Efficient Three-Dimensional Direct Simulation Monte Carlo for Complex Geometry Problems

D. F. G. Rault

NASA Langley Research Center, Hampton, Virginia

Reprinted from *Rarefied Gas Dynamics: Theory and Simulations*, edited by Bernie D. Shizgal and David P. Weaver, Vol. 159 of *Progress in Astronautics and Aeronautics*, AIAA, Washington, DC, ISBN 1-56347-080-2.

Efficient Three-Dimensional Direct Simulation Monte Carlo for Complex Geometry Problems

Didier F. G. Rault*
NASA Langley Research Center, Hampton, Virginia 23681

Abstract

The simulation of flowfields in the transition flow regime is notoriously difficult with high demands on computer resources (CPU time and storage) and user expertise/labor. This paper describes a new, efficient code which has been developed to simulate high Knudsen number flowfields in three dimensions about bodies of arbitrarily complex geometry. The algorithm has been tested over a wide range of conditions, from free molecular to near-continuum flow regimes, for slender and blunt bodies, for re-entry vehicles and spacecraft. A series of validation tests have been conducted using both wind-tunnel measurements and flight data.

Introduction

Early three-dimensional direct simulation Monte Carlo (DSMC) codes, such as Bird's G3 code,[1] were very difficult to set up-sometimes requiring several months of user's time-long to run, and difficult to diagnose. To be useful to the engineering community, however, a flow simulation code should be easy to set up, easy to use, and fast as well as accurate. The code we are developing attempts to meet these needs. The basic features of the code are presented in the first section. The code has already been used to simulate flowfields around a series of slender and blunt re-entry vehicles and spacecraft, and some results are presented and discussed. A special preprocessor, which was designed to allow the user to define and input complex vehicle surfaces, is then described.

*Research Engineer, Space Systems Div.

The DSMC code is written as a set of independent modules, each of which can be customized for special application or CPU. The methods and techniques used to optimize the code for vector processors are highlighted. A solution adaptation routine, which was developed to periodically tune up the code parameters (grid, time step, and molecular weight factors) during a simulation, is then described. The last section discusses the performance of the code. It is shown that, when fully optimized, our present code has a computational efficiency (expressed in μsec/time step/molecule) comparable with the one reported by Feiereisen and McDonald[13] in the case of the flow simulation around the Aerobraking Flight Experiment (AFE) vehicle at 100 km. The code runs on scalar and vector computers and has recently been modified to run on parallel processors.

Code Characteristic Features

The code has the following features:

i) It incorporates full implementation of the DSMC methodology developed over the past 30 years by Bird[3,4] and others (decoupling of molecular motion and collision, variable-hard-sphere (VHS) model, no time counter method (NTC), Larsen-Borgnakke inelastic collision model, and Bird's chemical reaction model).

ii) The code uses two-level Cartesian grid, as described in Refs. 5-8. Cartesian grids are known to be specially suited to particle simulations. A pure Cartesian grid, however, cannot usually be used efficiently in flow simulation codes since grid resolution must be tied to local physical lengths such as the collision mean free path or gradient scales. Figure 1 illustrates the computational grid used for the AFE flowfield simulation.

iii) An optimized algorithm has been specially developed to exploit the full capability of vector processors. Molecules are processed in a group, rather than one by one, and each computational loop is optimized for vectorization. Nonvectorizable sections of the algorithm are isolated and treated separately.

iv) Fast preprocessors define the vehicle surface, lay out the initial computational grid, and establish conditions at the computational domain boundaries.

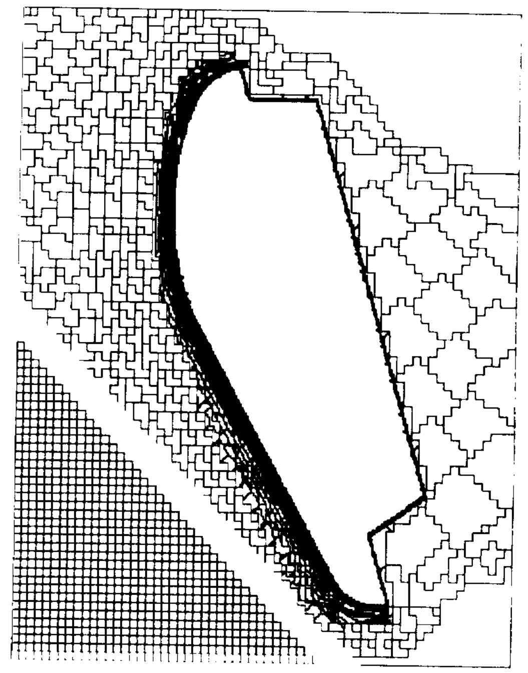

Fig. 1 Computational grid for AFE simulation at 100-km altitude.

v) Interactive graphical diagnosis monitor the code and verify that DSMC criteria are being met.

vi) Interactive graphical postprocessors are used, which compute the macroscopic thermodynamic and flow properties from the simulated molecular quantities and provide an interface to commercial, three-dimensional flowfield visualization software, such as TecPlot from AMTEC Corporation.

Code Applications

The code has been used to simulate flowfields around several vehicles:

i) The first studies were made on a 10-cm-long delta wing at a Knudsen number of 0.016, for which wind-tunnel data are available (Ref. 6 and Fig. 2), and a 0.7-cm-long delta wing

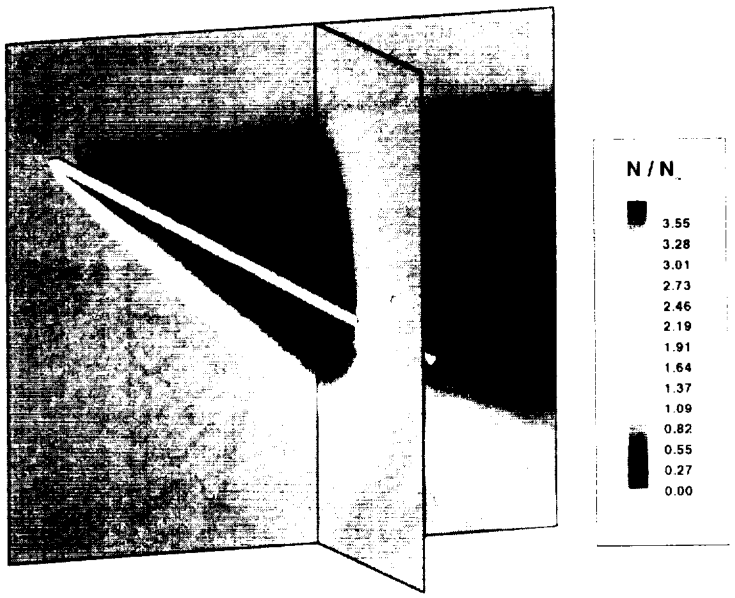

Fig. 2 Density contours around 10-cm delta wing at 30-deg incidence (Ref. 6).

at a Knudsen number of 0.39, for which freejet data are available.[7] These simulations showed the code to be accurate in predicting aerodynamic forces and heat transfer rates.

ii) A flowfield was simulated around the Space Shuttle at 120-km altitude and 40-deg incidence, for which flight data are available (Fig. 3). Lift-to-drag ratios computed from 160 km to 120 km matched measured values.

iii) A flowfield was simulated around a ramp compression corner, for which wind-tunnel data are available (Ref. 9 and Fig. 4). The simulated flow structure and separation characteristics are similar to those deduced from oil flow patterns.

iv) The code was used on a 5-m-long viscous optimized waverider in the altitude range 97-145 km.[8]

v) The code was tested on the Aeroassist Flight Experiment (AFE) vehicle at 100-km altitude (Fig. 5). This simulation produced aerodynamic characteristics similar to those computed by Celenligil (Ref. 1 and Table 1).

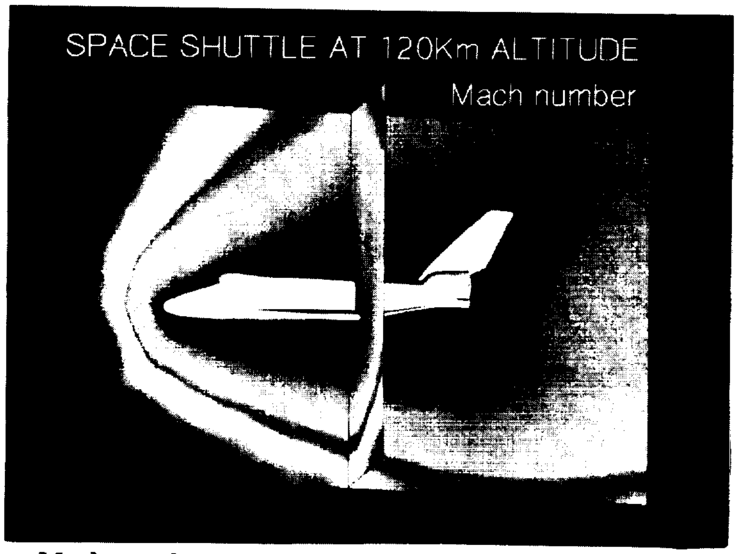

Fig. 3 Mach number contours around Shuttle at 40-deg incidence, 120-km altitude.

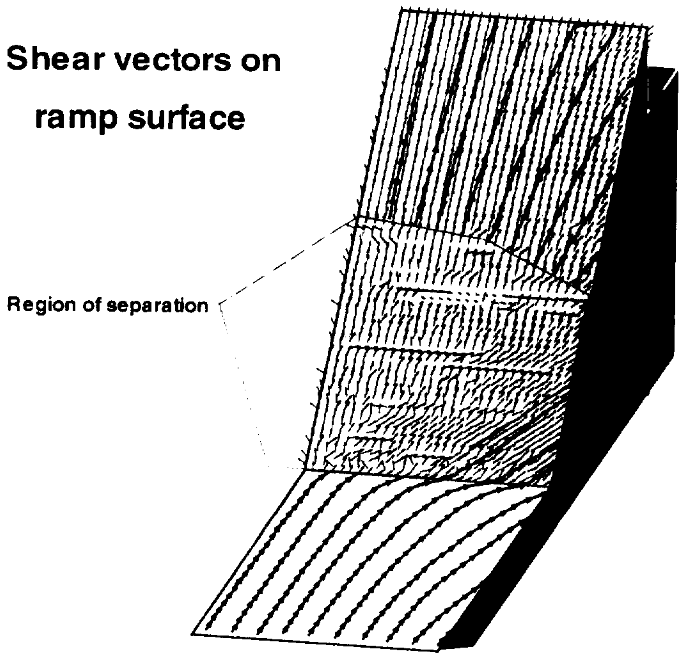

Fig. 4 Flow field structure on compression corner (Ref. 9).

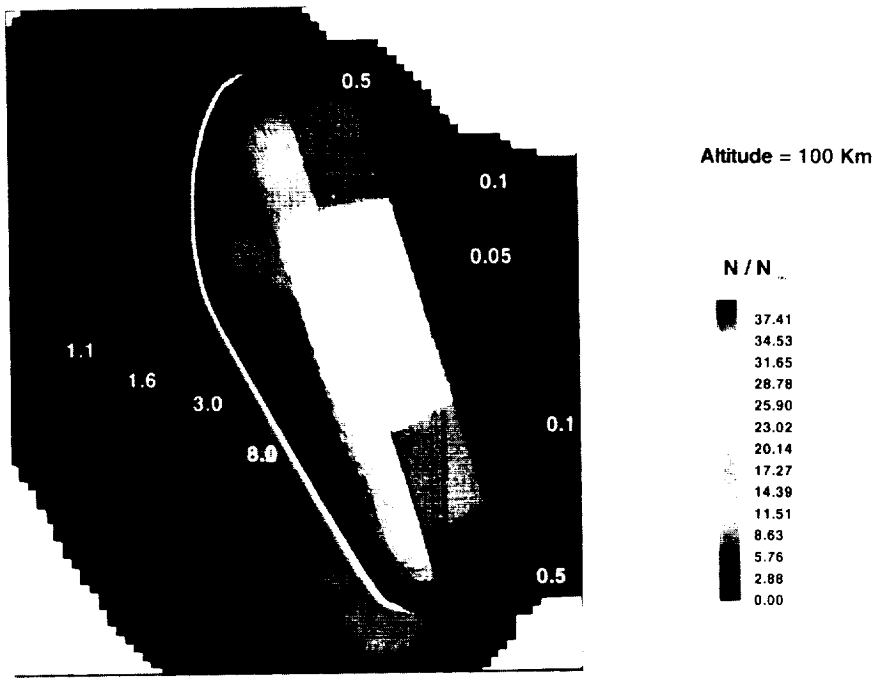

Fig. 5 Density contours around AFE at 100-km altitude.

Table 1 Aerodynamic characteristics of AFE at 100 km

Coefficients	Present Work	Celenligil (Ref. 1)	Modified Newtonian
Lift	0.305	0.318	0.360
Drag	1.56	1.50	1.32
Lift to Drag	0.195	0.212	0.272
Heat Transfer	0.362	--	--

vi) The code was used to study the aerodynamic characteristics of the Viking aeroshell used in Mars landings (Fig. 6).

vii) The code was tested on the Magellan spacecraft, presently orbiting Venus and mapping the planet's surface (Fig. 7). This simulation showed the capability of the code to readily handle bodies with complex geometry.

viii) The gaseous environment around the Upper Atmospheric Research Satellite (UARS), was simulated with the code and contamination deposition on reflective and refractive optics was quantified (Fig. 8).

Graphical Preprocessor

Complex geometries can readily be defined with our CAD-like preprocessor. The preprocessor relies on the fact that even a very complex three-dimensional body geometry can be decomposed into a series of simple primitives such as spheres, cones, cylinders, planes, ellipsoids, etc., or parts thereof. The geometric properties of each primitive subelement constituting the vehicle surface are

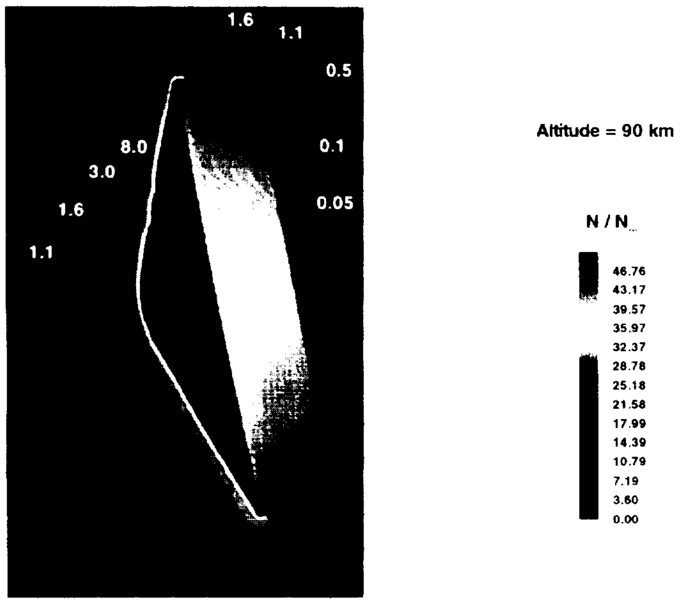

Fig. 6 Density contours around the Viking aeroshell at 90-km altitude.

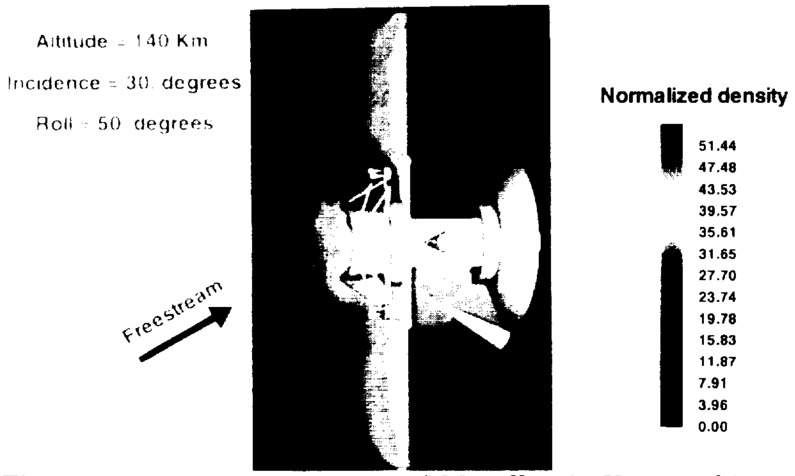

Fig. 7 Density contours around Magellan in Venus orbit at 140-km altitude.

Table 2 Library of geometric primitives used in Magellan geometry definition

Primitives	Definitions
SphereA	Complete sphere (radius, center)
SphereC	Spherical arc (point, direction , depth)
SphereD	Spherical section (point, direction, depth, 2 radii)
SphereE	Hemisphere (point, direction, radius)
CylinderA	Straight cylinder (2 point, radius)
CylinderB	Straight cylinder (point, radius, direction, length)
CylinderC	Skewed cylinder (2 points, radius, 2 directions)
ConeA	Straight cone (half angle, apex, direction, length)
CodeB	Straight cone (apex, length, exit radius, direction)
CodeC	Cone frustum (point, half angle, radius, length direction)
CodeD	Cone frustum (point, 2 radii, length, direction)
PlateA	Triangular plate (3 points)
PlateB	4 sided plate (4 points)
Disk	Circular disk (center, direction, radius)
RingA	Ring between 2 circles (center, 2 radii, direction)
RingB	Ring between circle and rectangle (center, radius, direction, height, width)

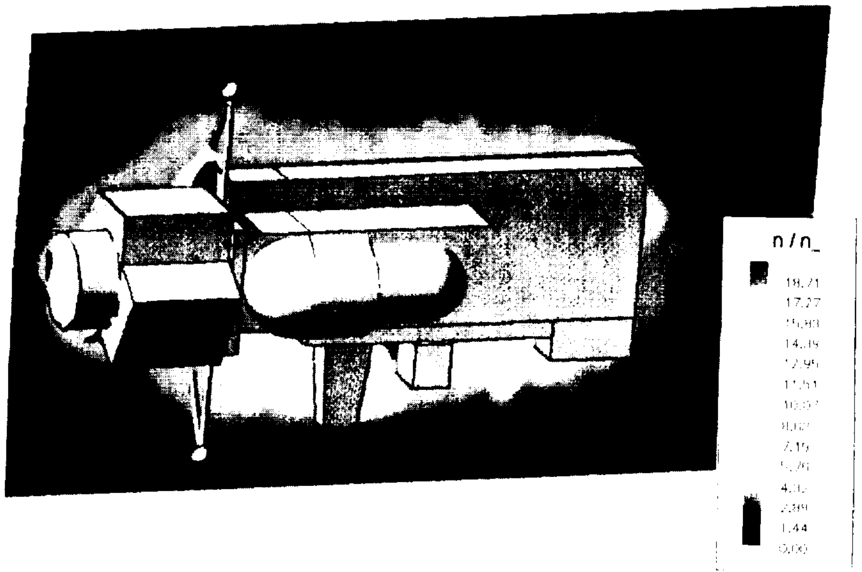

Fig. 8 Density contours around UARS satellite.

entered into the preprocessor through an editable spreadsheet. The spreadsheet is broken down into records, each one corresponding to a primitive subelement, and each record is made up of a series of fields, each one describing the nature of the subelement, its dimensions and coordinates. Vehicle geometries have been defined using 1 to 350 subelements selected among a library of 16 basic primitives, the list of which is shown in Table 2. This library of primitives can be further expanded and edited by the user. The preprocessor reads the spreadsheet data file and creates a binary graphical file in TecPlot format. The body geometry thus generated can be readily visualized in three dimensions and analyzed using the several TecPlot utilities (Rotation, Zoom, Shading, etc.). By interactively editing the spreadsheet data file and graphically examining the generated geometry, it is possible to construct and reproduce the geometry of very complex objects, as illustrated in Fig. 9 which shows the surface definition of the UARS satellite. Upon completion, the preprocessor generates a geometry file in the format required by the DSMC code.

Code Modularization

The code is composed of a series of distinct and independent routines, as shown in Fig. 10. Alternate routines can be constructed

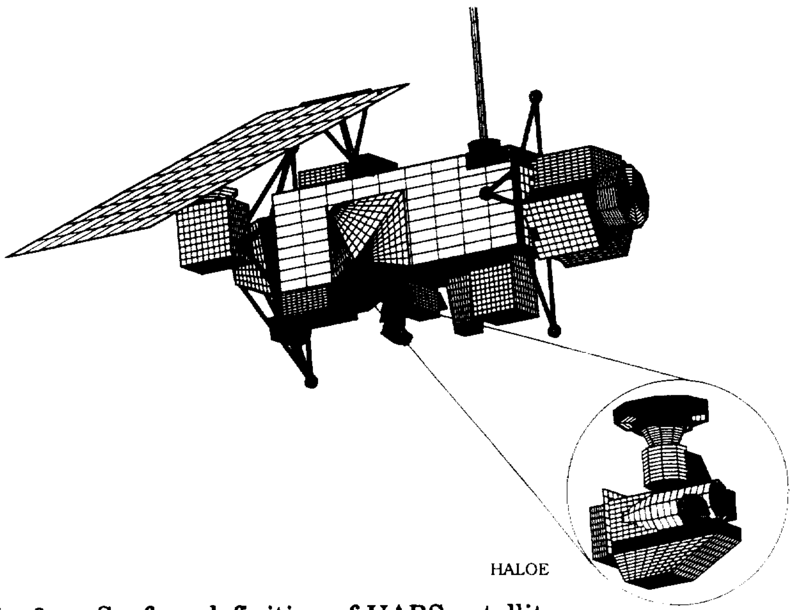

Fig. 9 Surface definition of UARS satellite.

and substituted to treat specific problems. The Initialization routine is used at the beginning of the simulation and distributes molecules uniformly throughout the computational domain. The Sample Reset routine is used to reset all simulation counters to zero. At each time step, molecules are inserted either at the computational domain boundaries (Insert routine for freestream or interface with parallel processors) or on the body surface (Outgassing routine for outgassing and venting). The Move routine translates molecules and computes interaction with body surface and computational domain boundaries. This routine is further described below. The Indexing routine orders the ensemble of simulated molecules according to the cell in which they reside. This routine is used in noncollisionless flows to allow the matching of closest collision partners. The Collision routine identifies actual collision partners and computes the postcollision characteristics of the molecules. This routine is further described below. The Chemistry and Inelastic subroutines determine whether a collision results in a chemical reaction and/or internal energy exchange between molecules. In the course of a simulation, simulated molecules may leave the computational domain or disappear through chemical reactions. Such molecules are earmarked for removal and purged in the Molecule Removal routine. The Time History routine computes the macroscopic properties (drag, lift, moment coefficients, momentum and energy fluxes[8]) and periodically

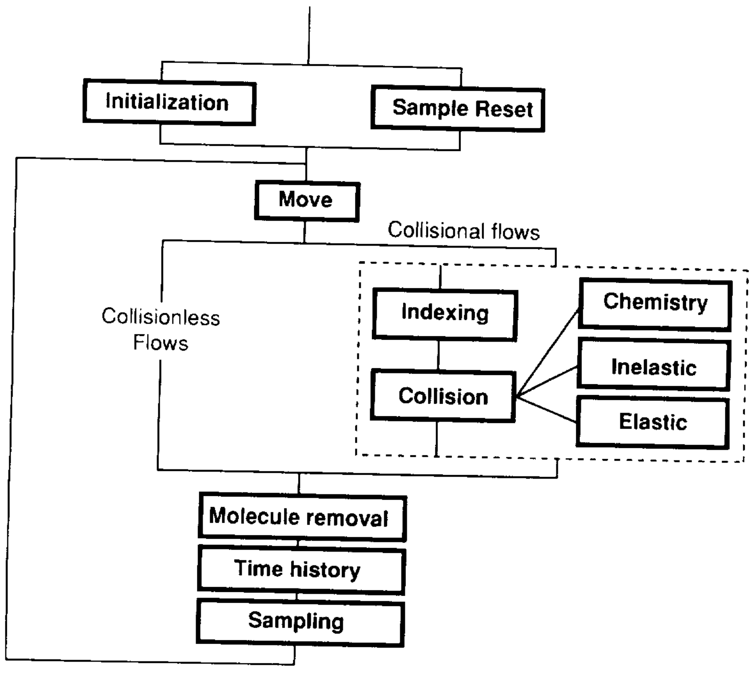

Fig. 10 DSMC code main modules.

writes a record in a TecPlot formatted file. This routine allows the user to control and diagnose the code during the transient phase of the simulation and ascertain its convergence characteristics. The Sampling routine increments all microscopic counters in each cell to establish the zero, first and second moments of the molecular velocity distribution function.

Code Vectorization

Each routine has been optimized for vector computers. Only the Move and Collision routines, however, are described here. In the Move routine, the spatial domain is normalized as

$$x_i = X_i / DS_i + 1 \tag{1}$$

where X_i is the ith position coordinate of the molecule in the body frame of reference, and DS_i is the size of the coarse Cartesian grid. The index of the coarse Cartesian grid cell in which the molecule resides is therefore:

$$N_i = \mathrm{int}(x_i) \qquad (2)$$

Figure 11 is a flowchart showing the main functions within the Move routine. Molecules can be moved in either of two ways: fast, for molecules located far enough away from the vehicle surface and slow, for other molecules.

The first step within the Move routine is to identify the molecules which can be moved with the fast module, translate molecules by V.dT and check for computational domain boundary intersection. Each cell of the coarse Cartesian grid is assigned a number, which is the distance to the nearest body surface element, and this number is used to check if molecules can be moved with the fast module. The remaining molecules must be tracked within the coarse Cartesian grid (CCG), and sometimes within the fine Cartesian grid (FCG). These molecules are entered into buffer arrays, the size of which depends on the computer, and is typically a few thousand. Within the buffer, molecules are arranged in four different groups: molecules which can be moved one CCG cell at a time, molecules

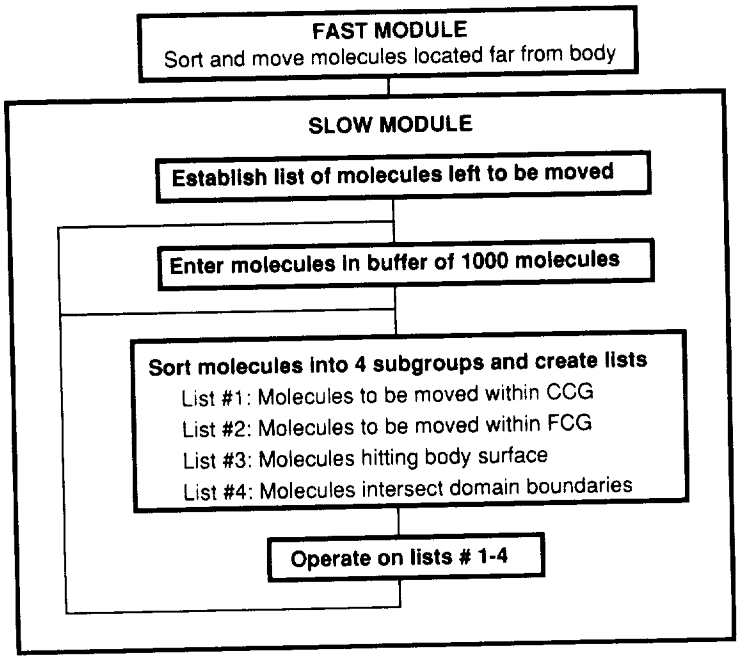
FAST MODULE
Sort and move molecules located far from body
SLOW MODULE
Establish list of molecules left to be moved
Enter molecules in buffer of 1000 molecules
Sort molecules into 4 subgroups and create lists
List #1: Molecules to be moved within CCG
List #2: Molecules to be moved within FCG
List #3: Molecules hitting body surface
List #4: Molecules intersect domain boundaries
Operate on lists # 1-4

Fig. 11 Flowchart for Move routine vectorization.

which must be moved one FCG at a time, molecules which hit the vehicle surface, and molecules which intersect the computational domain boundaries.

Molecules in each of the different groups are moved altogether to optimize vectorization. During a given time step, molecules may move from one group to another. Molecules which have finished their time step, are removed from the buffer and replaced by new molecules. The process is continued until all molecules have been moved for their allotted time step. All the functions have been vectorized, except the vehicle surface interaction, which is a relatively minor function since only a very small fraction of the molecules actually intersect the body surface during a time step.

In the case of blunt vehicles, an alternate routine was constructed which takes into account the fact that molecules have little probability of traversing the vehicle body during a time step. All the molecules are, therefore, moved with the fast module and only the molecules which are found to lay inside the body at the end of the time step are moved back and tracked with the slow module. As was observed during the study on the AFE vehicle, this routine can be considerably faster than the original Move routine.

The Collision routine is similarly broken down into a series of steps, as shown in Fig. 12, most of which have been vectorized. In the first step, the number of potential collisions in each cell is determined following the NTC method developed by Bird. A table of random numbers is then generated, which is used to match candidate collision partners within each cell. Table 3 shows an example of such a table. To reduce computational costs, the cells containing the same number of molecules use the same list of random numbers. Each pair of candidate collision partners is then examined for collision using Bird's VHS algorithm and lists of colliding molecules are established. For a monatomic gas, the post collision characteristics of each molecule is computed in the Elastic routine, whereas, for a polyatomic gas mixture, chemical reaction and internal energy exchange are first checked and lists are created and acted upon. The return loop shown in Fig. 12 is necessary since the NTC derived number of potential collisions in a cell may exceed the number of simulated molecules in that cell. This loop, therefore, allows for molecules to collide several times per time step.

Solution Adaptation

Early DSMC simulation codes were tuned up manually. For a given problem, the computational grid, time step, and molecular weight factor were initially set a priori and the simulation was

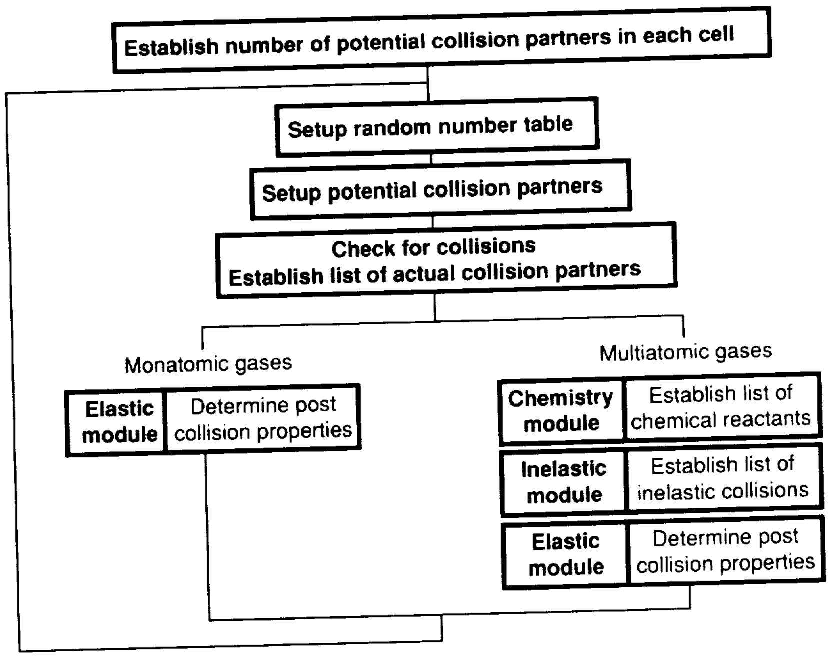
Establish number of potential collision partners in each cell
Setup random number table
Setup potential collision partners
Check for collisions
Establish list of actual collision partners
Monatomic gases
Multiatomic gases
Elastic module
Determine post collision properties
Chemistry module
Establish list of chemical reactants
Inelastic module
Establish list of inelastic collisions
Elastic module
Determine post collision properties

Fig. 12 Flowchart for Collision routine vectorization.

allowed to run for several CPU hours, after which the grid and simulation parameters were adjusted, and the simulation was restarted from time $t = 0$, disregarding the present state of the simulated molecule ensemble. Such manual iterative procedure is highly time consuming and is extremely difficult in three-dimensional simulations. A solution adaptation routine was therefore devised with the following functions:

i) Adjust the size of the computational domain by properly positioning the upstream and downstream boundaries.

Table 3 Random number table

Number of molecules in cell	2	3	4	5	...
List of Random Numbers	2 1	3 1 2	3 2 4 1	1 3 5 2 3	

The upstream boundary is set just in front of the "disturbed" region, at a location where the local mean free path exceeds the freestream mean free path by 5-10 percent. The downstream boundary is set just outside the "domain of influence" of the flowfield around the vehicle. When computing re-entry vehicle aerodynamics, the domain of influence can be viewed as the flow region which has an effect on the forces, moments, and heat transfer rates acting on the vehicle.

ii) Restructure the two background Cartesian grids, expanding/shrinking the region in which the fine grid must be used (where the mean free path is smaller/larger than the coarse grid cell size).

iii) Restructure the computational grid cell ensemble so that cells near the body are no larger than the local mean free path in the direction normal to the body, the density fluctuations within any given cell is small, and all cells contain about the same number of simulated molecules. This last requirement becomes important in the steady state phase of the simulation as each cell would, therefore, accumulate sampled molecules at the same rate, thus ensuring that random noise on the thermodynamic and aerodynamic properties dampens uniformly throughout the computational domain.

iv) Reset the time step, if necessary, to better meet the basic DSMC criteria that the time step must be smaller than the mean collision time and molecules move less than a cell length during a time step.

v) Reset the molecule weight factor so that there are on average two simulated molecules per "subcell," a subcell being defined as a cell of the coarse background Cartesian grid.

vi) Translate the vehicle within the computational domain to, for example, increase the distance between the vehicle and the upstream freestream boundary.

vii) Alter the value of a range of simulation parameters, such as the maximum number of simulated molecules, the number of time steps between printouts, etc.

viii) Change gas properties, such as adding chemistry or internal energy structure within molecules.

Table 4 Code performance comparison (AFE at 100 km)

	Present Code		G3	PSim
Processor	SPARC2	CRAY2	CRAY2	C RAY2
Setup time	2 weeks		6 months	Not available
Computational Efficiency (μsec/mol/time)	40	6	200	5
Number of Molecules	150000		75000	10^7
CPU hours for 1000 steps	1.7	0.25	4.2	13.9
Memory Requirement	35MBytes	8Mwords	4Mwords	Not available

One important feature of the solution adaptation routine is that the simulation is restarted using the ensemble of simulated molecules (characterized by position, velocity vectors, and species), which were present in the computational domain at the time the simulation was stopped.

Code Performance

The code performance was studied during our simulation of the flowfield around the Aerobrake Flight Experiment (AFE) vehicle at 100 km altitude. This vehicle has been investigated under similar conditions by Celenligil,[1] who used Bird's G3 code, and Feiereisen and McDonald,[2] who used the NASA Ames PSim code. Table 4 compares the performance of the three codes. Our present code is about 30 times faster than the G3 code on a per-molecule basis, but uses twice as many molecules and computer memory. The setup time for our code is very short, thanks to the graphical preprocessor and the solution adaptation routines. The computational efficiencies of PSim and our present code are comparable. Our simulation, however, included chemistry, which had not yet been implemented in PSim, and required 60 times fewer simulated molecules. Table 5 shows the relative CPU time associated with each routine and subroutines. The Move routine consumes about 1/3 of the total CPU time, whereas the Collision routines used about 50 percent of CPU time. This result is in contrast with earlier DSMC

Table 5 Code performance on CRAY-2 (AFE at 100 km)

Routines	Relative Timing	Subfunctions	Relative Timing Within Routine
Move	38%	Fast move	42%
		Slow move	50%
		Wall reflection	8%
Indexing	6%		
Collision	26%	Molecule pairing	25%
		Collision Check	63%
		Other	12%
Chemistry	7%		
Inelastic	7%		
Elastic	1%		
Sampling	7%		
Other	8%		

codes in which the computation of molecular motion was the most time consuming procedure, sometimes accounting for 90 percent of the total CPU time.

Conclusion

The three-dimensional DSMC code described above is intended to be an efficient code both in the sense of CPU time, and most importantly, user's time. The computational grid structure, CAD-like preprocessor, diagnosis, postprocessor, adaptation and the code modular structure have been shown to be very effective in reducing simulation times from months/year down to weeks. This, together with proven code accuracy, should allow for a greater and more widespread use of three-dimensional DSMC in engineering design and tradeoff studies for a wide range of applications from the free molecular to the near continuum flow regime, for slender and blunt reentry vehicles, for spacecraft, and general fundamental studies.

References

[1]Celenligil, M. C., Moss, J. N., and Blanchard, R. C., "Three Dimensional Flow Simulation About the AFE Vehicle in the Transitional Flow Regime," AIAA Paper 89-0245, Reno, Nevada, January 1989.

[2]Feiereisen, W. J., and McDonald, J. D., "Three-Dimensional Discrete Particle Simulation of an AOTV," AIAA Paper 89-1711, 1989, Buffalo, New York, 1989.

[3]Bird, G. A., Molecular Gas Dynamics, Clarendon Press, Oxford, England, U.K., 1976, pp. 118-132.

[4]Bird, G. A., "Monte Carlo Simulation in an Engineering Context," Progress in Astronautics and Aeronautics, Rarefied Gas Dynamics, Vol. 74, Pt. 1, edited by S. S. Fisher, New York 1981, pp. 239-255..

[5]Bird, G. A., "Application of the Direct Simulation Monte Carlo Method to the Full Shuttle Geometry," AIAA Paper 90-1962. Seattle, Washington, June 1990..

[6]Rault, D. F. G., "An Efficient DSMC Algorithm Applied to a Delta Wing," AIAA Paper 91-1316, Honolulu, Hawaii, June 1991.

[7]Rault, D. F. G., "Aerodynamic Performance of Delta Wings in the Hypersonic Rarefied Flow Regime, Second Workshop on Hypersonic Flows for Reentry Problems, (Antibes, France), April 1991.

[8]Rault, D. F. G., "Aerodynamic Characteristics of a Hypersonic Viscous Optimized Waverider at High Altitude," AIAA Paper 92-0306, Reno, Nevada, 1992.

[9]Moss, J. N., Rault, D. F. G., and Price, J. M., "DSMC Simulation of Hypersonic Viscous Interactions Including Separation," 18th Rarefied Gas Dynamics Symposium, (Vancouver, Canada), 1992.