AIAA Journal
Vol. 34, No. 7, July 1996

Parallelization of Direct Simulation Monte Carlo Method Combined with Monotonic Lagrangian Grid

C. K. Oh* and R. S. Sinkovits[†]
U.S. Naval Research Laboratory, Washington, D.C. 20375
B. Z. Cybyk[‡]
U.S. Air Force Wright Laboratory, Dayton, Ohio 45433
and
E. S. Oran[§] and J. P. Boris[¶]
U.S. Naval Research Laboratory, Washington, D.C. 20375

The monotonic Lagrangian grid (MLG) and the direct simulation Monte Carlo (DSMC) methodology were combined on the Thinking Machines CM-5 to create a fast DSMC-MLG code with automatic grid adaptation based on local number densities. The MLG is a data structure in which particles that are close in physical space are also close in computer memory. Using the MLG data structure, physical space is divided into a number of templates (cells), each containing the same number of particles. An MLG-regularization method, stochastic grid restructuring, is implemented to minimize the occurrence of highly skewed cells. Parallelization of the DSMC-MLG is achieved by two different mapping techniques. First, simulated particles are mapped onto the parallel processors for the particle-oriented processes, such as convection, boundary interactions, and MLG sorting. Second, particle templates are mapped onto the processors for computing the macroscopic quantities (i.e., pressure, velocity, density, and temperature) and statistical sampling. In both levels of mapping, the code logic focuses on the structured and fast communications on the CM-5 architecture. The computing time required by the parallel DSMC-MLG code was significantly decreased compared with other parallel efforts and its parallel efficiency on 512 processors achieved approximately 80% for simulation involving one-half million particles.

Introduction

In the high Knudsen number (Kn) flow regime, where the mean free path of the gas molecules becomes comparable to or larger than the characteristic length scale of the system, the Navier–Stokes equations break down and alternative techniques are needed to describe the flow. One of the well-established methods for describing the regime $Kn > 0.1$ (length scale based on local flow gradient)[1] is the direct simulation Monte Carlo (DSMC) method,[1] a direct particle simulation technique based on kinetic theory. The method is based on the same basic approximations from which the Boltzmann equation is derived. However, its regime of validity also extends to systems involving ternary chemical reactions, because DSMC does not rely on the principle of the invariance of inverse collisions, a necessary condition for the validity of the Boltzmann equation.[1] The fundamental idea of DSMC is to track a very large number of test particles, representing one or more actual gas molecules of physically correct molecular size, through representative collisions and boundary interactions, and then to modify their positions and velocities appropriately in time. The core of the DSMC algorithm consists of four primary processes: convection, sorting and indexing, collision, and flowfield sampling. The primary approximation of DSMC is to decouple the molecular motions and intermolecular collisions based on the scale of the mean collision time. Particle motions are modeled deterministically, whereas collisions are treated statistically.

There is a wide range of applications that can be effectively addressed by DSMC methods. In the last few years, DSMC has been extensively used for rarefied aerothermodynamic problems characterized by high temperatures and thermochemical nonequilibrium at low gas densities. These problems include spacecraft aerobraking,[2–4] plume interactions,[5–7] low-density jets,[8,9] and low-density nozzle flows.[10,11] The method is also applicable to many types of microelectromechanical system (MEMS) flows, such as flows in microchannels, microgenerators,[12–16] and materials processing[17] because DSMC is the general numerical technique for any high Kn flows. A detailed understanding of MEMS flow could lead to the development of efficient interactive flow control system for practical flowfields.[18,19]

A conventional DSMC approach involves tracking the trajectories of simulated particles on a fixed spatial grid that defines the spatial cells. The cells are used to group together neighboring particles and thus to select possible collision pairs randomly. A time-consuming part of the simulation is tracking particles in physical space and indexing particles in computer memory and then determining which particles are in which spatial cells for selecting collision pairs.

In an effort to automatically adapt the grid to account for changing flow properties, a new technique based on the monotonic Lagrangian grid[20] (MLG) has been developed by Cybyk[21] and Cybyk et al.[22] The MLG is a general data structure in which the nodes can represent particles, which can be, for example, atoms, molecules, fluid elements, or droplets. Since the MLG maintains a direct correspondence between the indexing and the position of the particle, it is well suited for problems involving substantial variations in local density. The MLG has been used previously in DSMC[21,22] and molecular dynamics[23–26] calculations.

The large-scale simulations of flows in complex geometries at near-continuum (low) values of Kn using conventional DSMC are often prohibitive because of very high computing requirements dictated by higher collision rates. Therefore, further reductions in computational time are necessary. Such cost reduction is possible through the development of efficient parallel DSMC algorithms and through advances in parallel computer architecture. Recent advances in computer hardware for parallel processing do offer the potential for significant reductions in computational times, but parallelization

Received Sept. 20, 1995; revision received Jan. 28, 1996; accepted for publication Feb. 19, 1996. This paper is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

*National Research Council Research Fellow; currently Research Aerospace Engineer, Laboratory for Computational Physics and Fluid Dynamics, Code 6400, 4555 Overlook Avenue, SW. Member AIAA.

[†]Research Physicist, Laboratory for Computational Physics and Fluid Dynamics, Code 6400, 4555 Overlook Avenue, SW.

[‡]Aerospace Engineer, Turbine Engine Division. Senior Member AIAA.

[§]Senior Scientist for Reactive Flow Physics, Laboratory for Computational Physics and Fluid Dynamics, Code 6400, 4555 Overlook Avenue, SW. Fellow AIAA.

[¶]Chief Scientist and Director, Laboratory for Computational Physics and Fluid Dynamics, Code 6400, 4555 Overlook Avenue, SW. Fellow AIAA.

of DSMC codes may be difficult to achieve using the traditional fixed spatial grid approach because the number of particles in each cell is arbitrary, therefore resulting in a number of load-balancing issues. This paper describes a DSMC-MLG algorithm parallelized on the multiprocessor CM-5. We examine the efficiency of the calculation procedure as function of the number of simulated particles and processors and then compare the performance of the parallel code to other parallel studies. Finally, general issues arising from parallelization on various types of massively parallel computers are discussed.

Combining DSMC and MLG

The MLG data structure requires minimal memory and uses monotone arrays for indexing geometric positions. The two-dimensional location of each node in the MLG data structure is determined by the set of constraints

$$\begin{aligned} x(i,j) &\le x(i+1,j) \quad &\text{for} \quad 1 \le i \le N_x - 1, \quad &\text{all } j \\ y(i,j) &\le y(i,j+1) \quad &\text{for} \quad 1 \le j \le N_y - 1, \quad &\text{all } i \end{aligned} \tag{1}$$

where the set (N_x, N_y) defines the data structure with N_x and N_y particles in the x and y directions, respectively, and (i, j) represents the grid indices. Although Eq. (1) is written for a two-dimensional system, the MLG data structure can be generalized to any number of dimensions. Initially particle positions are sorted in memory to achieve this MLG order. When the grid becomes distorted in the course of the flow, particles are sorted again to maintain this order throughout the calculation. Once the particles are sorted into MLG order, the interactions between nearby particles are found by searching through index space rather than physical space. For example, the neighbors of a particle stored at the location (i, j) in a two-dimensional array are in the neighboring array locations $(i+1, j)$, $(i+1, j+1)$, $(i, j+1)$, etc. Therefore, particles close in physical space are close in computer memory.

The conditions defined by Eq. (1) are not sufficient to define a unique order for the particles within the data structure because of the number of coordinate inequalities,

$$2N[1 - (1/N_x) - (1/N_y)] \tag{2}$$

where the total number of particles N is less than the number of degrees of freedom of the system. There are many possible MLGs for even very small data structures. For example, a system containing as few as 25 particles has been shown to have from 10,000 to 100,000 valid 5 x 5 MLG orderings[27] that meet the requirements of Eq. (1).[20] Previous molecular dynamics applications have demonstrated that it is possible to find MLG orderings with very poor grid structures. Recently, the question of quality of an MLG and issues involved in constructing an MLG with the best properties have been addressed by Sinkovits et al.,[26] who determined that the best quality MLGs are the easiest to obtain.

Figures 1a and 1b show schematics of a space-fixed uniform-cell and a stretched-cell system used in conventional DSMC. Such a stretched-cell system could, for example, be used to obtain better resolution at boundary layers. It is possible to replace the fixed spatial grid in conventional DSMC calculations by using the MLG, as shown in Fig. 1c. Particle templates are defined by groups of particles subdivided into the nearest-neighbor blocks, with each block containing an equal number of particles. Therefore, the locations of the outermost particles themselves can be used to define the boundaries of the templates. The fact that each template has the same number of particles is important for parallelization of the DSMC-MLG code. An identical pattern of communication can be used in each template to evaluate template properties such as area, center of mass, and average particle momentum. The identification of collision pairs is also greatly simplified since the indices of a molecule automatically define the template in which the particle is located.

In contrast to conventional DSMC, each template in the DSMC-MLG contains the same number of particles throughout a given simulation. The result is that the template grid automatically adapts to account for the local and continually changing number densities in the system. Figure 2a shows the initial conditions of a calculation in which a circular high-density region is separated from the

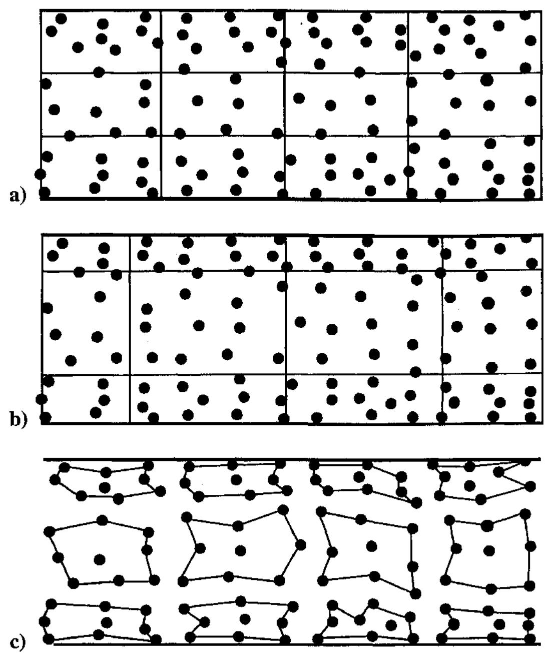

Fig. 1 a) Space-fixed cell system of conventional DSMC, b) stretched-cell system of conventional DSMC, and c) typical DSMC-MLG template shapes.

low-density regions.[22] As the calculation evolves, the high-density material expands into its surroundings. Figures 2b–2e show the time evolution of the adaptive grid system represented by lines that connect the average center-of-mass locations of the time-varying templates. The shapes and areas of the templates gradually adapt to changes in the local number densities. The result of this grid adaptation is higher accuracy for a given grid size.

Combining the MLG and the DSMC requires three primary steps: eliminating space-fixed cells in lieu of nearest-neighbor templates, replacing the conventional indexing algorithm with MLG particle tracking and sorting routines, and developing a general routine for the calculating areas of the time-varying templates. These three modifications result in a number of procedural changes to the conventional DSMC method. The new DSMC-MLG flowchart is compared with the old flowchart in Fig. 3,[21] where Δt_g is the global time increment of an unsteady calculation, Δt_s is the sampling time step for controlling unsteady sample interval, and t_L is the specified computational time step. The new procedures added are highlighted in the DSMC-MLG flowchart presented in Fig. 3b.

The first step, to eliminate space-fixed cells, requires two additional operations immediately after the zero-time state of the gas is set: 1) construction of the MLG and 2) subdivision of the MLG into a specified number of nearest-neighbor templates. Taken together, these are analogous to, and effectively replace, the grid generation required before any standard DSMC application. The current MLG implementation specifies the grid size by choosing the number of templates along each axis. Note that every molecule belongs to one and only one template. The positions of the templates relative to each other remain the same during the course of a simulation, although the particles composing the templates do vary. All templates have the same number of particles and are square in index space. This is not necessary but leads to more efficient parallelization.

The second step requires replacing the standard DSMC molecular indexing with MLG-based routines (Fig. 3). Sorting and tracking algorithms are used to resort particles into MLG order. The resorting process may change a template's physical size and shape but does not change the layout of the template in array space. It is this resorting step that automatically adapts the grid physical space to local number densities.

The third step is to evaluate template areas and template-averaged quantities. The areas are necessary because macroscopic properties generated during the sampling step are valid at the center of mass of each of the templates, which, in turn, are a function of area.

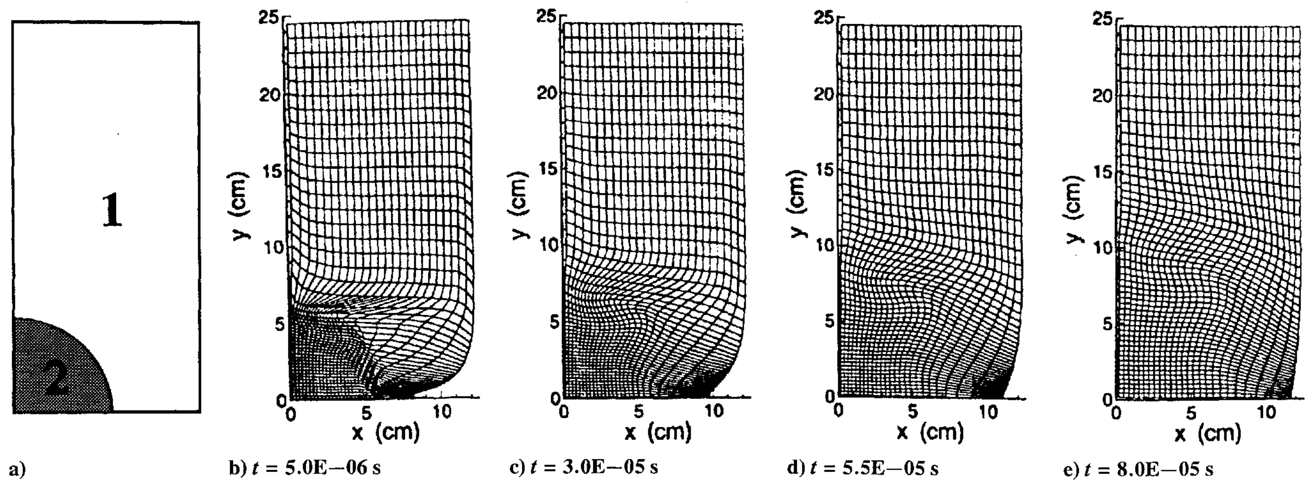

Fig. 2 a) Initial setup of circular diaphragm problem: 1) low-density gas and 2) high-density gas; and b–e) DSMC-MLG adaptive grid at four different sampling time intervals.

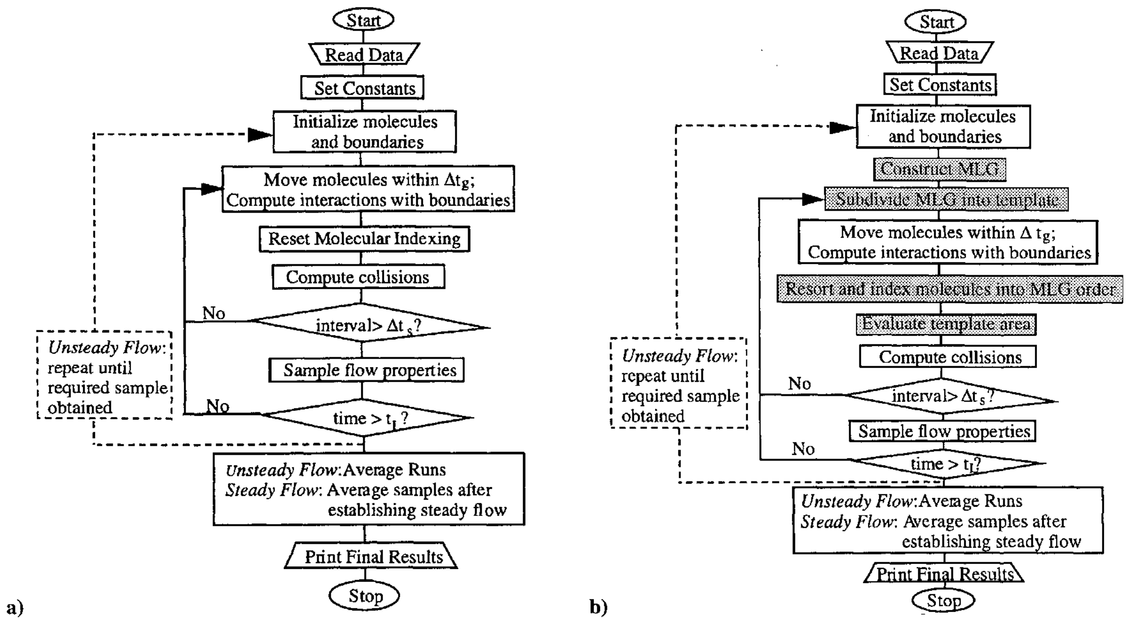
a)
Start
Read Data
Set Constants
Initialize molecules and boundaries
Move molecules within Δtg; Compute interactions with boundaries
Reset Molecular Indexing
Compute collisions
No
interval> Δts?
Sample flow properties
No
time > tf ?
Unsteady Flow: repeat until required sample obtained
Unsteady Flow: Average Runs
Steady Flow: Average samples after establishing steady flow
Print Final Results
Stop
b)
Start
Read Data
Set Constants
Initialize molecules and boundaries
Construct MLG
Subdivide MLG into template
Move molecules within Δ tg; Compute interactions with boundaries
Resort and index molecules into MLG order
Evaluate template area
Compute collisions
No
interval> Δts?
Sample flow properties
No
time > tf?
Unsteady Flow: repeat until required sample obtained
Unsteady Flow: Average Runs
Steady Flow: Average samples after establishing steady flow
Print Final Results
Stop

Fig. 3 a) Standard DSMC flowchart and b) DSMC-MLG flowchart; processes highlighted in gray are added for the combination of DSMC and MLG.

In addition, relations used in the collision modeling process also depend on instantaneous template areas. The physical domain of a given template encompasses all of its particles, as does the domain of a cell used in traditional cell-based methods. A template area, or the area of the domain containing a given template, is evaluated in a two-step process. First, the area contained within the template's outer boundary, formed by simply connecting its outermost particles, is evaluated. If we label the bounding particles $P_1, P_2, \ldots, P_{n-1}, P_n$ in counterclockwise order, the area of the template is given by

$$A = \tfrac{1}{2}[(x_1y_2 + x_2y_3 + \cdots + x_{n-1}y_n + x_ny_1) - (y_1x_2 + y_2x_3 + \cdots + y_{n-1}x_n + y_nx_1)] \tag{3}$$

where x_i and y_j are the coordinates of two different bounding particles P_i and P_j (i and j are integers from 1 to n). This step accounts for approximately 77% of the computational domain, a percentage that varies only slightly with time. Next, the portion of a template's physical domain that lies outside the template's outermost particles is computed using weighting factors based on relative areas from the previous step. These weighting factors are used to proportionally increase individual areas computed using Eq. (3) until the area summation of all templates accounts for exactly 100% of the computational domain.

One problem in using the MLG is that the grid can become very distorted in regions of high-density gradients. This can result in the nonphysical situation in which collisions between spatially close particles are ignored while interactions between more widely separated particles are considered. In addition, the irregular shapes of the highly skewed templates in the distorted region of the grid can lead to errors in the evaluation of the template areas.[22] To minimize these problems, a grid-restructuring technique, stochastic grid regularization[26] (SGR), is employed. In applying SGR, the following additional steps are taken. The particles that have been put in MLG order are randomly displaced in space, and the data are swapped until monotonicity conditions are satisfied for the perturbed node positions. Finally, data are swapped until the monotonicity conditions are satisfied for unperturbed node positions. Figure 4 shows how the poorly ordered MLG is improved by SGR. The SGR process is performed between the MLG ordering and the template-area evaluation, as shown in the flowchart in Fig. 5. The application of SGR results in a new grid in which the arrangement of the particles in index space more closely corresponds to their ordering in physical space. The DSMC simulations using the restructured grids

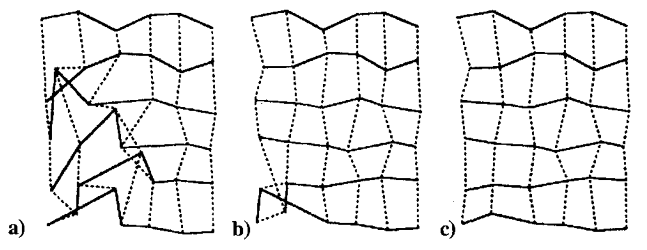

Fig. 4 Examples of three different monotonic Lagrangian grids based on the same set of particle locations. Solid and dashed lines connect x and y positions, respectively. Grid a was obtained by sorting modes into MLG order from a random order. Grids b and c are derived from grid a by one and two iterations, respectively, of SGR.

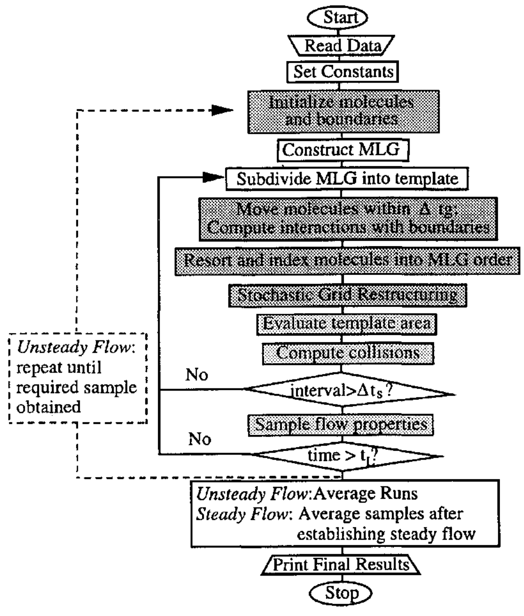
Start
Read Data
Set Constants
Initialize molecules and boundaries
Construct MLG
Subdivide MLG into template
Move molecules within Δ t_g; Compute interactions with boundaries
Resort and index molecules into MLG order
Stochastic Grid Restructuring
Evaluate template area
Compute collisions
Unsteady Flow: repeat until required sample obtained
No
interval>Δt_s?
Sample flow properties
No
time > t_l?
Unsteady Flow: Average Runs
Steady Flow: Average samples after establishing steady flow
Print Final Results
Stop

Fig. 5 DSMC-MLG flowchart including SGR process; steps highlighted in dark gray show particle-level parallelization, and light gray steps represent template-level parallelization.

are better because nonphysical collisions between widely separated particles are minimized.

Parallelization of DSMC-MLG

The parallelization technique used in this study was developed for the DSMC-MLG algorithm and implemented on the Thinking Machines 256 processor CM-5. Each processor has four vector units and the theoretical peak performance of the CM-5 is 40 Gflop/s. The memory is distributed over all of the processors and a total of 32 Gbytes is available. The CM-5 is a data parallel computer that can be used in either multiple instruction multiple data (MIMD) or single instruction multiple data (SIMD) mode and provides high performance for both message passing and data parallelism. For this application, we found it most convenient to use the SIMD mode. The programming language used is CM-Fortran, which is designed for parallel array processing. CM-Fortran also provides many optimized intrinsic functions, such as those for manipulating arrays and a fast parallel random number generator.

There are two major factors hampering the parallelization of a DSMC code. First, the algorithm can require a great deal of interprocessor communication, which is relatively slow. Second, load balancing among the processors can be poor so that many processors can be idle while just a few are active. Therefore, an efficient and structured way to handle interprocessor communications is essential.

The parallelization technique that we have developed, shown schematically in Fig. 6, addresses both questions. First, we map the data to processors at two levels: particles are mapped onto the processors, and templates are mapped separately onto processors.

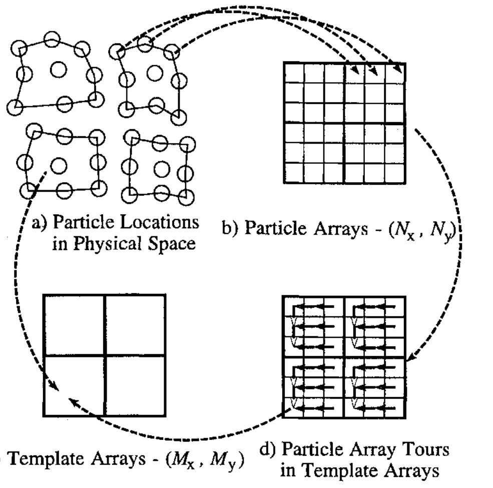
a) Particle Locations in Physical Space
b) Particle Arrays - (N_x, N_y)
c) Template Arrays - (M_x, M_y)
d) Particle Array Tours in Template Arrays

Fig. 6 Procedure for mapping particles and templates onto processors.

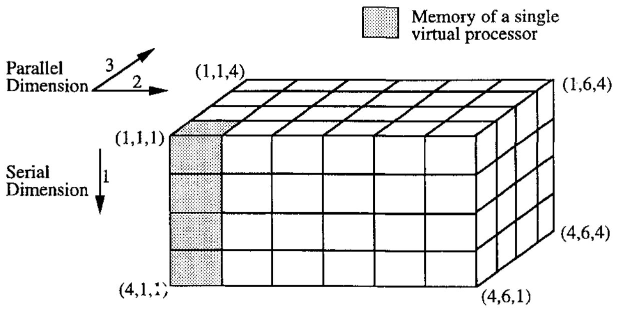

Fig. 7 $4 \times 6 \times 4$ array with one serial and two parallel dimensions.

We have then tried to optimize the communication between the two levels. In addition, a method to improve load balancing was developed, which also has the benefit of decreasing required computer memory. Particle data, such as positions and velocities, are stored in arrays of dimensions (L_x, L_y), where $L = L_x \times L_y$ is the total number of particles. This particle-level mapping is shown in Figs. 6a and 6b. Template data, including template areas, centers of mass, and average particle momenta, are stored in arrays of size $M_x \times M_y$. Schematically, this template-level mapping is also shown in Figs. 6a and 6c. Although the parallelization of a DSMC-MLG code could be done at the particle level only, introducing the two-level parallelization results in significant memory and computational savings.

Besides the two different levels of mapping described earlier, we also use serial mapping in the vector units in processors. This is useful for the inherently serial operations, such as the unsteady sampling shown in Fig. 5. Figure 7 shows an array with one serial dimension and two parallel dimensions. For example, the two parallel dimensions could be used to map either particles or templates. The serial dimension could be used to store statistical averages. The array in Fig. 7 can be viewed as a matrix of local vectors, where each local vector resides in the memory of a unique virtual processor.

Another aspect of DSMC that makes it difficult to parallelize is that the CPU-intensive processes of the algorithm are different for different problems. This makes it critical that each part is optimized separately. In the following discussion, we break the processes into a number of steps: moving the particles, indexing and sorting particles, evaluating template quantities, evaluating effects of particle collisions, and performing statistical average over ensembles (sampling).

Several of the processes in the DSMC-MLG code involve communications and computations at the particle level only. The simplest of these, the initialization and convection steps, do not involve interprocessor communications and could be done directly at the particle level. The boundary interactions, which could be, for example, fully diffuse or specular reflecting, were parallelized by selecting only those particles interacting with the boundaries and then computing their velocities and positions. Although this process causes a load-balancing problem, its effect on the overall computing time was very small since it is done only once for each unsteady loop (Fig. 5).

Sorting particles into MLG order and restructuring the grid using SGR involves communications at the particle level. Parallelization of these steps may be carried out very efficiently by taking advantage of the fast regular patterns of communication. When these processes are parallelized, the MLG sorting routine is based on the bubble-sort algorithm. On all processors, the x position of each particle is compared with the x positions of its nearest neighbors simultaneously. The locations of the particles at different memory locations are swapped until the monotonicity constraint in the x direction is satisfied. Then the same sorting process is repeated in the y direction, and eventually all of the particles are sorted into an MLG-ordered data structure. When particles are sorted from a random order, a bubble sort is considered to be one of the poorest performing sorting algorithm.[28] However, in time-dependent MLG applications, where particle locations in physical space change with time, bubble sorting is ideal for use in the MLG sorting routing since the particles are never allowed to get very far out of MLG order. In DSMC computation, the time step is extremely small because of the decoupling of convection and collision processes, meaning that violation from the monotonic data structure is small. Such restructuring is discrete, reversible, and efficient; it corrects local monotonicity violations without noticeable global changes. This local bubble sorting was also efficiently used in the serial code.[21] An additional advantage of the bubble sort on the CM-5 is that the entire process can be performed using only nearest-neighbor grid communications rather than more expensive general patterns of communications.

The evaluation of template properties takes place entirely at the particle level, and the results are stored at the template level. Because all templates are identical in size in index space, this can be done efficiently using the same regular pattern of communication within each template. To carry out this process, we perform a process called a tour. The x and y positions of particles are shifted one index in the x direction in computer memory by using a CM-Fortran cshift statement, and these values are added to those at the original positions. This shifting and adding process is repeated until all of the values of the particles in the x direction in a template are summed. For example, all particles are shifted twice if a template has three particles in the x direction, as shown in Fig. 6d. At this stage, each particle array has values summed in the x direction. Exactly the same processes are then performed again in the y direction. Then the original position (lower left-hand corner in Fig. 6d) contains the final value summed over all particles, and this value is selected using a CM-Fortran forall statement. The sequence is shown in Figs. 6b, 6d, and 6c. This parallelization process may be directly extended to three dimensions by adding the shift process in the z direction.

Since DSMC is a statistical technique, it is necessary to average template data over a large number of independent ensembles. For this reason, rather than storing template properties in arrays at the particle level, it is a much more efficient use of memory to move the newly evaluated template data to the smaller arrays at the template level. Consequently, the sampling process uses the same technique used in evaluating template quantities.

The most intensive communication between the particle and template arrays takes place during the calculation of binary collision process. Calculating binary collisions at the particle level is inefficient because it leads to a situation where the majority of the processors are idle and thus to a load-balancing problem. For example, in a template containing N_t particles, only two of the particles will be actively involved in a collision at any given time. While the two virtual processors containing the colliding particles are active, the remaining $N_t - 2$ virtual processors are idle. In the parallelization technique used here, a collision pair is chosen randomly at the particle level, and then interprocessor communications are used to transfer the data for the pair from the particle arrays to the template arrays. All operations are then carried out at the template level, and the results after the collisions are then communicated back to these particle arrays. These procedures are repeated until each template satisfies the requirements set by the no time counter[1] collision method at each time step. All independent particles in each template participate in collision process. Collision partners are chosen at random and without restriction from all of the particles occupying a given template. This means that the parallel collision method preserves the statistical independence in selecting collision pairs. The results from this process agree well with the results from the traditional collision process. Although this parallel process results in additional interprocessor communications, the savings realized by avoiding the unnecessary computations at the particle level more than compensate for the cost associated with the extra communications. As a result, the remapping converts global communications between particles into the local communications between particles in each template. The heavy and light shaded steps in Fig. 5 show the particle-level and template-level parallelization, respectively.

Since template quantities, such as areas and centers of mass, do not have to be stored in the larger arrays used for particle data, the two-level scheme reduces the storage requirement for the code by a factor equal to the template size. Since the ensemble averages of template quantities may be required at a large number of time intervals to study the evolution of an unsteady flow, it is important to limit the data that must be stored for each time interval.

Validation and Tests

The Rayleigh problem has been used as a standard problem for validating DSMC codes.[29–31] The DSMC one-dimensional solutions of this problem have been validated by comparing them with both Bhatnagar–Gross–Krook and analytical solutions.[1] Variations of the Rayleigh problem were used to validate two-dimensional versions of our parallelized DSMC code. Here results of the parallelized version of DSMC-MLG are compared with those obtained by Cybyk et al.,[22] who used a serial version of DSMC-MLG to study the two-dimensional Rayleigh problem.

A schematic of the test problem is shown in Fig. 8. The size of the computational domain is $20\lambda_{ug} \times 10\lambda_{ug}$, where λ_{ug} is the mean free path of the molecules in the undisturbed gas. The domain contains 39,200 simulated particles that are arranged in a 280×140 MLG. The MLG is subdivided into 40×20 arrays of templates, each containing 7×7 particles. The flat plate in Fig. 8 is heated instantaneously and then accelerated to a constant speed through the undisturbed gas. The plate speed u_{lw} is twice the most probable molecular speed of the undisturbed gas V_{mp}, and the plate temperature T_{lw} is 1.6 times that of the undisturbed gas T_{ug}. The plate surface is modeled as a fully diffuse boundary, and the remaining boundaries are specularly reflecting.[1] The gas initially consists of a uniform distribution of hard-sphere monatomic molecules. The one-dimensional Rayleigh solution is reproduced at $x = x_{\max}/2$ at early times in the calculations, before disturbances from the left, right, and upper boundaries have had a chance to disrupt the one-dimensional nature of the flow.

In previous work,[22] results of standard DSMC and the DSMC-MLG were compared in some detail and showed better accuracy of the DSMC-MLG solution for a fixed number of particles. Here we describe results of simulations with identical physical parameters for both the scalar and parallelized versions of the DSMC-MLG code. Results were collected from an ensemble of 1000 independent

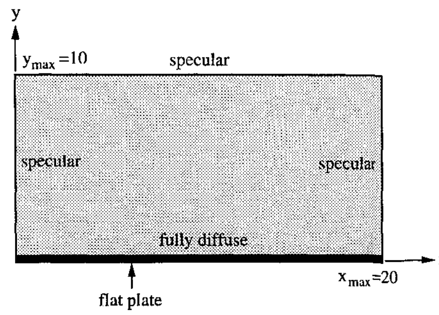

Fig. 8 Schematic of a two-dimensional Rayleigh validation problem. Boundary conditions are $u(x, y, 0) = 0$ and $u(x, 0, t) = 2V_{mp}$ for $t > 0$ and $T(x, 0, t) = 1.6T_{ug}$ for $t > 0$.

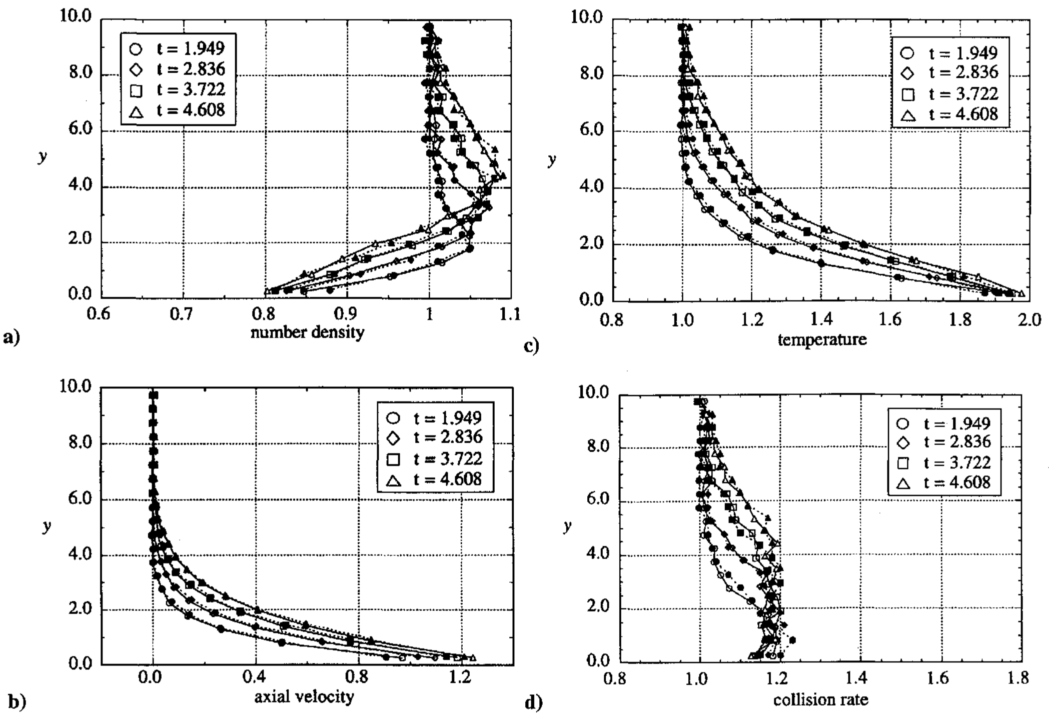

Fig. 9 Comparison between DSMC-MLG serial and parallel results for the Rayleigh validation problem, with u_{1w} = 2.0 and T_{1w} = 1.6: a) axial velocity, b) number density, c) temperature, and d) collision-rate profiles at various sampling times. The results are sampled at $x = x_{\max}/2$.

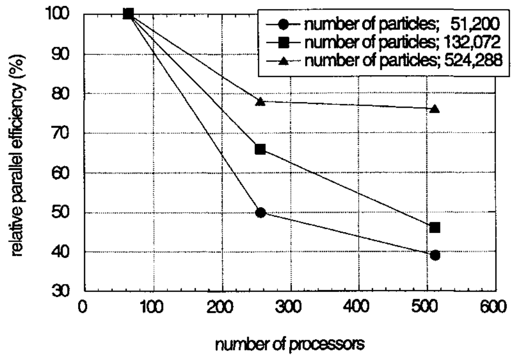

Fig. 10 Comparison of the relative parallel efficiencies on 64, 256, and 512 processors using three different numbers of simulated particles: 51,200, 132,072, and 524,288.

unsteady computations. The global time step Δt_g is 20% of the mean collision time of the undisturbed gas. All variables are nondimensionalized such that one unit of length is a freestream mean free path, and a unit of time is the freestream mean collision time.

Time histories of various macroscopic properties show the effects of the moving hot wall on the undisturbed gas. Comparisons between serial and parallel results in terms of number density, axial velocity, temperature, and collision-rate profiles at four different sampling time intervals are shown in Figs. 9a–9d. The location of each symbol in the curves represents the ensemble-averaged coordinates (the grid distribution) of the center of mass of a particular template. The open symbols connected by solid lines and closed symbols connected by dotted lines indicate parallel and serial results, respectively. In general, all profiles show very good agreement.

We now turn our attention to the performance of the parallel code. The parallel efficiency, defined as the increase in computing speed divided by the number of processors used, is shown in Fig. 10 for runs based on 51,200, 132,072, and 524,288 simulated particles. The results have been scaled so that the parallel efficiency using 64 processors is defined to be 100%. For the calculations done using 51,200 particles, the parallel efficiency is poor, dropping below 40% at 512 processors. Although the number of particles is much larger than the number of physical processors, the number of template arrays (for this case 800) is less than the effective number of physical processors. Thus the computer is not fully used during template-level routines such as collisions. Since each physical processor has four vector units, the effective number of processors is really four times the number of physical processors. The performance of the code improves dramatically as the number of particles is increased from 51,200 to 132,072. In this case, the number of templates, 2048, is equal to the effective number of processors (512 × 4) when the entire machine is used. Thus the computing speed on the CM-5 is sensitive to the number of array elements. Further gains in efficiency are seen upon increasing the number of particles to 524,288. The better performance in this case is because the ratio of communications to computations costs goes down as the problem size is increased.

Discussion and Conclusions

A number of attempts have been made to develop efficient parallel DSMC codes. Since each study was carried out for a different test problem, with different numbers of particles and machine architectures, it is difficult to make a direct comparison between them. Nonetheless, in an attempt to compare the different efforts, we have scaled the reported increases in computing speeds to the performance on a single-processor Cray Y-MP. The parallel efficiencies for the present work are defined as the speedup, relative to the simulations using 64 processors, divided by the number of processors. The results of the various parallelization efforts are summarized in Table 1. In the present study we were able to achieve parallel efficiencies as high as 77%.

In the previous studies, comparable parallel efficiencies have been obtained using MIMD and SIMD architectures. The parallel MIMD codes achieved their speedups primarily through the development of load-balancing algorithms based on physical space partitioning.[32–34] On SIMD machines, such as the Thinking Machine CM-2[29,35] and IBM SP-1,[36] most of the efforts have focused on hardware optimization and the development of data structures to use the hardware most efficiently. Even in the best cases, the parallel efficiencies of the DSMC codes are significantly below those obtained for continuum computational fluid dynamics codes. This is primarily a result of the communication-intensive nature of the collision, sorting, and indexing procedures of DSMC codes.

Table 1 Recent DSMC parallelization efforts

Effort	Year	Computer	Number of processors	Reported speed increase	Scaled speed increase[b]
Goldstein and Sturtevant[32]	1989	iPSC/1 (MIMD)	128	——	——
Furlani and Lordi[33]	1989	iPSC/2(MIMD)	32	16 over sp[a] iPSC/2	0.8
Dagum[35]	1991	CM-2(SIMD)	32 K	comparable to sp Cray-2	0.7
Wong and Long[29]	1992	CM-2(SIMD)	16 K	1.6 over sp Cray Y-MP	1.6
Wilmoth[34]	1992	iPSC/860(MIMD)	128	34 over sp iPSC/860	8.5
Dietrich and Boyd[36]	1994	SP-1(SIMD)	32	16 over sp SP-1	6
Present study	1995	CM-5(SIMD and MIMD)	256 and 512	140 over Sun Sparc 2 workstation	14

[a]Single processor. [b]Scaled to single processor Cray Y-MP performance.

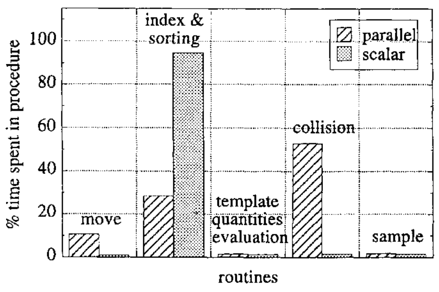

Fig. 11 Comparison of the load distribution by process inside the main execution loop for the serial and parallel DSMC-MLG codes for a simulation involving 39,200 particles.

The primary feature of the DSMC-MLG algorithm is the automatic grid adaptation provided by the Lagrangian data structure. A number of computational benefits arise as a direct consequence, including automatic changes in grid resolution according to properties of the flow, satisfaction of the DSMC requirement that cell dimensions be very small in directions of large macroscopic gradients,[37] higher accuracy to grid dimension, and simplified representation of complex geometries. All of these help to improve the performance of the original DSMC algorithm.

On a serial computer, such as a Sun workstation, the DSMC-MLG may take three or four times longer to run than the standard DSMC.[22] However, coupling an MLG data structure to the DSMC method provides significant improvements in the results, primarily because of the automatic grid restructuring. The present study, which uses a 256 processor CM-5, shows a large speed increase. The gain in computational speed on the CM-5 also comes from optimizing the MLG sorting and tracking procedures for a given parallel application.

The basis of our parallelization of the DSMC-MLG algorithm is the use of a two-level mapping scheme involving parallel arrays on both the template and particle levels. This approach minimizes the computational costs during the binary collision process and makes efficient use of memory for storing template quantities. For small numbers of particles, the parallel efficiency of the DSMC-MLG code decreases dramatically, but as the number of particles is increased, the CM-5 is very efficiently used, and we obtain 80% of parallel efficiency, which is the best ever reported. Further optimization of the DSMC-MLG algorithm should result in even faster computation and therefore allow simulation of more physically and geometrically complex flows.

Figure 11 shows a comparison of the load distribution by process inside the main execution loop for the serial and parallel DSMC-MLG codes. In the serial code, the sorting and indexing procedures accounted for nearly all of the CPU time. Parallelizing the code redistributes the computing cost among the indexing, collisions, and convection processes. The sorting and indexing processes, which take most of the time in the serial case, are fast in the parallel case because of the efficient communication scheme. However, sorting and indexing still take 30% of computational time. The parallel collision process takes almost half of the computational time. Further efforts will focus on optimizing both the collisions and the optimizing sorting and indexing.

One objective in parallelizing the DSMC-MLG is to achieve load balancing during the collision evaluation procedure. Because of the statistical nature of the collision process, a different number of collisions occurs in each template. The time spent in the collision routine depends on the largest number of collisions required in any one template. Therefore, a load-balancing algorithm is needed that measures the load at the template level and distributes the load evenly over idle processors. In principle, this could be done by using the MIMD feature of the CM-5.

The sorting procedures of the current DSMC-MLG algorithm require specification of a grid of size $M_x \times M_y$ at the beginning of the computation, and this is kept constant during the course of a simulation. However, the use of a fixed grid size throughout the course of a simulation may result in directionally biased grid resolution[22] since the optimal aspect ratio (x/y) may change as a function of time in a multidimensional flow. Incorporating a method that changes the aspect ratio as a function of the developing flowfield would ensure comparable grid resolution in every direction. For example, if there are larger gradients in the y direction, it would be better to use a larger percentage of the total number of MLG templates in the y direction. The ability to update the grid size as a function of time could be done by using a grid-reshaping function available in CM-Fortran.

One reason for developing DSMC-MLG is the possibility of treating complex and moving geometry with automatic regridding ability. Modeling more complex geometries could be possible by distributing molecules along stationary or moving solid boundaries that define the geometry itself. A flag added to the physical attributes stored by the MLG data structure identifies these molecules as boundary points so that they could be treated appropriately.

Acknowledgments

This research is jointly supported by the Advanced Research Projects Agency and the Office of Naval Research through the Naval Research Laboratory. The first author thanks the National Research Council (NRC) for support as a NRC research fellow.

References

[1]Bird, G. A., *Molecular Gas Dynamics and the Direct Simulation of Gas Flows*, Clarendon, Oxford, England, UK, 1994.

[2]Haas, B. L., and Feiereisen, W. J., "Particle Simulation of Rarefied Aeropass Maneuvers of the Magellan Spacecraft," *Journal of Spacecraft and Rockets*, Vol. 31, No. 1, 1994, pp. 17–24.

[3]Moss, J. N., Mitcheltree, R. A., Dogra, V. K., and Wilmoth, R. G., "Direct Simulation Monte Carlo and Navier–Stokes Simulations of Blunt Body Wake Flows," *AIAA Journal*, Vol. 32, No. 7, 1994, pp. 1399–1406.

[4]Haas, B. L., and Milos, F. S., "Simulated Rarefied Entry of the Galileo Probe into the Atmosphere of Jupiter," AIAA Paper 94-2043, June 1994.

[5]Boyd, I. D., Penko, P. F., Meisser, D. L., and DeWitt, K. J., "Experimental and Numerical Investigations of Low-Density Nozzle and Plume Flows of Nitrogen," *AIAA Journal*, Vol. 30, No. 10, 1992, pp. 2453–2461.

[6]Woronowicz, M. S., and Rault, D. F., "On Plume Flowfield Analysis and Simulation Techniques," AIAA Paper 94-2048, June 1994.

[7]Kannenberg, K. C., Boyd, I. D., and Dietrich, S., "Development of an Object-Oriented Parallel Code for Plume Impingement Studies," AIAA Paper 95-2052, June 1995.

[8]Boyd, I. D., Beattie, D. R., and Cappelli, M. A., "Numerical and Experimental Investigations of Low-Density Supersonic Jets of Hydrogen," *Journal of Fluid Mechanics*, Vol. 280, Dec. 1994, pp. 41–67.

[9]Gilmore, M. R., and Warburton, K., "Axisymmetric Hypersonic Jet Interaction: A Combined Experimental and Computational Study II," AIAA Paper 95-0414, Jan. 1995.

[10]Chung, C. H., Kim, S. C., Stubbs, R. M., and DeWitt, K. J., "DSMC and Continuum Analyses of Low-Density Nozzle Flow," AIAA Paper 93-0727, Jan. 1993.

[11]Zelesnik, D., Micci, M. M., and Long, L. N., "Direct Simulation Monte Carlo Model of Low Reynolds Number Nozzle Flow," *Journal of Propulsion and Power*, Vol. 10, No. 4, 1994, pp. 546–553.

[12]O'Conner, L., "MEMS: Microelectromechanical System," *Mechanical Engineering*, Vol. 114, Feb. 1992, pp. 40–47.

[13]Trimmer, W. S. N., "Microrobots and Micromechanical Systems," *Sensors and Actuators*, Vol. 19, No. 3, 1989, pp. 267–287.

[14]Pong, K.-C., Ho, C.-M., Liu, J., and Tai, Y.-C., "Non-Linear Pressure Distribution in Microchannels," American Society of Mechanical Engineers Winter Annual Meeting, Chicago, IL, Nov. 1994.

[15]Liu, C., Tai, Y.-C., and Ho, C.-M., "Micromachined Magnetic Actuators for Active Fluid Control," Winter Annual Meeting, Chicago, IL, American Society of Mechanical Engineers, Nov. 1994.

[16]Scott, W. B., "Micro-Machines Hold Promise for Aerospace," *Aviation Week and Space Technology*, Vol. 138, March 1993, pp. 36–39.

[17]Ikegawa, M., and Kobayashi, J., "Deposition Profile Simulation Using the Direct Simulation Monte Carlo Method," *Journal of Electrochemical Society*, Vol. 136, No. 10, 1989, pp. 2982–2986.

[18]Gad-el-Hak, M., "Interactive Control of Turbulent Boundary Layers: A Futuristic Overview," *AIAA Journal*, Vol. 32, No. 9, 1994, pp. 1753–1765.

[19]Ho, C.-M., Leu, T.-S., and Miu, D., "Control of Macro Machine by Micro Actuators," *Bulletin of American Physical Society*, Vol. 39, No. 9, 1994, p. 1909.

[20]Boris, J. P., "A Vectorized 'Near Neighbor' Algorithm of Order N Using a Monotonic Logical Grid," *Journal of Computational Physics*, Vol. 66, No. 1, 1986, pp. 1–20.

[21]Cybyk, B. Z., "Combining the Monotonic Lagrangian Grid with Direct Simulation Monte Carlo; A New Approach for Low-Density Flows," Ph.D. Dissertation, Aerospace Engineering, Univ. of Maryland, College Park, MD, 1994.

[22]Cybyk, B. Z., Oran, E. S., Boris, J. P., and Anderson, J. D., Jr., "Combining the Monotonic Lagrangian Grid with a Direct Simulation Monte Carlo Model," *Journal of Computational Physics*, Vol. 122, No. 2, 1995, pp. 323–334.

[23]Lambrakos, S. G., Peyrard, M., Oran, E. S., and Boris, J. P., "Molecular Dynamics Simulation of Shock-Induced Detonations in Solids," *Physical Review B: Condensed Matter*, Vol. 39, No. 2, 1989, pp. 993–1005.

[24]Lambrakos, S. G., Boris, J. P., Guirguis, R. H., Page, M., and Oran, E. S., "Molecular Dynamics Simulation of $(N_2)_2$ Formation Using the Monotonic Lagrangian Grid," *Journal of Chemical Physics*, Vol. 90, No. 8, 1989, pp. 4473–4481.

[25]Phillips, L., Sinkovits, R. S., Oran, E. S., and Boris, J. P., "The Interaction of Shocks and Defects in Lennard-Jones Crystals," *Journal of Physics: Condensed Matter*, Vol. 5, No. 5, 1993, pp. 6357–6376.

[26]Sinkovits, R. S., Boris, J. P., and Oran, E. S., "A Technique for Regularizing the Structure of a Monotonic Lagrangian Grid," *Journal of Computational Physics*, Vol. 108, No. 2, 1993, pp. 368–372.

[27]Sinkovits, R. S., Boris, J. P., and Oran, E. S., "The Stability and Multiplicity of the Monotonic Lagrangian Grid," *SIAM Journal on Scientific and Statistical Computing* (submitted for publication).

[28]Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vettering, W. T., *Numerical Recipes*, Cambridge Univ. Press, Cambridge, England, UK, 1989.

[29]Wong, B. C., and Long, L. N., "Direct Simulation Monte Carlo (DSMC) on the Connection Machine," AIAA Paper 92-0564, Jan. 1992.

[30]Wilmoth, R. G., "Direct Simulation Monte Carlo Analysis on Parallel Processors," AIAA Paper 89-1666, June 1989.

[31]Pryor, D. V., and Burns, P. J., "Vectorized Monte Carlo Molecular Aerodynamics Simulation of the Rayleigh Problem," *Proceedings of Supercomputing '88* (Orlando, FL), 1988.

[32]Goldstein, D., and Sturtevant, B., "Discrete Velocity Gasdynamics Simulations in a Parallel Processing Environment," AIAA Paper 89-1668, June 1989.

[33]Furlani, T. R., and Lordi, J. A., "A Comparison of Parallel Algorithms for the Direct Simulation Monte Carlo Method II: Application to Exhaust Plume Flowfields," AIAA Paper 89-1667, June 1989.

[34]Wilmoth, R. G., "Application of a Parallel Direct Simulation Monte Carlo Method to Hypersonic Rarefied Flows," *AIAA Journal*, Vol. 30, No. 10, 1992, pp. 2447–2452.

[35]Dagum, L., "Three Dimensional Direct Particle Simulation on the Connection Machine," AIAA Paper 91-1365, July 1991.

[36]Dietrich, S., and Boyd, I., "A Scalar Optimized Parallel Implementation of the DSMC Method," AIAA Paper 94-0355, Jan. 1994.

[37]Bird, G. A., "Perception of Numerical Methods in Rarefied Gasdynamics," edited by E. P. Muntz, Vol. 118, Progress in Astronautics and Aeronautics, AIAA, Washington, DC, 1989.