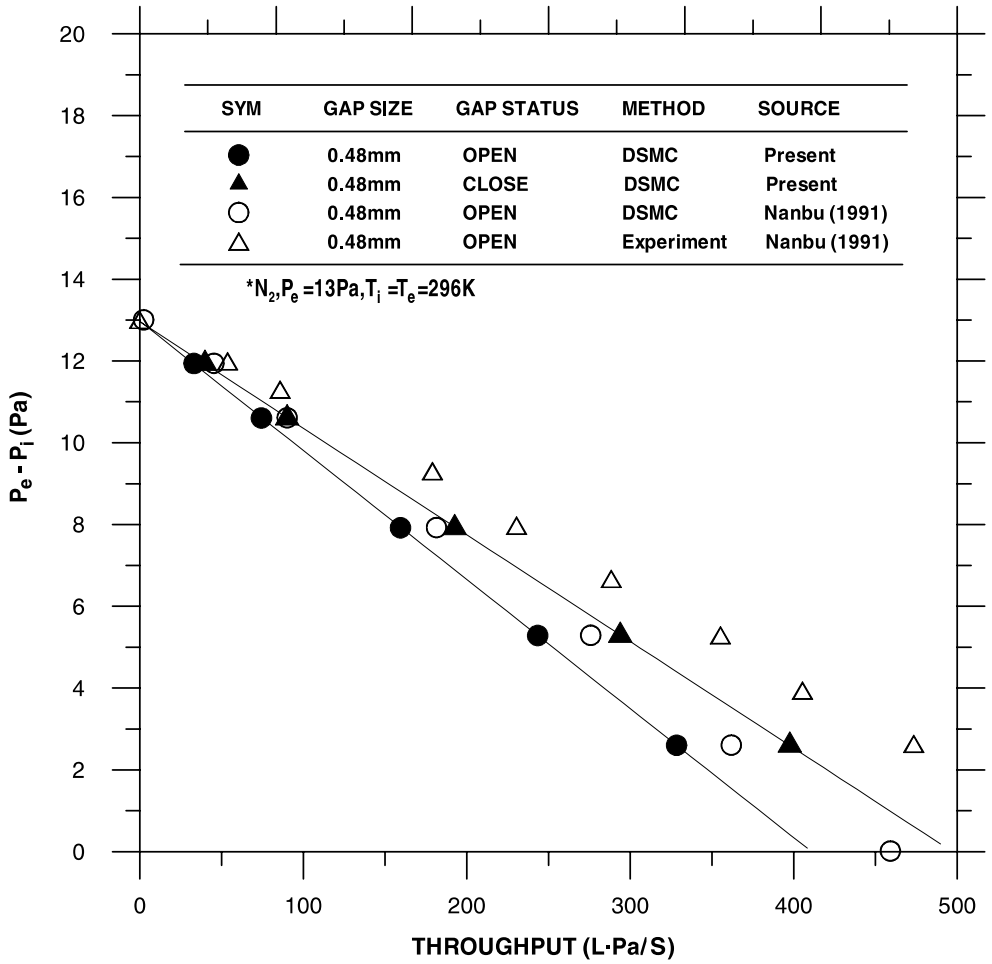

Fig. 22. Comparison of pumping characteristics for spiral-groove pump.

4. Conclusions

This investigation has successfully developed a three-dimensional parallel DSMC code using proposed particle ray-tracing technique in unstructured mesh and static graph partitioning for domain decomposition. Proposed particle ray-tracing technique and parallel implementation of DSMC in unstructured mesh are verified by simulating a supersonic corner flow and a sonic orifice expanding flow, respectively. Simulated data are then compared with available numerical or experimental data. Finally, completed three-dimensional parallel DSMC code is applied to compute typical external (hypersonic flow over a sphere) and internal (flow field of a spiral drag pump) flows, respectively. Results are then compared favorably with available experimental data.

Although the parallel performance has not been discussed in the current study, this study raises the question of load balancing among the processors during DSMC simulation. Due to the particle nature of DSMC, computational load is not balanced by static domain decomposition,

especially during the development of the transient period. Thus, further study to implement dynamic domain decomposition technique in parallel DSMC to improve the parallel performance is worthwhile and is currently in progress.

References

[1] Lee YK, Lee JW. Direct simulation of compression characteristics for a simple drag pump model. Vacuum 1996;47:807–9.
[2] Chang YW, Jou RY, Cheng HP. Pumping performance analyses of a turbo booster pump by direct simulation Monte Carlo method. Vacuum 2001;60:291–8.
[3] Plimpton S, Bartel T. Parallel particle simulation of low-density fluid flows. US Department of Energy Report no. DE94-007858, 1993.
[4] Yang X, Yang JM. Micromachined membrane particle filters. Sensors Actuators 1999:184–91.
[5] Piekos ES, Breuer KS. Numerical modeling of micromechanical devices using the direct simulation Monte Carlo method. Trans ASME J Fluids Engrs 1996;118:464–9.
[6] Nance RP, Hash DB. Role of boundary conditions in Monte Carlo simulation of microelectromechanical systems. J Thermophys Heat Transfer 1998;12:447–9 (technical notes).
[7] Wu JS, Tseng KC. Analysis of micro-scale gas flows with pressure boundaries using direct simulation Monte Carlo Method Computer Fluids 2001;30:711–35.
[8] Bird GA. Molecular gas dynamics and the direct simulation of gas flows. New York: Oxford University Press; 1994.
[9] Bird GA. Molecular gas dynamics. England: Clarendon Press; 1976.
[10] Nanbu K. Theoretical basis on the direct Monte Carlo method. In: Boffi V, Cercignani C, editors. Rarefied gas dynamics, vol. 1. Stuttgart: Teubner; 1986.
[11] Wagner W. A convergence proof for Bird's direct simulation Monte Carlo method for the Boltzmann equation. J Stat Phys 1992;66(3/4):1011–44.
[12] Wu JS, Tseng KC, Kuo CH. The direct simulation Monte Carlo method using unstructured adaptive mesh and its application. Int J Numer Meth Fluids 2002;38(4):351–75.
[13] Boyd ID, Jafry Y, Beukel JW. Particle simulation of helium micro-thruster flows. J Spacecraft Rockets 1996;1: 271–81.
[14] Kannenberg KC. Computational method for the direct simulation Monte Carlo technique with application to plume impingement. PhD Thesis, Cornell University, Ithaca, NY, 1998.
[15] Furlani, TR, Lordi JA. Implementation of the direct simulation Monte Carlo method for an exhaust plume flowfield in a parallel computing environment. AIAA Paper no. 88-2736, 1988.
[16] Matsumoto Y, Tokumasu T. Parallel computing of diatomic molecular rarefied gas flows. Parallel Comput 1997;23:1249–60.
[17] Nance RP, Wilmoth RG, Moon B, Hassan HA, Saltz J. Parallel solution of three-dimensional flow over a finite flat plate. AIAA Paper no. 94-0219, 1994.
[18] Ota M, Tanaka T. On speedup of parallel processing using domain decomposition technique for direct simulation Monte Carlo method. JSME (B) 1991;57(540):2696–700.
[19] Ota M, Taniguchi H, Aritomi M. A parallel processings for direct simulation Monte Carlo method. JSME (B) 1995;61(582):496–502.
[20] Dietrich S, Boyd I. Scalar and parallel optimized implementation of the direct simulation Monte Carlo method. J Comp Phys 1996;126:328–42.
[21] LeBeau GJ. A parallel implementation of the direct simulation Monte Carlo method. Comp Meth Appl Mech Engrs 1999;174(3–4):319–37.
[22] Robinson CD. Particle simulations on parallel computers with dynamic load balancing. PhD Thesis, Imperial College of Science, Technology and Medicine, UK, 1998.
[23] Barnard ST, Simin HD. A fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems. Concurr Practice Exp 1994;6(2):101–17.

[24] Hendrickson B, Leland R. The Chaco Users Guide: Version 2.0. Technical Report SAND (1994) 94-2692, Sandia National Laboratories, USA.
[25] Karypis G, Kumar V. Metis: unstructured graph partitioning and sparse matrix ordering, Version 2.0. User Manual. Minneapolis MN55455, Computer Science Department, University of Minnesota, USA, 1995.
[26] Kernighan BW, Lin S. An efficient heuristic procedure for partitioning graphs. Bell Syst Tech J 1970;49:291–308.
[27] Vanderstraeten D, Farhat C, Chen PS, Keunings R, Zone O. A retrofit based methodology for the fast generation and optimization of large-scale mesh partitions: beyond the minimum interface size criterion. Comp Meth Appl Mech Engrs 1996;133:25–45.
[28] Vanderstraeten D, Keunings R. Optimized partitioning of unstructured finite element meshes. Int J Numer Meth Engrs 1995;38(3):433–50.
[29] Walshaw C, Cross M, Everett MG, Johnson S, McManus K. Partitioning and mapping of unstructured meshes to parallel machine topologies. In: Ferreira A, Rolim J, editors. Proc Irregular 95: Parallel Algorithms for Irregularly Structured Problems, vol. 980, LNCS. Springer; 1995. p. 121–6.
[30] *Hypermesh*, Version 2.0, Altair Computing, Inc., 1757 Maplelawn, USA.
[31] Weaver DP, Muntz EP, Campbell DH. Direct simulation Monte Carlo calculations of compared with sonic orifice expansion flows of argon and N_2. In: Progress in astronautics and aeronautics, vol. 158. Washington DC, USA: AIAA; 1994. p. 98–108.
[32] Dogra VK, Moss JN, Wilmoth RG. Hypersonic rarefied flow past sphere including wake structure. J Spacecraft Rockets 1994;31(5):713–8.
[33] Nanbu K, Kubota H, Igarashi S. Application of DSMC to the design of spiral grooves on a rotor of turbomolecular pumps. Trans Jpn Soc Mech Engrs (B) 1991;57:172–7.
[34] Wu JS, Tseng KC, Yang TJ. Parallel implementation of the direct simulation Monte Carlo method using unstructured mesh and its application. Int J Computat Fluid Dynam, submitted for publication.