Journal of Computational Physics **149,** 95–113 (1999)
Article ID jcph.1998.6140, available online at http://www.idealibrary.com on IDEAL®

# A Method for Estimating the Computational Requirements of DSMC Simulations[1]

Marc A. Rieffel

*Scalable Concurrent Programming Laboratory, Computer Science Department, Syracuse University, Syracuse, New York 13210*
E-mail: marc@scp.syr.edu

Received February 4, 1998; revised October 13, 1998

This paper presents a model for predicting the runtime and storage requirements for direct simulation Monte Carlo (DSMC) simulations of rarefied gas flow. A variety of flow configurations are considered, including internal, external, steady, and unsteady. The analysis is independent of the simulation architecture, gridding technique, collision model, and implementation technique. The model is validated and constants of the model determined for simple test cases. The model is then used to predict the requirements of a realistic three-dimensional simulation, and the results are shown to agree with experiments. Additional predictions define the boundaries of simulations that are feasible with existing computational resources. © 1999 Academic Press

## 1. INTRODUCTION

A variety of simulation techniques are used for the simulation of fluid flow, or gas dynamics. The characteristic parameter that determines gas flow properties is the Knudsen number, $Kn = \lambda/L$, where $\lambda$ is the mean free path in a gas and $L$ is the reference flow scale. In the *continuum regime,* where the Knudsen number tends toward zero, microscopic structure can be ignored, and a system can be completely described in terms of macroscopic parameters such as density, temperature, and velocity. In the *free-molecular regime,* where the Knudsen number tends toward infinity, collisions between molecules can be neglected, and the flow behavior is controlled by interactions between molecules and boundary surfaces. The region between the continuum and free-molecular regimes, where the Knudsen number is close to unity, is called the *transition regime*.

[1] Infrastructure, support, and computing resources for this research were provided by BMDO under Contract DAAH04-96-1-0319. The research described in this report is sponsored by Intel Corporation and the Advanced Research Projects Agency under Contract DABT63-95-C-0116. This work includes Russian participation that is supported by the U.S. Civilian Research and Development Foundation under Award RE1241.

In the transition regime, viscosity, heat conduction, relaxation, diffusion, and chemical processes are important, and it is also possible for velocity distribution functions to be non-Maxwellian, resulting in strong thermal nonequilibrium. As thermal and chemical relaxation lengths may be comparable to the reference flow scale, differences between translational, rotational, and vibrational temperatures may be important.

Several numerical techniques for simulating transitional gas flow have been developed in the past 20 years. Navier–Stokes and viscous shock layer equations can typically be used for the simulation of near-continuum flows, with appropriate extensions for modeling slip velocity and temperature jumps at surfaces. Because the Navier–Stokes equations assume only small deviations from thermal equilibrium, however, they are not suitable for studying rarefied flows with flow disturbances, such as shock waves, in which the velocity distribution functions are strongly nonequilibrium.

The governing equation in the transition regime is the Boltzmann equation, a detailed treatment of which can be found in [11, 12, or 21]. It is a nonlinear integral–differential equation, closed with respect to the one-particle distribution function, which in turn determines the density of particles in a six-dimensional phase space of particle coordinates and velocities.

Some approaches for solving the Boltzmann equation include direct integration, molecular dynamics methods, the direct simulation Monte Carlo (DSMC) method, techniques coupling both DSMC and continuum methods [8], model equation approaches [31], and the test particle method [14]. The DSMC method is the approach of choice for the study of complex multidimensional flows of rarefied hypersonic aerothermodynamics. Reasons for this include the simple transition from one-dimensional to two- and three-dimensional problems, and the ease with which complex models of particle interaction can be incorporated without substantial increase in computational costs [18]. It is also well suited for use on modern concurrent architectures [28].

Systems that the DSMC method can be used to simulate include space vehicles in the upper atmosphere [6, 16, 17], plasma reactors for semiconductor manufacturing [2, 29, 33], lava flow from volcanoes [1], and many others.

The DSMC method was pioneered by Bird [4, 5, 7]. It can be used to model chemical reactions and has been extended to address translational and rotational effects in gaseous expansions [3] and to include the maximum entropy (ME) and Borgnakke–Larsen variants (BL) [23]. Sophisticated models have been developed for energy transfer between vibrational and translational modes, such as those used in simulating flow over a two-dimensional wedge [9]. Chemical reaction models have been used to model reacting flows [6]. DSMC has also been combined with fluid electron models and self-consistent electric fields to simulate plasma systems [2, 26].

In principle, the DSMC technique can account for all of the physics needed for any problem [24]. It is, therefore, a pure form of computational fluid dynamics. In practice, however, the technique can be substantially more computationally intensive than continuum approaches. The goal of the present work is to study the computational cost of DSMC simulations, in terms of the physical parameters of the systems that are being modeled, such as density, temperature, and velocity. Predictive models for simulation time and storage requirements are developed. These models are independent of the simulation architecture, gridding technique, collision model, and implementation technique. The models are developed for three-dimensional simulations, but they can also be generalized to $n$-dimensional simulations [30].

The applications of these models are threefold. First, for existing DSMC applications, they facilitate the understanding of how changes in simulation parameters affect changes in computational requirements. Second, they may be useful in comparing DSMC to other techniques for a given problem. Finally, they provide a straightforward method for determining whether a desired simulation is feasible, given available time, processing, and memory constraints.

## 2. NUMERICAL METHOD

The direct simulation Monte Carlo method is an approach for solving the Boltzmann equation by simulating the behavior of individual particles. Since it is impossible to simulate the actual number of particles in a realistic system, a smaller number of simulation particles are used, each representing a large number of real particles. A computational grid is used to represent the simulated region. Statistical techniques are employed to reproduce the correct macroscopic behavior. Figure 1 shows a schematic of simulated particles and grid cells in a DSMC computation. In three dimensions, cubic, hexahedral, tetrahedral, or prismatic cells may be used, depending on the implementation and specific simulation requirements.

The DSMC algorithm is shown in Fig. 2. Initially, grid cells are filled with simulation particles according to density, temperature, and velocity specifications. A simulation then takes discrete steps in time. Particle motion and interactions are decoupled over the duration of a timestep. Each timestep is composed of two phases, *transport*, where particles move between grid cells, and *collisions*, where particles interact within a cell. Macroscopic properties, such as density and temperature, are computed by appropriate averaging of particle properties.

The transport phase, concerned with moving particles through the computational grid for a specified period of time, may be implemented in several ways. For simple Cartesian grids, particle destinations are quickly computed, and particle cell destinations are computed using indexing schemes [15]. For more complex grids, such as hexahedral or tetrahedral, ray-tracing techniques can be used to determine particle positions in space and in the grid at the end of a timestep [28]. Interactions between particles and boundary surfaces may also take place during the transport phase.

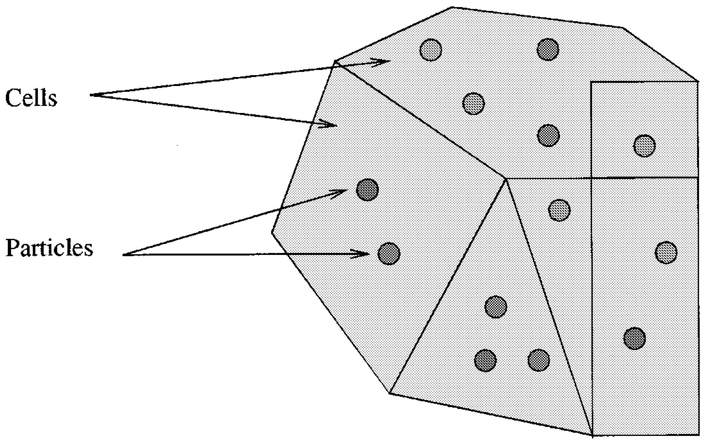

**FIG. 1.** Cells and particles in a DSMC simulation.

```
1. Initialize cells according to initial conditions
2. While more steps are necessary
    (a) Move particles (Transport phase)
    (b) Collide particles (Collision phase)
    (c) Compute global information,
        such as the total number of simulated particles
3. Compute results from cell and particle information
4. Conclude computation
```

**FIG. 2.** The DSMC algorithm.

The collision phase implements particle–particle interactions. Cell sizes are chosen so that only collisions between particles in the same cell must be considered. The number of collisions that take place in a cell is a function of the number of particles in the cell and the volume of the cell, as well as model-specific parameters, such as the particle mass and collision cross section.

While the DSMC algorithm appears simple, its complexity can only be determined from careful consideration of computational and physical parameters. The primary constraints on the DSMC method are: (1) the cell size must be proportional to the local mean free path; (2) the number of particles per cell must be roughly constant in order to preserve collision statistics; and (3) the simulation timestep must be chosen so that particles only traverse a fraction of the average cell length per timestep [7]. The following sections study the implications of these constraints and present models for the run time and memory requirements of DSMC simulations.

Several classes of simulations must be considered when evaluating computational requirements. Simulations can be either steady state or unsteady and can be for either internal or external flows. The computational requirements of unsteady simulations are different from, and may be much greater than, those of steady simulations. Internal flow simulations typically use simulation volumes that are specified by the physical geometry of the problem, while external flows use simulation volumes determined by flow properties. The performance implications of these different classes of simulations are discussed below.

## 3. COMPUTATIONAL COMPLEXITY ANALYSIS

The important physical parameters for a simulation depend on the type of simulation. Internal-flow systems, such as plasma reactors, are typically characterized by the particle density, the simulated volume, and the collision cross section. Together, the particle density and collision cross section determine the mean-free path and therefore the required size of computational grid cells, while the simulated volume determines the number of cells that are required and the time taken for information to travel across the system. For certain external-flow systems, however, the effects of these parameters are different. For simulations of unsteady flows, it is also important to consider the time during which the system must be simulated, as well as the characteristic oscillation time of the system.

### *Simulation Parameters*

In order to predict the performance cost of the collision and transport phases of a DSMC computation, it is useful to calculate some general system parameters, such as the required number of cells $C$, the simulation timestep $\Delta t$, and the total number of required timesteps, $S$. The number of cells required for a simulation can be determined from the DSMC constraint that the typical cell size, or characteristic cell length, $\bar{l}$ be proportional to the mean free path $\lambda$,

$$\bar{l} = c_\lambda \lambda = \frac{c_\lambda}{n\sigma}, \tag{1}$$

where $c_\lambda$ is a proportionality constant, $n$ is the particle number density (particles per unit volume), and $\sigma$ is the collision cross section. The cell size is also typically proportional to the cube root of the average cell volume,

$$\bar{l} = c_v (V/C)^{1/3}, \tag{2}$$

where $c_v$ is a constant that reflects the type of grid and the skewness of grid cells. Combining Eqs. (1) and (2) and solving for the number of cells $C$ yields

$$C = \left(\frac{c_v^3}{c_\lambda^3}\right) n^3 \sigma^3 V. \tag{3}$$

The number of particles required by a simulation, $N$, is chosen to be proportional to the number of cells required, $N = c_p C$. The number of particles per cell, $c_p$, must be large enough to allow for a sufficient number of collisions per cell and to provide adequate samples for statistics, as discussed below. Using the value of $C$ from (3) gives

$$N = c_p C = c_p \left(\frac{c_v^3}{c_\lambda^3}\right) n^3 \sigma^3 V. \tag{4}$$

A typical simulated system may contain a number of particles comparable to Avogadro's number. As it is computationally infeasible to simulate this number of particles, it is necessary for each simulated particle to represent a large number of real particles. The *weight* of each simulated particle, or the number of real particles represented by each simulated particle, $w_p$, can be written as the ratio of real to simulated particles, which can in turn be written in terms of the number of cells, using Eqs. (3) and (4),

$$w_p = \frac{nV}{c_p C} = \frac{c_\lambda^3}{c_p c_v^3} \left(\frac{1}{n^2 \sigma^3}\right). \tag{5}$$

The total amount of simulation time required for a steady-state simulation to converge depends on the geometry of the system and the thermochemical properties of the gases being simulated. For the purposes of a performance model, however, the convergence time may be based on the acoustic time, the amount of time that it takes for thermal information to traverse the entire width of the simulated region, $L$. The quantity $L$ can be approximated as proportional to the cube root of the simulated volume, $L = c_L V^{1/3}$, where the constant $c_L$ reflects the shape of the simulated region. A long, narrow volume will have a higher $c_L$ than a spherical volume.

The thermal speed $v_t$ for a single-species gas can be computed by, $v_t = \sqrt{8kT/\pi m}$, where $k$ is the Boltzmann constant, $T$ is the gas temperature, and $m$ is the particle mass. Information propagates fastest in high-temperature, low-mass gases. Assuming that $c_a$ acoustic periods are required for convergence, the acoustic, or convergence time $T_{\text{conv}}$, is given by

$$T_{\text{conv}} = c_L c_a \frac{V^{1/3}}{v_t}. \tag{6}$$

The simulation timestep $\Delta t$ should be chosen so that the average distance traveled by a particle in a timestep, $d$, is some fraction, $c_t$, of the average cell length, $\overline{l}$. If particles traverse too many cells in one timestep, results may be inaccurate. On the other hand, too small a timestep will result in inefficient computation. The average distance traveled by a particle in a timestep can be estimated as the product of the timestep $\Delta t$ and the sum of stream and thermal velocities, $\overline{v} + v_t$. Using these approximations, with (6) and (3), yields

$$\Delta t = c_t \frac{l}{v_{\text{total}}} = c_t c_V \frac{(V/C)^{1/3}}{\overline{v} + v_t}, \tag{7}$$

Using the value of $C$ from (3), this can be rewritten

$$\Delta t = \frac{c_t c_\lambda}{(\overline{v} + v_t) n\sigma}. \tag{8}$$

It is important to note that an increase in any of the parameters, $n, \sigma, T$, or $\overline{v}$, results in a decrease in the timestep duration. This in turn results in an increase in the number of steps required for a simulation and, therefore, the simulation time. The number of steps required for convergence, $S_{\text{conv}}$, is the ratio of the acoustic time $T_{\text{conv}}$ to the timestep $\Delta t$. Using (8),

$$S_{\text{conv}} = \frac{T_{\text{conv}}}{\Delta t} = \frac{c_L c_a}{c_t c_\lambda}\left(1 + \frac{\overline{v}}{v_t}\right) n\sigma V^{1/3}. \tag{9}$$

In addition to considering the time required for a simulation to converge, it is also important to examine the trade-off between execution time, memory usage, and solution quality. One measure of the quality of a solution is determined by the noise, or statistical scatter. The statistical scatter is determined by the number of *samples*, or the product of the average number of particles in a cell, $c_p$, and the number of steps over which macroscopic properties are averaged, $S_s$. Assuming that the scatter follows a Poisson distribution, the fractional error, $e$, is inversely proportional to the square root of the number of samples, $N_s$ [7],

$$e = \frac{1}{\sqrt{N_s}} = \frac{1}{\sqrt{c_p S_s}}. \tag{10}$$

In order to obtain $r$ samples per cell, it is necessary to average results over $r/c_p$ steps.

In the following sections, these parameters are used to compute the amount of time required for the transport and collision phases of a DSMC timestep.

## *Transport Phase*

The time required for the transport phase of a timestep is given by the product of time required to move one particle, $T_t$, and the number of particles, $N$,

$$T_{\text{trans}} = T_t N = T_t c_p C. \tag{11}$$

The parameter $T_t$ is dependent upon both the machine speed and the implementation of the transport model. Using the value for $N$ computed in (4), $T_{\text{trans}}$ can be written

$$T_{\text{trans}} = T_t N = T_t \frac{c_p c_v^3}{c_\lambda^3} n^3 \sigma^3 V. \tag{12}$$

*Collision Phase*

The time required to compute collisions in a DSMC timestep is proportional to the number of collisions. Consider a computational cell of volume $V_{\text{cell}}$ that contains $c_p$ particles. Using the hard sphere (HS) collision model, the number of collisions in that cell during a given timestep $\Delta t$ is given by,

$$N_c = \frac{c_p(c_p - 1)\sigma v_r w_p}{2V_{\text{cell}}} \Delta t, \tag{13}$$

where $\sigma$ is the collision cross section and $v_r$ is the relative velocity between particles. For a single-species gas, the mean relative velocity is given by the equation $v_r = \sqrt{16kT/\pi m}$, and can therefore be written in terms of the mean thermal velocity, $v_r = v_t\sqrt{2}$. Using the timestep duration $\Delta t$ from (8) and the particle weight computed in (5), as well as the approximation, $V_{\text{cell}} = V/C$, yields

$$N_c = \frac{(c_p - 1)c_t c_\lambda}{\sqrt{2}} \left( \frac{v_t}{\overline{v} + v_t} \right). \tag{14}$$

The total time spent on collisions is one timestep is then the product of the time spent on each collision, $T_c$, the number of collisions per cell, $N_c$, and the number of cells, $C$. Combining (14) and (3) yields

$$T_{\text{col}} = T_c N_c C = T_c \frac{(c_p - 1)c_t c_v^3}{\sqrt{2}c_\lambda^2} \left( \frac{v_t}{\overline{v} + v_t} \right) n^3 \sigma^3 V. \tag{15}$$

As with transport, timestep collision time is proportional to the cube of both density and cross section, and to the first power of the domain volume. This analysis was developed for the HS model, where $\sigma$ is a constant. For the VHS model, $\sigma$ is a function of relative velocity. Thorough analysis of the cost of the variable hard sphere (VHS) model could be completed with integration over relative velocities. The result, however, would have roughly the same dependence on $\sigma^3$. The extension to the variable soft sphere (VSS) model does not affect the number of collisions, only the postcollision scattering angle. Because more computation is required with each collision, this would have the effect of increasing $T_c$, but it would not change the dependence on the other parameters, $n$, $\sigma$, and $V$.

*Timestep Time*

The preceding sections facilitate the prediction of the time required per timestep of a simulation. Combining Eqs. (12) and (15) yields the entire time required for one timestep, given by the sum of transport and collision times,

$$T_{\text{one}} = \frac{c_v^3}{c_\lambda^3} \left[ c_p T_t + \frac{c_t(c_p - 1)c_\lambda}{\sqrt{2}} \left( \frac{v_t}{\overline{v} + v_t} \right) T_c \right] n^3 \sigma^3 V. \tag{16}$$

This analysis does not depend on the type of grid used or the implementation of the transport or collision phases. All particle transport algorithms must execute in time proportional to the number of timesteps and the number of particles, and all collision algorithms must execute in time proportional to the number of collisions, thus yielding the same dependence on the physical parameters $n$, $\sigma$, and $V$, and the constants $c_p$, $c_l$, $c_a$, $c_v$, $c_t$, and $c_\lambda$.

### *Memory Requirements*

In addition to modeling the required execution time for convergence and averaging phases, it is also important to predict the storage requirements for a simulation. The two primary uses of memory are particles and cells. For small simulations, it is also important to consider the amount of *overhead* memory, $M_0$, consumed by the application code and any constant-sized data structures. If the memory required for a single particle is $M_p$ and the memory required for a single cell is $M_c$, the total memory required for a simulation, $M_{\text{DSMC}}$, can be written

$$M_{\text{DSMC}} = M_0 + M_pN + M_cC = M_0 + M_pc_pC + M_cC = M_0 + (M_pc_p + M_c)C. \quad (17)$$

## 4. FLOW CONFIGURATIONS

This section considers the four possible flow configurations: steady-state internal, steady-state external, unsteady internal, and unsteady external. The computational requirements of each of these configurations have fundamentally different dependences on the physical parameters. In order to provide a complete understanding of the complexity of the DSMC method, it is essential to consider each configuration separately.

### *Steady-State Internal Flows*

In order to compute a steady-state flow, the simulation is first run to convergence, and then run for additional steps in order to sample and average macroscopic parameters. The total simulation time is the sum of convergence and averaging times. The time required to converge an internal-flow, steady-state simulation, $K_{\text{steady}}^{\text{internal}}$, is the product of the time for each timestep and the number of steps required. Using results from the previous section, this yields

$$K_{\text{steady}}^{\text{internal}} = T_{\text{one}}S_{\text{conv}} = \frac{c_v^3 c_L c_a}{c_\lambda^3}\left(\frac{(c_p-1)T_c}{\sqrt{2}} + \frac{c_pT_t}{c_tc_\lambda}\right)n^4\sigma^4V^{4/3}. \quad (18)$$

The duration of the averaging portion of a steady computation, $A_{\text{steady}}^{\text{internal}}$, is governed by the desired accuracy, or number of samples. Averaging time, for a desired number of samples $r$, is the product of the time required for each timestep, $T_{\text{one}}$, and the number of steps required, $S_s$,

$$A_{\text{steady}}^{\text{internal}} = T_{\text{one}}S_s = T_{\text{one}}\frac{r}{c_p}. \quad (19)$$

The averaging time can then be written

$$A_{\text{steady}}^{\text{internal}} = \frac{c_v^3}{c_\lambda^3}\left[T_t + T_c\frac{c_tc_\lambda}{\sqrt{2}}\left(1-\frac{1}{c_p}\right)\left(\frac{v_t}{\bar{v}+v_t}\right)\right]n^3\sigma^3Vr. \quad (20)$$

In other words, the time required to obtain smooth results is proportional to the cube of the density, the cube of the cross section, and the volume. It is also proportional to the the desired number of samples, or the inverse square of the desired error.

For a steady-state, fixed-volume simulation, the memory requirements from Eq. (17) reduce to

$$M_{\text{steady}}^{\text{internal}} = M_0 + (M_p c_p + M_c)\frac{c_v^3}{c_\lambda^3} n^3 \sigma^3 V. \tag{21}$$

The storage requirements for a simulation are therefore proportional to the cube of the density, the cube of the cross section, and the volume of the simulated domain.

## *Steady-State External Flows*

For certain classes of external flow problems, it is appropriate to adjust the size of the computational domain according to the other physical parameters. As a first approximation, it is sometimes possible to use a computational volume with length proportional to the mean free path, $\lambda$, or volume proportional to $\lambda^3$,

$$V = (c_d\lambda)^3 = \frac{c_d^3}{n^3\sigma^3}, \tag{22}$$

where $c_d$ is the number of mean free paths to be simulated. The convergence time for an external flow can then be written

$$K_{\text{steady}}^{\text{external}} = \frac{c_d^3 c_v^3 c_L c_a}{c_\lambda^3}\left(\frac{(c_p - 1)T_c}{\sqrt{2}} + \frac{c_p T_t}{c_t c_\lambda}\right). \tag{23}$$

In other words, the convergence time is not a function of cross section or density. Similarly, the averaging time for an external flow, $A_{\text{steady}}^{\text{external}}$, for a desired number of samples, $r$, can then be computed as

$$A_{\text{steady}}^{\text{external}} = \frac{c_v^3 c_d^3}{c_\lambda^3}\left[T_t + T_c\frac{c_t c_\lambda}{\sqrt{2}}\left(1 - \frac{1}{c_p}\right)\left(\frac{v_t}{\bar{v} + v_t}\right)\right] r. \tag{24}$$

The memory requirements of a steady external simulation can be written

$$M_{\text{steady}}^{\text{external}} = M_0 + \frac{c_d^3 c_v^3}{c_\lambda^3}(M_p c_p + M_c). \tag{25}$$

Just as simulation time is not a function of cross section or density for this class of problems, storage requirements are not functions of cross section or density. For some important systems, it is possible to adjust the volume with the mean free path, but not in a directly proportional manner. For these cases, the computational requirements may be estimated by using the analysis of this section, together with the analysis of the previous section.

## *Unsteady Internal Flows*

For an unsteady problem, the total simulated time is a specified parameter, not determined by convergence time. Consider an unsteady simulation of a time interval $T_u$, with a characteristic oscillation time $\tau$. The ratio $T_u/\tau$ is the number of periods to be simulated. Because the flow is changing, it is not possible to average results over a large number of steps. There are therefore two ways to obtain smooth results for unsteady flows. One is to choose a number of particles such that results averaged over a small number of steps will be sufficiently smooth. The other approach is to run a number of simulations with a small number of particles, but using a different random seed for each. The results from the different simulations can then be averaged together to obtain smooth results. While both methods require approximately the same amount of simulation time, the first method requires significantly more memory. The following analysis assumes that $P$ separate simulations are used, where $P = 1$ corresponds to the first approach, and $P > 1$ corresponds to the second approach.

For unsteady flows, results can only be averaged over a short period of time during which the flow remains approximately unchanged. The number of particles per cell, $c_p$, must be chosen so that the desired number of samples can be obtained while the flow is unchanged. It must be assumed that the flow is unchanged over some (small) fraction of $\tau$. Sampling can then take place during a time $c_\tau \tau$. The number of steps over which it is possible to average is given by

$$S_u = \frac{c_\tau \tau}{\Delta t}. \tag{26}$$

The number of samples, $r$, obtained with $P$ separate simulations is the product of particles and steps,

$$r = c_p S_u = c_p P \frac{c_\tau \tau}{\Delta t}. \tag{27}$$

Equation (27) can be solved to determine the minimum number of particles required per cell, $p_{\min}$,

$$p_{\min} = \frac{r\,\Delta t}{c_\tau \tau P}, \tag{28}$$

which can be rewritten using (8)

$$p_{\min} = \frac{c_t}{c_\tau c_\lambda}\left(\frac{1}{\overline{v} + v_r}\right)\frac{r}{P n \sigma \tau}. \tag{29}$$

The computational time required for $P$ unsteady simulations is given by the product of the time taken for each timestep, $T_{\text{one}}$, and the number of steps that must be simulated, $S_u$. This, in turn, is the ratio of the unsteady time $T_u$ to the timestep $\Delta t$,

$$A^{\text{internal}}_{\text{unsteady}} = T_{\text{one}} \frac{T_u}{\Delta t}. \tag{30}$$

Using previously computed values and the approximation $c_p - 1 \approx c_p$, this yields

$$A^{\text{internal}}_{\text{unsteady}} = \frac{c_p c_v^3}{c_t c_\lambda^2}\left[T_t + T_c \frac{c_t}{c_\lambda}\left(\frac{v_r}{\overline{v} + v_t}\right)\right](\overline{v} + v_t) n^4 \sigma^4 V T_u. \tag{31}$$

Substituting the minimum number of particles $p_m$ for $c_p$ in (31) yields

$$\begin{aligned} A^{\text{internal}}_{\text{unsteady}} &= \frac{p_m c_v^3}{c_t c_\lambda^2}\left[T_t + T_c \frac{c_t}{c_\lambda}\left(\frac{v_r}{\overline{v} + v_t}\right)\right](\overline{v} + v_t) n^4 \sigma^4 V T_u \qquad (32)\\ &= \frac{c_v^3}{c_t c_\lambda^3 c_\tau}\left[T_t + T_c \frac{c_t}{c_\lambda}\left(\frac{v_r}{\overline{v} + v_t}\right)\right]\frac{n^3 \sigma^3 V r T_u}{\tau}. \qquad (33)\end{aligned}$$

This shows that for unsteady simulations with a given oscillation period $\tau$, the computation time is proportional to the cube of both the density and the cross section, proportional to the volume, the number of desired samples, and the simulated time, but inversely proportional to the oscillation time. The cost of unsteady simulations therefore does not grow as fast as the cost of constant-volume steady simulations, primarily for the reason that unsteady simulations are not required to converge.

For typical values of $n$, $\sigma$, $V$, and $\tau$, however, the number of particles required for an unsteady simulation is very much greater than the number required for a steady simulation. The time and memory requirements of unsteady calculations are therefore substantially greater than for steady calculations. In some cases, the initial conditions may be sufficiently uncertain or complicated that a steady simulation must be converged to determine those conditions before an unsteady computation can begin, further increasing the cost of unsteady simulations.

In order to estimate the memory requirements for each unsteady internal flow simulation, the previously computed value of $p_m$ can be used in (21) to obtain

$$M_{\text{unsteady}}^{\text{internal}} = M_0 + \frac{c_v^3}{c_\lambda^3}\left[M_p \frac{c_t}{c_\tau c_\lambda}\left(\frac{1}{\bar{v} + v_r}\right)\frac{n^2\sigma^2 r}{\tau P} + M_c n^3\sigma^3\right]V. \tag{34}$$

## *Unsteady External Flows*

The simulation time for an unsteady external flow computation can be obtained by combining (32) and (28), yielding

$$A_{\text{unsteady}}^{\text{internal}} = \frac{c_v^3 c_d^3}{c_t c_\lambda^3 c_\tau}\left[T_t + T_c \frac{c_t}{c_\lambda}\left(\frac{v_r}{\bar{v} + v_t}\right)\right]\frac{r T_u}{\tau}. \tag{35}$$

Similarly, memory requirements can be obtained by reducing Eq. (34) to

$$M_{\text{unsteady}}^{\text{external}} = M_0 + \frac{c_d^3 c_v^3}{c_\lambda^3}\left[M_p \frac{c_t}{c_\tau c_\lambda}\left(\frac{1}{\bar{v} + v_r}\right)\frac{r}{n\sigma\tau} + M_c\right]. \tag{36}$$

## *Summary*

The simulation execution time for the different configurations is summarized in Table I, while the storage requirements for the different configurations are summarized in Table II.

**TABLE I**
**Summary of Simulation Times for Different Configurations**

| | Internal | External |
|---|---|---|
| Steady | $\frac{c_v^3 c_L c_a}{c_\lambda^3}\left(\frac{(c_p-1)T_c}{\sqrt{2}} + \frac{c_p T_t}{c_t c_\lambda}\right) n^4\sigma^4 V^{4/3} + \frac{c_v^3}{c_\lambda^3}\left[T_t + T_c \frac{c_t c_\lambda}{\sqrt{2}}\left(1 - \frac{1}{c_p}\right)\left(\frac{v_t}{\bar{v}+v_t}\right)\right] n^3\sigma^3 V r$ | $\frac{c_d^3 c_v^3 c_L c_a}{c_\lambda^3}\left(\frac{(c_p-1)T_c}{\sqrt{2}} + \frac{c_p T_t}{c_t c_\lambda}\right) + \frac{c_v^3 c_d^3}{c_\lambda^3}\left[T_t + T_c \frac{c_t c_\lambda}{\sqrt{2}}\left(1 - \frac{1}{c_p}\right)\left(\frac{v_t}{\bar{v}+v_t}\right)\right] r$ |
| Unsteady | $\frac{c_v^3}{c_t c_\lambda^3 c_\tau}\left[T_t + T_c \frac{c_t}{c_\lambda}\left(\frac{v_r}{\bar{v}+v_t}\right)\right]\frac{n^3\sigma^3 V r T_u}{\tau}$ | $\frac{c_v^3 c_d^3}{c_t c_\lambda^3 c_\tau}\left[T_t + T_c \frac{c_t}{c_\lambda}\left(\frac{v_r}{\bar{v}+v_t}\right)\right]\frac{r T_u}{\tau}$ |

**TABLE II**
**Summary of Memory Requirements for Different Configurations**

| | Internal | External |
|---|---|---|
| Steady | $M_0 + (M_p c_p + M_c)\frac{c_v^3}{c_\lambda^3} n^3 \sigma^3 V$ | $M_0 + \frac{c_d^3 c_v^3}{c_\lambda^3}(M_p c_p + M_c)$ |
| Unsteady | $M_0 + \frac{c_v^3}{c_\lambda^3}\left[M_p \frac{c_t}{c_\tau c_\lambda}\left(\frac{1}{\bar{v} + v_r}\right)\frac{n^2\sigma^2 r}{\tau} + M_c n^3 \sigma^3\right] V$ | $M_0 + \frac{c_d^3 c_v^3}{c_\lambda^3}\left[M_p \frac{c_t}{c_\tau c_\lambda}\left(\frac{1}{\bar{v} + v_r}\right)\frac{r}{n\sigma\tau} + M_c\right]$ |

## 5. PARAMETER ESTIMATION

This section considers the parameters required for predicting runtime and storage requirements for DSMC simulations. These parameters can be grouped in two classes: those that are implementation-dependent or architecture-dependent and those that are not. The former can only be discussed in the context of a specific implementation, while the latter should be common among all DSMC implementations.

### *General DSMC Parameters*

In general, particles should not traverse more than about one cell per timestep, so $c_t$ should be less than one. Typical values are $0.3 < c_t < 1$. The ratio of the cell size to the mean-free path, $c_\lambda$, should be less than one. Many implementations ensure that this constraint is met by adaptively adjusting cell sizes appropriately. For such approaches, values for $c_\lambda$ are typically between 0.3 and 1.

In order to understand the convergence time of a simulation, it is important to consider the grid shape and boundary conditions. For a spherical grid with a uniform external boundary, information at the boundaries will quickly propagate throughout the domain. On the other hand, the simulation of a long curved tube with different boundary conditions at opposite ends will require a long time to converge. In order to find typical values for the number of acoustic periods required for convergence, $c_a$, a series of simulations was conducted. The duration of simulated time required for convergence was measured for different grids, densities, simulated volumes, and boundary conditions. Typical values were found to be in the range from 3 to 10.

The number of particles per cell, $c_p$, must be large enough that a reasonable number of collisions will take place in each cell. Using larger values of $c_p$ also reduces statistical scatter. On the other hand, both runtime and memory usage are proportional to $c_p$. For steady-state simulations, $c_p$ is typically chosen between 3 to 10. Some advanced statistical techniques have been used to produce reasonably accurate results for as few as one particle per cell [15].

The parameter $c_v$ represents the ratio of the typical cell dimension to the cube root of the cell volume and is primarily grid-dependent. For typical tetrahedral grid cells, $c_v \approx 2$, while hexahedral cells have slightly smaller values of $c_v$. For skewed grid cells, $c_v$ can be arbitrarily large. The ratio of the domain length to the cube root of the domain volume, $c_L$, is effectively the aspect ratio of the domain and may take values between 1 and 5 for representative simulations.

The choice of the number of mean-free paths that must be simulated, $c_d$, is largely problem-specific, but representative simulations may use values between 10 and 1000. Similarly, the choice of $c_\tau$, the fraction of the oscillation period over which the flow is

**TABLE III**
**General DSMC Parameters**

| Parameter | Description | Typical values |
|---|---|---|
| $c_t$ | Fraction of typical cell length traveled by typical particle in one timestep | 0.3–1 |
| $c_\lambda$ | Ratio of cell length to local mean free path, or minimum local Knudsen number | 0.3–1 |
| $c_a$ | Acoustic periods required for convergence | 3–10 |
| $c_p$ | Ratio of particles to cells | 3–10 |
| $c_v$ | Ratio of cell length to cube root of cell volume | 1–5 |
| $c_L$ | Ratio of domain width to cube root of domain volume | 1–5 |
| $c_d$ | Number of mean free paths to be simulated for external flow | 10–1000 |
| $c_\tau$ | Fraction of the oscillation period over which samples can be averaged | 0.01–0.3 |

considered unchanging, is problem-specific. For a sinusoidal oscillation, however, $c_\tau = 0.1$ is a reasonable approximation, providing 10 separate results for each oscillation period.

The implementation-independent parameters are summarized in Table III.

*Implementation-Specific Parameters*

The parameters $T_t$, $T_c$, $M_p$, and $M_c$, are both implementation- and architecture-specific. For illustrative purposes, typical values were obtained for a DSMC implementation, *Hawk*, designed for the simulation of neutral flow in plasma reactors, and for spacecraft reentry calculations [27]. Other DSMC implementations will have different associated constants, but they must obey the same dependences on the physical parameters.

Several tests were conducted on a Silicon Graphics Power Challenge with 75-MHz R8000 processors. Test cases included neutral flow in plasma reactors, hypersonic reentry flows, and uniform thermal relaxation tests. Implementation-specific parameters did not vary significantly between the different test cases. In order to measure transport time, simulations were conducted with the collision phase disabled. Similarly, collision time was measured on simulations with particle transport disabled. A value of $c_p = 10$ was used, and the measured values were $T_t = 34$ $\mu$s and $T_c = 40.0$ $\mu$s. Note that a larger value of $c_p$ will increase the number of particles being simulated for the same amount of per-cell overhead and result in smaller values for $T_t$ and $T_c$. The variable soft sphere (VSS) collision model was employed. Using a simpler model, such as variable hard sphere (VHS) or hard sphere (HS) simply reduces the amount of computation required per collision, $T_c$.

The overhead memory, $M_0$, was estimated for *Hawk* by running a simulation with only 12 cells and with no particles. On the SGI Power Challenge, this value was found to be approximately $M_0 = 2.97$ MB. The memory usage per particle has a lower bound of six floating point values, three for position and three for velocity (in three dimensions). Most implementations, however, use additional storage space for storing additional per-particle data structures that help to reduce runtime. The particle memory usage in *Hawk* was estimated by running simulations with varying numbers of particles and recording the memory usage reported by the operating system, then subtracting the overhead memory

**TABLE IV**
**Implementation-Specific Parameters**

| Parameter | Description | Typical values |
|---|---|---|
| $T_t$ | Time required to move one particle for one timestep | 34 $\mu$s |
| $T_c$ | Time required to perform one collision | 40.0 $\mu$s |
| $M_0$ | Memory required for overhead | 2.96 MB |
| $M_p$ | Memory required for a particle | 55 B |
| $M_c$ | Memory required for a cell | 1482 B |

$M_0$ and dividing by the number of particles. This yielded an an approximate value, $M_p = 55$ bytes. It must also be noted that simulations using more sophisticated chemistry models may require additional memory to store, for example, internal energy or species information.

Per-cell memory requirements are likely to vary more between DSMC implementations. In *Hawk*, each cell stores several values for unstructured grid information, a pointer to a linked list of particles, local information used for collisions, and macroscopic parameter information. Many of these data structures are irregular and dynamic in nature, with sizes that depend on the nature of the problem and may even change during a computation. For the relatively simple computations discussed here, cell sizes were typically around $M_c = 1482$ bytes.

Table IV summarizes the implementation-dependent parameters.

## 6. PREDICTIVE MODELING

In order to illustrate the application of the model to the prediction of actual simulation requirements, a series of internal flow simulations are considered. In order to assess the accuracy of the performance prediction model, *Hawk* simulations were conducted on five box grids, each with a different number of cells. Execution time per timestep and memory usage were measured. The execution time per timestep was predicted using Eq. (16) and the parameters in Section 5.

Figure 3a plots predicted and measured step times as functions of the quantity $n^3\sigma^3V$. For each simulation, the predicted step time is within 8% of the measured step time. The differences are largest for the small grids (low values of $n^3\sigma^3V$), which can be attributed to the effects of computational overhead and setup time. The linear dependence of step time on $n^3\sigma^3V$ is clearly demonstrated by this experiment.

Figure 3b shows the predicted and measured memory requirements for the same simulations. For the larger simulations, memory usage is proportional to the quantity $n^3\sigma^3V$, while for small simulations, the overhead memory, $M_0$, is the dominant term. These results show excellent agreement between predictions and measurements. The difference between predictions and measurements is consistently less than 4%.

In addition to the internal flow configurations considered in this section, the authors have conducted external flow simulations, including hypersonic reentry calculations with gas mixtures and internal degrees of freedom, that are consistent with the performance prediction model. The model has also been applied by Ivanov *et al.* to simulations of high-altitude capsule aerodynamics with real gas effects, using a completely different DSMC implementation [19]. Using suitable implementation-specific parameters, their data agree well with the model [20].

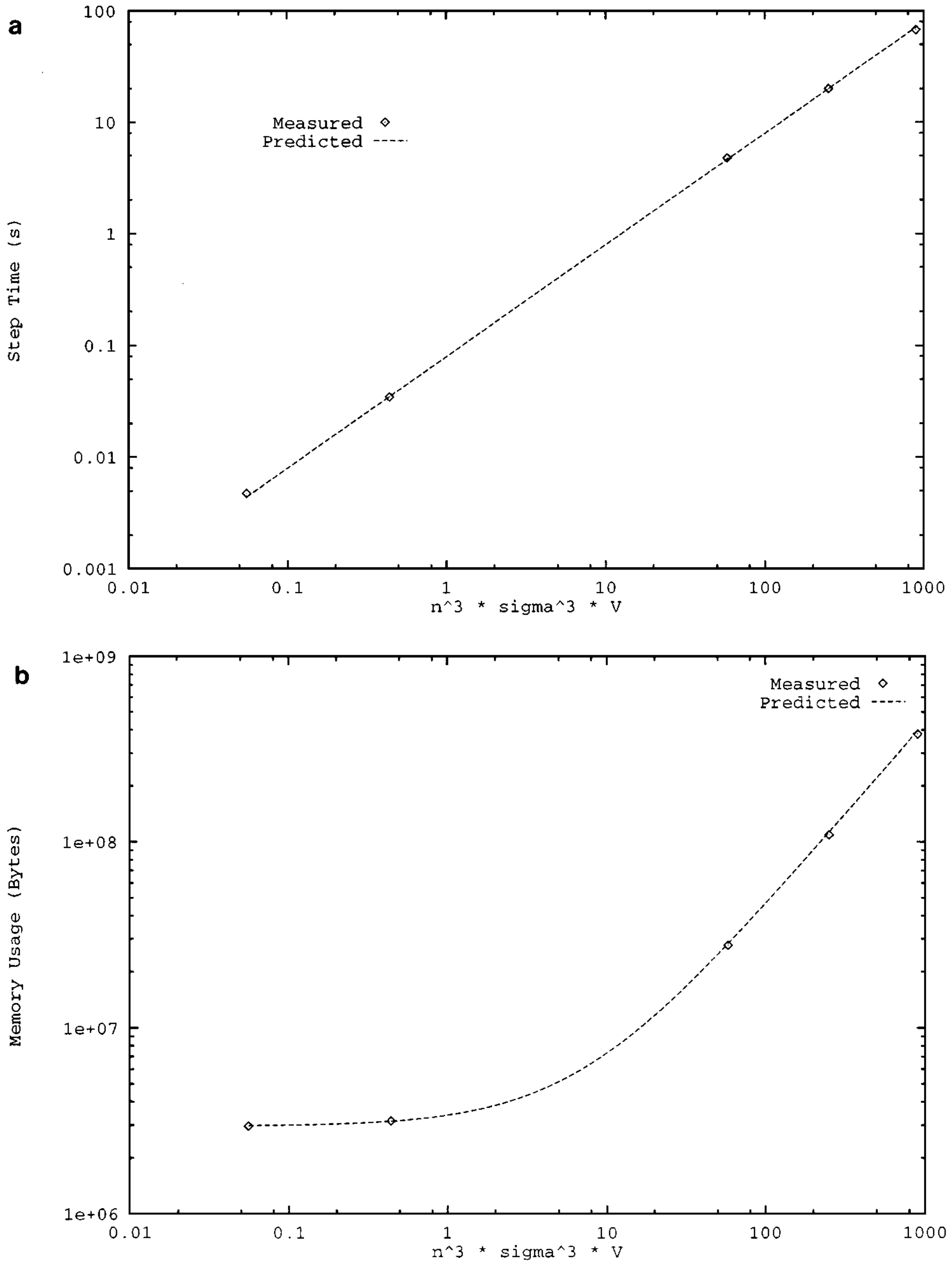

**FIG. 3.** Predicted and measured step time (left) and memory usage (right) as functions of physical parameters.

## 7. LARGE-SCALE SIMULATIONS

While the preceding experiments were performed on simple box grids, the analysis still holds for complex three-dimensional geometries. As an example of realistic simulations of industrial relevance, argon simulations of a plasma reactor, the Gaseous Electronics Conference (GEC) reference cell, were considered. A picture of this reactor, and a typical computational grid used to model it, are shown in Fig. 4. This reactor has a complex

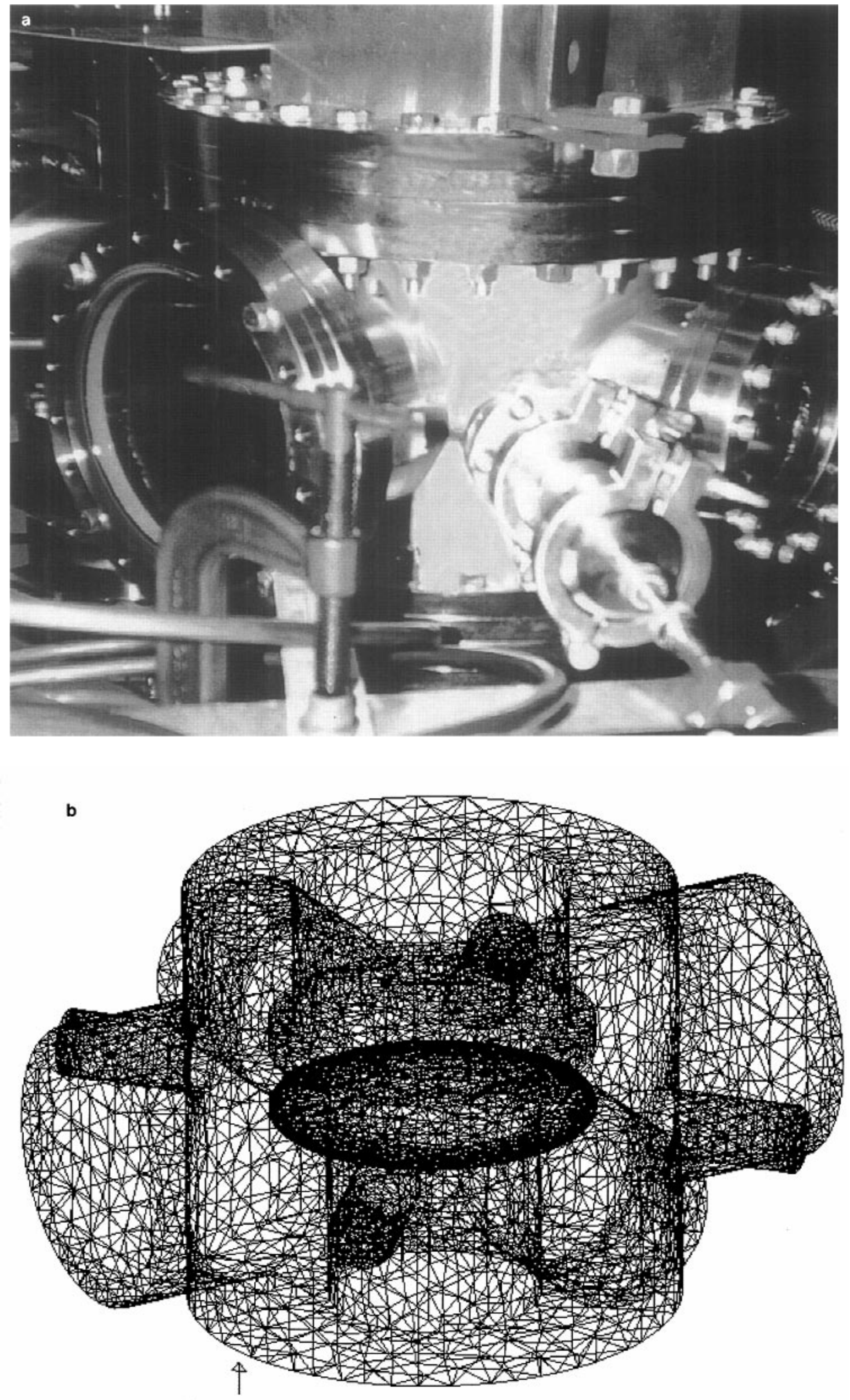

**FIG. 4.** The Gaseous Electronics Conference (GEC) reference cell reactor (left) and a computational grid used to represent it (right).

**TABLE V**
**GEC Simulation Predictions**

| Press. (Pa) | Press. (m Torr) | Density ($m^{-3}$) | Cells | Particles | $T_{one}$ (s) | $T_{conv}$ (s) | Mem (B) |
|---|---|---|---|---|---|---|---|
| 0.291 | 2.19 | $7.0 \times 10^{19}$ | $1.4 \times 10^{5}$ | $1.4 \times 10^{6}$ | 51.7 | $4.0 \times 10^{3}$ | $2.88 \times 10^{8}$ |
| 2.66 | 20 | $6.4 \times 10^{20}$ | $1.1 \times 10^{8}$ | $1.1 \times 10^{9}$ | $3.9 \times 10^{4}$ | $2.8 \times 10^{7}$ | $2.17 \times 10^{11}$ |
| 6.65 | 50 | $1.61 \times 10^{21}$ | $1.68 \times 10^{9}$ | $1.68 \times 10^{10}$ | $6.2 \times 10^{5}$ | $1.1 \times 10^{9}$ | $3.42 \times 10^{12}$ |
| 13.3 | 100 | $3.21 \times 10^{21}$ | $1.3 \times 10^{10}$ | $1.3 \times 10^{11}$ | $4.9 \times 10^{6}$ | $1.8 \times 10^{10}$ | $2.71 \times 10^{13}$ |

three-dimensional geometry with a volume of 0.013 $m^2$ and typically operates at a temperature of 300 K.

Using the model and constants developed above, runtime and storage requirements can be predicted for simulations of the GEC cell at several densities, or pressures. Table V lists predictions for the number of cells, number of particles, timestep duration, convergence time, and memory usage, for three different operating pressures. These values were obtained using the machine-specific parameters for the 75-MHz R8000 SGI Power Challenge.

In order to assess the applicability of the model to realistic three-dimensional geometries, the first case, at 0.291 Pa, was configured and simulated on an SGI Power Challenge. Using the model and parameters above, memory usage for this simulation was predicted to within 3%. Because the model does not take into account the additional cost of moving particles in the high grid-density regions, the model underpredicted the timestep time by about 25%. In general, the model can be expected to provide an accurate estimate of memory usage, and a reasonable lower bound for simulation time, for realistic simulations.

For the other simulations listed in Table V, the higher operating pressure results in vastly larger computational costs, both in terms of simulation time and storage requirements. For a simulation at 2.66 Pa to be conducted to the same accuracy, 22 GB of RAM would be required, and the convergence portion of the simulation would take 327 days. With a 512-processor machine, assuming 70% utilization, this simulation could be completed in about 22 h, using 43 MB per processor.

A simulation at 6.65 Pa would require 35 years on a single-processor machine with 3 TB RAM. On a machine with 1024 R8000 processors with 3 GB RAM each, such a simulation could be completed in 18 days at 70% utilization. For the 13.3 Pa case, a single SGI Power Challenge would require 27 TB of RAM, and convergence would take 570 years. On a 8192-processor Power Challenge, assuming 70% utilization, this simulation might be possible in 36 days, using 3.3 GB RAM per processor.

## 8. CONCLUSION

The results of this work show that the runtime and memory requirements can be accurately predicted on the basis of physical properties and machine-specific parameters. The algorithm is fundamentally polynomial in the physical parameters, and the degree of the polynomial can range from 0 to 4. When considering the applicability of the DSMC method to a specific problem, it is essential to consider the runtime and storage requirements for the simulation. For certain high-density or large-volume problems, these requirements may be prohibitive. By comparing the requirements of the DSMC method with the requirements of other methods, it is possible to determine the best approach for each specific problem. It is also possible to predict how chances in physical parameters will affect runtime and storage requirements and, thereby, to determine bounds on the class of problems that can be solved will the DSMC technique, given finite computational resources.

## ACKNOWLEDGMENTS

The information contained herein does not necessarily reflect the position or policy of the government of the United States, and no official endorsement should be inferred. The author would like to gratefully acknowledge the contributions of Maura Benton, Evan Cohn, Sergey Gimelshein, Ross Harvey, Mikhail Ivanov, Vincent McKoy, Bradley Nelson, Sadasivan Shankar, Stephen Taylor, and Jerrell Watts.

## REFERENCES

1. J. Austin and D. Goldstein, Direct numerical simulation of low-density atmospheric flow on io, *Bull. Am. Phys. Soc. Ser. II* **39**(9) (1994).
2. T. Bartel, Low density gas modeling in the microelectronics industry, in *Rarefied Gas Dynamics 19*, Vol. 1 (Oxford Univ. Press, Oxford, 1995).
3. G. Bird, Breakdown of translational and rotational equilibrium in gaseous expansions, *AIAA J.* **8**(11) (1970).
4. G. Bird, Direct simulation of the Boltzmann equation, *Phys. Fluids* **13**(11) (1970).
5. G. Bird, *Molecular Gas Dynamics* (Oxford Univ. Press, Oxford, 1976).
6. G. Bird, Simulation of multi-dimensional and chemically reacting flows, in *Rarefied Gas Dynamics* (Oxford Univ. Press, Oxford, 1979).
7. G. Bird, *Molecular Gas Dynamics and the Direct Simulation of Gas Flows* (Clarendon Press, Oxford, 1994).
8. J. F. Bourgat, P. Le Tallec, and M. D. Tidriri, Coupling Boltzmann and Navier–Stokes equations by friction, *J. Comput. Phys.* **127**, 227 (1996).
9. I. Boyd, Analysis of vibrational-translational energy transfer using the direct simulation Monte Carlo method, *Phys. Fluids* **3**(7) (1991).
10. I. Boyd, G. Pham-Van-Diep, and E. Muntz, Monte Carlo computation of nonequilibrium flow in a hypersonic iodine wind tunnel, *AIAA J.* **32**(5) (1994).
11. C. Cercignani, *Theory and Application of the Boltzmann Equation* (Scottish Academic Press, Edinburgh/London, 1975).
12. S. Chapman and T. G. Cowling, *The Mathematical Theory of Nonuniform Gases* (Cambridge Univ. Press, New York, 1952).
13. S. Gimelshein, G. Markelov, and M. Rieffel, *Collision Models in the Hawk DSMC Implementation,* Caltech Technical Report CS-96-16, 1996.
14. J. K. Haviland and M. L. Levin, Application of Monte Carlo method to heat transfer in rarefied gases, *Phys. Fluids* **5**, 1399 (1962).
15. M. S. Ivanov and S. V. Rogasinsky, Theoretical analysis of traditional and modern schemes of the DSMC method, invited paper in *Rarefied Gas Dynamics, 1991*.
16. M. Ivanov, S. Antonov, S. Gimelshein, and A. Kashkovsky, Computational tools for rarefied aerodynamics, in *Proc. XVII Intern. Symp. on Rarefied Gas Dynamics, Vancouver, Canada, 1994.*
17. M. Ivanov, S. Gimelshein, and A. Beylich, Hysteresis effect in stationary reflection of shock waves, *Phys. Fluids* **7**(4) (1995).
18. M. Ivanov and S. Gimelshein, Computational hypersonic rarefied flows, *Annu. Rev. Fluid Mech.* **30**, 469 (1998).
19. M. Ivanov, G. Markelov, S. Gimelshein, L. Mishina, A. Krylov, and N. Grechko, High-altitude capsule aerodynamics with real gas effects, *J. Spacecraft Rockets* **35**(1) (1998).
20. M. Ivanov and S. Gimelshein, private communication, 1998.
21. M. N. Kogan, *Rarefied Gas Dynamics* (Plenum, New York, 1969).

22. K. Koura and H. Matsumoto, Variable soft sphere molecular model for air species, *Phys. Fluids* **4**(5) (1992).

23. P. Marriott and T. Bartel, Comparisons of DSMC flow field predictions using different models for energy exchange and chemical reaction probability, in *Rarefied Gas Dynamics 19*, Vol. 1 (Oxford Univ. Press, Oxford, 1995).

24. E. P. Muntz, Rarefied gas dynamics, *Ann. Rev. Fluid Mech.* **21**, 387 (1989).

25. K. Nanbu and Y. Watanabe, Relaxation rates of inverse-power and rigid-sphere molecules, *Rep. Ins. High Speed Mach.* **43**, 334 (1981).

26. K. Nanbu and S. Uchida, Application of particle simulation to plasma processing, in *Rarefied Gas Dynamics 19*, Vol. 1 (Oxford Univ. Press, Oxford, 1995).

27. M. Rieffel, Concurrent simulations of plasma reactors for VLSI manufacturing, Masters Thesis, Caltech CS-95-012, 1995.

28. M. Rieffel, S. Taylor, and J. Watts, Concurrent simulation of plasma reactors, in *Proceedings of High Performance Computing '97*, p. 163.

29. M. Rieffel, S. Taylor, and S. Shankar, Reactor simulations for semiconductor manufacturing, in *Proceedings of High Performance Computing '98*.

30. M. Rieffel, Performance modeling for concurrent particle simulations, Doctoral thesis, Computer Science Department, Caltech, 1998.

31. E. M. Shakhov, *Method of Studying the Rarefied Gas Motion* (Nauka, Moscow, 1974). [Russian]

32. S. Taylor, J. Watts, M. Rieffel, and M. Palmer, The concurrent graph: Basic technology for irregular problems, *IEEE Parallel Distrib. Technol.* **4**(2) (1996).

33. D. Wadsworth, Development and application of a three-dimensional parallel direct simulation Monte Carlo code for materials processing problems, in *Parallel CFD '95*, Pasadena.

34. X. Zhong and K. Koura, Comparison of solutions of the Burnett equations, Navier–Stokes equations, and DSMC for Couette flow, in *Rarefied Gas Dynamics 19*, Vol. 1 (Oxford Univ. Press, Oxford, 1995).