JOURNAL OF COMPUTATIONAL PHYSICS 96, 411–427 (1991)

Vectorization of a Monte Carlo Simulation Scheme for Nonequilibrium Gas Dynamics

IAIN D. BOYD*

Eloret Institute, 3788 Fabian Way, Palo Alto, California 94303

Received January 10, 1990; revised April 16, 1990

The numerical performance of a Monte Carlo scheme used in the analysis of non-equilibrium gas dynamics has been greatly improved. This improvement is attained by careful implementation of the algorithm in order to take advantage of the vector hardware of super-computers. The performance of the modified implementation is demonstrated by application to three different flow problems. First, the one-dimensional standing shock wave is considered. Due to the relative simplicity of this example, it is shown that an adequate solution is obtained in a very small computational time. The second problem considered is the flow of an expanding gas through an axisymmetric nozzle. Lastly, the hypersonic flow of argon over a three-dimensional wedge is computed. This problem illustrates the increase in the number of molecules which may be employed in the simulation due to the improved performance of the algorithm. In fact, over 10 million particles are employed which is the largest number reported in the literature for the simulation scheme considered. © 1991 Academic Press, Inc.

1. INTRODUCTION

The direct simulation Monte Carlo method (DSMC) has become a popular numerical tool used in the solution of rarefied and nonequilibrium gas dynamics. This technique was developed by Bird [1] over a number of years to the extent that many complex physical phenomena may be modelled in a simulation. In this method, the large number of molecules in a real gas is simulated through a much smaller set of representative particles. The paths of these simulated molecules are traced out in physical space by decoupling motion from intermolecular collisions. The time step over which decoupling occurs must be smaller than the mean time between collisions. A computational grid is generated in physical space for the determination of local flow conditions. Each particle has three velocity components, up to three spatial coordinates, and perhaps internal energy associated with it. The collisions are treated statistically, with the probability of collision obtained from kinetic theory. The method obtains macroscopic flow quantities such as temperature and density by averaging over the microscopic values. Reviews of the status and importance of the method have recently been given by Bird [2] and Muntz [3].

* Mailing adress: NASA Ames Research Center, MS 230-2, Moffett Field, CA 94035.

0021-9991/91 $3.00

The use of this technique has become more widespread with the increased interest in rarefied gas dynamics which has been generated by modern aerospace projects such as hypersonic flight and interplanetary exploration. One of the main disadvantages of the DSMC technique has been its large computational penalty. An illustrative example is provided by the three-dimensional calculations of Celenligil *et al.* [4] which required 20 CPU hours on the CRAY-2 while employing 82,000 simulated molecules and 5000 cells. A new computationally efficient particle scheme has been developed by Baganoff and McDonald [5]. The substantial increase in numerical performance is attained by appropriate structuring of their algorithm on the vector architecture of the CRAY-2 computer. The purpose of this paper is to report upon the vectorization of the DSMC algorithm. The implementation alterations made to a typical DSMC code will be described. To highlight the improved numerical performance of the vectorized implementation three different flow problems are computed. Specifically, these are a standing shock wave, an axisymmetric nozzle flow, and a hypersonic, three-dimensional wedge flow.

2. The Scalar DSMC Algorithm

A particle method may be conveniently divided into seven individual tasks which are performed (not necessarily in order):

(i) generation of new molecules,
(ii) movement of molecules,
(iii) determination of molecule location,
(iv) interaction with boundary,
(v) indexing of molecules,
(vi) calculation of collisions,
(vii) sampling of molecular properties.

Each of these will now be discussed in more detail.

In the DSMC technique, molecules are contantly entering and leaving the computational domain. The properties of entering molecules are specified by the known boundary conditions. Due to the fact that the number of entering molecules is usually a small fraction of the total number in the entire flowfield, this aspect of the technique requires little computational effort. The movement of each particle through the distance given by the product of the velocity vector and the decoupling time step is a straightforward operation. Once a particle has been moved, its location in the computational grid must be determined. This may be achieved in a number of different ways. The method originally employed by Bird [1] involved making a check of the adjacent cells to discover whether the molecule has moved closest to that location. In a two-dimensional calculation this process requires the computation of nine distances and is clearly numerically intensive. One practical alternative is the use of a body-fitted coordinate scheme [6] of which the simplest form is uniform cell lengths [7]. After establishing the location of the particle, any interactions with the cell boundaries are imposed. Such conditions include reflection from a solid surface, removal from the simulation, or transfer to a different computational region. It is also generally found that the boundary conditions can be processed very efficiently. After the location of the particle has been finalized, the counter of the number of molecules in the current cell is updated.

Once all the molecules have undergone the previous operations, they must be indexed by cell location for the subsequent selection and calculation of intermolecular collisions. Each cell must be given a base index from which all other molecules in the cell can be reached through a cross reference list [1]. The algorithm for the tasks (ii)–(v) is listed in Fig. 1 for a one-dimensional problem. The cell

```
CCCC  X(I) : x coordinate of molecule I.
CCCC  U(I) : u velocity component of molecule I.
CCCC  NP(I) : cell index of molecule I.
CCCC  NCELL(J,1) : number of molecules in cell J.
CCCC  NCELL(J,2) : base index of cell J.
CCCC  LCR(I) : cross reference pointer for molecule I.
CCCC  NX : total number of cells.
CCCC  NTOT : total number of molecules.
CCCC  DELTAT : decoupling time step.
CCCC  TRUNCX : truncation factor for determination of cell index.
CCCC

CCCC  Set number of molecules in each cell equal to zero.
      DO 10 J=1,NX
   10   NCELL(J,1)=0

CCCC  Loop over all molecules.

      DO 11 I=1,NTOT

CCCC    Movement of molecules.
        X(I)=X(I)+U(I)*DELTAT

CCCC    Determination of cell index.
        NP(I)=1+X(I)*TRUNCX

CCCC    Compute any boundary interaction.
        CALL BOUNDARY(I)

CCCC    Update sum of molecules in current cell.
        NCELL(NP(I),1)=NCELL(NP(I),1)+1

   11 CONTINUE

CCCC  Process the base index for each cell.
      M=0
      DO 12 J=1,NX
        NCELL(J,2)=M
        M=M+NCELL(J,1)
   12   NCELL(J,1)=0

CCCC  Process the cross reference pointer to each molecule.
      DO 13 I=1,NTOT
        J=NP(I)
        NCELL(J,1)=NCELL(J,1)+1
        L=NCELL(J,2)+NCELL(J,1)
   13   LCR(L)=I
```

FIG. 1. Scalar DSMC algorithm for the move, locate, and sort operations.

```
CCCC  DSUM(J) : sum of molecules in cell J.
CCCC  USUM(J) : sum of u velocity in cell J.
CCCC  UUSUM(J) : sum of the square of u velocity in cell J.
CCCC  ESUM(J) : sum of internal energy in cell J.
CCCC  E(I) : internal energy of molecule I.
CCCC

      DO 20 I=1,NTOT
        J=NP(I)
        DSUM(J)=DSUM(J)+1.0
        USUM(J)=USUM(J)+U(I)
        VSUM(J)=VSUM(J)+V(I)
        WSUM(J)=WSUM(J)+W(I)
        UUSUM(J)=UUSUM(J)+U(I)*U(I)
        VVSUM(J)=VVSUM(J)+V(I)*V(I)
        WWSUM(J)=WWSUM(J)+W(I)*W(I)
   20   ESUM(J)=ESUM(J)+E(I)
```

FIG. 2. Scalar DSMC algorithm for the sampling of microscopic quantities.

size is assumed uniform so that the cell location of each molecule is obtained through simple truncation. All the codes developed in the current study were written in standard FORTRAN 77 and employed the CFT77 compiler available on the CRAY machines.

After the indexing of the molecules has been completed, a number of collisions is calculated in each cell. This aspect of the DSMC technique has received a great deal of attention in the literature and will be further examined in the following section. The current description is completed with consideration of the sampling of microscopic flow properties. This is normally performed by passing through the list of all particles. For each molecule, the cell location is obtained. Then, for each of the three components, the velocity and the square of the velocity are added to the cumulative totals for that cell. A record of internal energy is also maintained if required. A representative algorithm is listed in Fig. 2 for the one-dimensional computation.

3. The DSMC Collision Algorithm

The DSMC collision algorithm may be divided into the selection of candidate collision pairs, and the calculation of post-collision properties for those molecules which are collided. While the procedure for the determination of post-collision quantities are well established, the manner in which collision pairs are selected in the DSMC method has been the subject of some debate, and several different schemes have been formulated. Most of these have been classified by Nanbu [8] on the basis of the mathematical model represented. Until quite recently, the two most common methods in use were the Time Counter scheme of Bird [1], which effectively solves the Kac Master equation, and the Modified Nanbu scheme [9], which solves the Boltzmann equation. These methods were compared by Boyd and Stark [10] and found to give excellent agreement. However, a later study revealed that more significant statistical fluctuations were associated with the Modified Nanbu scheme [11]. The deliberation over which scheme is preferable is dependent upon the information desired from the computations. It has been expressed by Bird [12] that the realistic modelling of the physics of the flow is a more important criterion than the solution of any particular mathematical equation. Therefore, for engineering applications, the Time Counter method has retained its level of popularity, with the Modified Nanbu scheme being best suited to providing solutions to the Boltzmann equation.

In the Time Counter scheme, two particles in the current cell are randomly selected. The probability that these molecules collide is then given by

$$P=\frac{\sigma g}{(\sigma g)_{\max}} \tag{1}$$

where σ is the total collision cross section and g is the relative velocity. If this pair is accepted for collision then a time counter for the current cell is incremented by an amount

$$\Delta t_c=\frac{2}{Nn\sigma g}, \tag{2}$$

where N is the number of simulated molecules in the cell and n is the number density. A number of collisions is calculated in the cell until the sum of a number of Δt_c's is just greater than the decoupling time step. It is shown rigorously in Ref. [5] that these procedures lead to the correct expression for the molecular collision rate obtained from kinetic theory. If selected, the postcollision properties of the molecules are determined through application of conservation of momentum and energy as described in Ref. [1].

Attempts to vectorize the Time Counter scheme have met with only limited success. A reasonable degree of speedup was achieved by Usami *et al.* [13]; however, the procedures proposed require forced vectorization which leads to a small degree of error in the calculations. As indicated by Baganoff and McDonald [15], the difficulty in treating the Time Counter scheme is due to the vector dependency associated with the implementation of summing several values of Δt_c given by Eq. (2). The vector architecture on a super computer achieves substantial improvement in numerical performance by simultaneously performing operations on a vector of data. A dependency occurs when an operation on one element of the vector requires a different element from the same vector. In the case of the Time Counter scheme the dependency results from the fact that the number of collisions to be performed in each cell is not known prior to calculating the collisions. In Ref. [5] a new collision algorithm is introduced in which a number of candidate collision pairs is sampled in each cell prior to entering the associated collision selection. The probability of collision for each of the pairs sampled may then be expressed as

$$P=ANn\sigma g\,\Delta t, \tag{3}$$

where Δt is the decoupling time step, and A scales the collision probability in each cell and is given in terms of the cell sample size and reference flow properties. Due to the circumstance that the number of pairs to be sampled is known, this collision algorithm may be efficiently vectorized. By careful implementation of their algorithm, Baganoff and McDonald reach a performance previously unheard of for particle simulation methods. It is demonstrated in Ref. [5] that three-dimensional simulations involving several million molecules can be completed in just 1 or 2 CPU on the CRAY-2. This clearly represents a substantial improvement on the performance obtained in Ref. [4] for an unvectorized code. However, direct comparison is made difficult due to the fact that the calculations undertaken in Ref. [4] included a much more realistic gas model with a complex computational grid.

Bird [12] has also published a new collision scheme which is termed the No Time Counter (NTC) method and has similar elements to that of Baganoff and McDonald. The idea of a time counter is replaced with the expression for the total number of collision pairs to be sampled in each cell,

$$N_c = \tfrac{1}{2} N n (\sigma g)_{\max} \Delta t \tag{4}$$

and the probability of collision for each pair sampled is again given by Eq. (1). This scheme was introduced to alleviate difficulties found with the Time Counter method for flows in which hypersonic conditions necessitated the use of very small time steps. Under such conditions, the time increment determined from Eq. (2) may substantially exceed the decoupled time step. This may then result in a reduction of the collision rate leading to an increase in shock wave thickness. This undesirable aspect of the Time Counter method is also exhibited in gas mixtures where a hot species which is present in small quantities can leak upstream of a shock and thus distort the temperature profiles in this vicinity [14].

The clearest difference between the two collision schemes given in Refs. [5, 12] pertains to the establishment of the pair sample size. In the work of Baganoff and McDonald this quantity is usually chosen as a fraction of the total number of molecules in the cell whereas in the NTC method this sample size is determined directly from Eq. (4). It is important to note that the NTC scheme employs Eq. (1) as the collision probability which by its very definition can never exceed unity as required. However, the same is not true of Eq. (3) and care must be taken in the coding to ensure that probabilities exceeding unity do not occur. The advantage in the approach adopted in Ref. [5] is that the formation of candidate pairs may be entirely vectorized.

In the current work, it is the aim to vectorize the DSMC method of Bird with the collisions calculated using the No Time Counter scheme. The purpose in this undertaking is to allow the large number of scalar DSMC codes which currently exist to be conveniently altered to obtain improved performance through the vector hardware on supercomputers. For reference, a listing of a scalar collision algorithm which employs the NTC scheme is given in Fig. 3. It should be noted that the

```
CCCC  FPAIRS(J) : number of collision pairs to be sampled in cell J.
CCCC  SIGMAGMX(J) : maximum value of the product of collision cross section
CCCC                and relative velocity for cell J.
CCCC  ANGLE(NANGLE,3) : NANGLE preprocessed unit vectors.
CCCC

CCCC  Loop over all cells
      DO 30 J=1,NX

CCCC    Determine number of pairs to be sampled from Eq.(4).
        FPAIRS(J)=FPAIRS(J)+FACTOR*NCELL(J,1)*NCELL(J,1)*SIGMAGMX(J)
        NPAIRS=INT(FPAIRS(J))
        FPAIRS(J)=FPAIRS(J)-NPAIRS

CCCC    Loop over number of samples for current cell.
        DO 31 M=1,NPAIRS

CCCC      Sample a collision pair.
          I1=NCELL(J,1)*RANF()+0.99999999
          I1=MAX(1,I1)+NCELL(J,2)
   33     I2=NCELL(J,1)*RANF()+0.99999999
          I2=MAX(1,I2)+NCELL(J,2)
          IF (I1.EQ.I2) GOTO 33
          I1=LCR(I1)
          I2=LCR(I2)

CCCC      Determine relative velocity.
          UR=U(I1)-U(I2)
          VR=V(I1)-V(I2)
          WR=W(I1)-W(I2)
          G=SQRT(UR*UR+VR*VR+WR*WR)

CCCC      Product of collision cross section and relative velocity.
          SIGMAG=FSIGMA(G)*G
          SIGMAGMX(J)=AMAX1(SIGMAGMX(J),SIGMAG)

CCCC      Probability of collision given by Eq.(1).
          IF (SIGMAG/SIGMAGMX(J).LT.RANF()) GOTO 31

CCCC      Determine center of mass velocity.
          UCM=0.5*U(I1)+0.5*U(I2)
          VCM=0.5*V(I1)+0.5*V(I2)
          WCM=0.5*W(I1)+0.5*W(I2)

CCCC      New relative velocity vector from lookup table.
          IANGLE=NANGLE*RANF()+0.99999999
          UR=G*ANGLE(IANGLE,1)
          VR=G*ANGLE(IANGLE,1)
          WR=G*ANGLE(IANGLE,1)

CCCC      Compute post collision velocities.
          U(I1)=UCM+UR*0.5
          U(I2)=UCM-UR*0.5
          V(I1)=VCM+VR*0.5
          V(I2)=VCM-VR*0.5
          W(I1)=WCM+WR*0.5
          W(I2)=WCM-WR*0.5

   31   CONTINUE

   30 CONTINUE
```

FIG. 3. Scalar DSMC algorithm for the No Time Counter collision scheme.

post-collision unit vector for the relative velocity is efficiently obtained through use of a lookup table. This idea follows the same line of thought as McDonald [7] in employing tabulated data for the speedup of various aspects of Monte Carlo simulations.

4. A Vectorized Implementation of the DSMC Algorithm

Having described the DSMC algorithm in its scalar form, the structural changes which must be made to allow efficient vectorization are now considered. Refering back to the list of tasks given in Section 2, it is noted that the vectorization of the procedures for molecule generation and boundary interaction are unnecessary. Parts of these routines may indeed be vectorized and their efficiency improved through the use of tabulated data. However, these aspects are largely unimportant for computational efficiency, and are therefore neglected in the following discussions. The movement of the molecules is vectorized in a straightforward manner and simply requires one pass through the list of all particles. This is performed separately from other operations. As discussed in Section 2, there are a number of different ways to determine the cell location of each molecule. Obviously the most efficient method should be selected for the restraints imposed by the geometry of the individual flow configuration. In any case, most of these methods are vectorizable and again just require a single pass to be made through all molecules in the simulation. The molecular indexing requires more careful consideration and does not appear to be fully vectorizable. The list of particles is looped over to obtain the number of molecules in each cell. Simultaneously, a record is maintained of the instantaneous value of the particle count in the cell that each particle occupies. This procedure then allows the processing of the cross reference array to be entirely vectorized. This is performed following the determination of the base index for each cell. Although this procedure is only partially vectorizable, this portion of the code consumes a small amount of computational effort. These ideas have been listed for the one-dimensional case in Fig. 4.

It should be noted that the efficient computation of boundary conditions is achieved by employing the base index and cross reference arrays which are later used in the calculation of collisions. These arrays may also be used to improve the efficiency of the algorithm for sampling the molecular quantities. Instead of passing down the list of molecules, partial vectorization is achieved by proceeding on a cell by cell basis. As the number of molecules in each cell is known, this allows the computer to form a vector over each cell. A listing of this routine is given in Fig. 5.

Lastly, the collision algorithm is considered. The total sample size of all collision pairs is first determined by summing over all computational cells. To be able to refer to the cell in which each of these collisions occurs it is necessary to generate a cross reference list. It is then possible to pass down this list to compute all collisions with complete vectorization. This portion of the code must be forcibly vectorized as the FORTRAN compiler on the CRAY-2 is suspicious of a vector

```
CCCC  LCT(I) : number of molecules in the cell occupied by molecule I at the
CCCC           instant I is reached in the pass through the list of particles.
CCCC

CCCC  Set number of molecules in each cell equal to zero.
      DO 40 J=1,NX
   40   NCELL(J,1)=0

CCCC  Movement of molecules.
      DO 41 I=1,NTOT
   41   X(I)=X(I)+U(I)*DELTAT

CCCC  Determination of cell index.
      DO 42 I=1,NTOT
   42   NP(I)=1+X(I)*TRUNCX

CCCC  Sum of molecules in each cell, and running total.
      DO 43 I=1,NTOT
        NCELL(NP(I),1)=NCELL(NP(I),1)+1
   43   LCT(I)=NCELL(NP(I),1)

CCCC  Process the base index for each cell.
      M=0
      DO 44 J=1,NX
        NCELL(J,2)=M
   44   M=M+NCELL(J,1)

CCCC  Process the cross reference pointer to each molecule.
      DO 45 I=1,NTOT
   45   LCR(NCELL(NP(I),2)+LCT(I))=I

CCCC  Compute all boundary interactions.
      CALL BOUNDARY
```

FIG. 4. Vector DSMC implementation for the move, locate, and sort operations.

```
      DO 50 J=1,NX
50      DSUM(J)=DSUM(J)+NCELL(J,1)

      DO 51 J=1,NX
        DO 51 K=1,NCELL(J,1)
          I=LCR(K+NCELL(J,2))
          USUM(J)=USUM(J)+U(I)
          VSUM(J)=VSUM(J)+V(I)
          WSUM(J)=WSUM(J)+W(I)
          UUSUM(J)=UUSUM(J)+U(I)*U(I)
          VVSUM(J)=VVSUM(J)+V(I)*V(I)
          WWSUM(J)=WWSUM(J)+W(I)*W(I)
51        ESUM(J)=ESUM(J)+E(I)
```

FIG. 5. Vector DSMC implementation for the sampling of microscopic quantities.

dependency which might occur if the same particle was collided more than once over the same time step. A further vector dependency may occur in the updating of the maximum value of the product of the total collision cross section and the relative velocity. However, as this product appears in both Eq. (1) and Eq. (4) then any error is effectively cancelled. The listing for the collision algorithm is given in Fig. 6. It should be noted that it is a fairly simple matter to generate sub-lists of collision pairs which are to undergo internal energy transfer, chemical reactions,

```
CCCC  NPAIRS(J) : number of collision pairs to be sampled in cell J.
CCCC  INDEX(M)  : cell index for each of M collision pair samples.
CCCC

CCCC  For each cell, determine number of pairs to be sampled from Eq.(4).
      DO 60 J=1,NX
        FPAIRS(J)=FPAIRS(J)+FACTOR*NCELL(J,1)*NCELL(J,1)*SIGMAGMX(J)
        NPAIRS(J)=INT(FPAIRS(J))
   60   FPAIRS(J)=FPAIRS(J)-NPAIRS(J)

CCCC  Process cross reference array for cell index of each sample.
      NTPAIRS=0
      DO 61 J=1,NX
        DO 62 M=NTPAIRS+1,NTPAIRS+NPAIRS(J)
   62     INDEX(M)=J
   61   NTPAIRS=NTPAIRS+NPAIRS(J)

CCCC  Compiler directive to force vectorization.
CDIR$ IVDEP
      DO 63 M=1,NTPAIRS

CCCC    Lookup current cell.
        J=INDEX(M)

CCCC    Sample a collision pair.
        I1=NCELL(J,1)*RANF()+0.99999999
        I1=MAX(1,I1)
        I2=(NCELL(J,1)-1)*RANF()+0.99999999
        I2=MAX(1,I2)+I1
        I2=I2-(I2/(NCELL(J,1)+1))*NCELL(J,1)
        I1=LCR(I1+NCELL(J,2))
        I2=LCR(I2+NCELL(J,2))

CCCC    Determine relative velocity.
        .... see Fig. 3 ....

CCCC    Product of collision cross section and relative velocity.
        .... see Fig. 3 ....

CCCC    Probability of collision given by Eq.(1).
        IF (SIGMAG/SIGMAGMX(J).GT.RANF()) THEN

CCCC      Determine center of mass velocity.
          .... see Fig. 3 ....

CCCC      New relative velocity vector from lookup table.
          .... see Fig. 3 ....

CCCC      Compute post collision velocities.
          .... see Fig. 3 ....

        ENDIF

   63 CONTINUE
```

FIG. 6. Vector DSMC implementation for the No Time Counter collision scheme.

and other physical phenomena. In this way, these important aspects of DSMC calculations may be included at little additional cost. Most of the procedures given in Figs. 4–6 may be readily extended to gas mixtures.

It is important to take into consideration the effect upon the clarity of the DSMC algorithm of the vectorized implementation. This may be judged by comparison of the listings in Figs. 1–3 with those in Figs. 4–6. Most of the implementation changes in converting from scalar to vector coding are quite straightforward in that operations are performed on lists of particles rather than one at a time. The most significant departures from the scalar code occur in the processing of NCELL and LCR in Fig. 4, as described above, and in the selection of candidate collision pairs. In the latter case, any method which ensures that a particle is not collided more than once over any one time step is suitable. The example listed in Fig. 6 is just one of several which can be constructed. It is therefore proposed that the introduction of vector programming technique does not greatly hinder the understanding of the algorithm.

5. Example Calculations

To demonstrate the advances made by the vectorized implementation of the DSMC algorithm described in the previous section, three different flow examples are considered. The first involves a one-dimensional standing shock wave of argon for a Mach number of 8. The conditions across the shock are initially specified by the continuum relationships and the boundary conditions employed are the reflecting pistons described by Bird [12]. The normalized density and temperature profiles are shown in Fig. 7 for these conditions. Results are presented for both scalar and vector codes and the solutions are found to match very well. The reciprocal shock thickness of 0.24 offers excellent agreement with the experimental

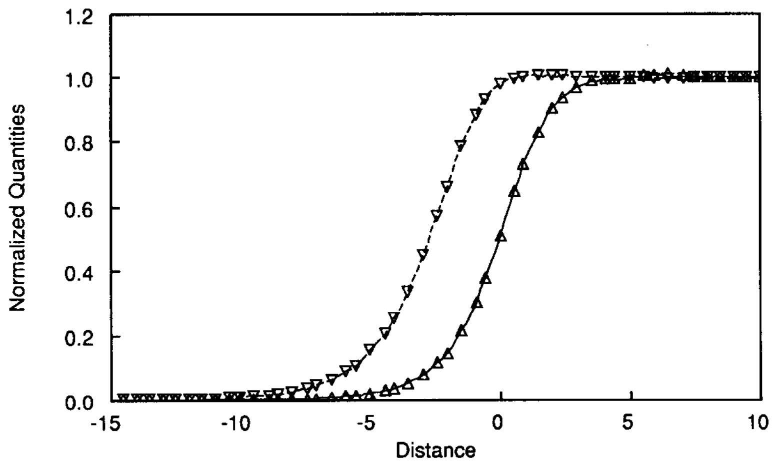

Fig. 7. Structure of a Mach 8 standing shock wave in argon: ——, density (vector); – – –, temperature (vector); Δ, density (scalar); ∇, temperature (scalar).

data given by Alsmeyer [15], thus verifying the simulation procedures. The simulations employed 100 uniform cells with about 12,000 simulated particles in the flow domain. The sampling of microscopic quantities was performed every four time step increments to improve the statistical independence of the results. Under these conditions, the solutions shown in Fig. 7 for the vectorized code required just 180 CPU s on a CRAY YMP in which time the sample sizes accumulated in each cell ranged from 7×10^5 to 2.8×10^6 and a total of 8 million collisions were computed. While a fair comparison between different codes and solution techniques is almost impossible, it is interesting to note that continuum solutions of the Burnett equations (computed with a vectorized code on the same CRAY YMP by Lumpkin [16]) for similar shock waves typically require 300–400 CPU s.

The vector DSMC code was also executed on a CRAY-2, where the performance was found to be reduced by a factor of about 2.4. To assess the improved performance attained through use of the vector hardware, the vector code was compiled and executed with the vectorization options disabled. Hence it was discovered that the speedup due to vectorization was a factor of 5. The maximum improvements in numerical performance obtained through vectorization of appropriate algorithms tend to give a speedup of around 20. By comparison with such performances the DSMC algorithm is found to be only moderately well vectorized. For the future, alternative computer hardware designs such as parallel machines and superscalar microprocessors should be considered for the DSMC algorithm. However, in comparison with previous investigations undertaken with the DSMC technique, and indeed with the continuum solution of the Burnett equations, it is clear that the improved performance of the algorithm achieved on the vector hardware of supercomputers is a very useful accomplishment.

The second example considered is the flow of carbon dioxide through a small axisymmetric nozzle. The conditions at the nozzle throat of this thruster are such that the flow may be considered rarefied long before the exit plane is reached. The nozzle conditions are given in Table I and have been specified by engineers of the Low Thrust Propulsion Branch at NASA Lewis Research Center. The DSMC calculation was begun at the throat with the conditions obtained from a Navier–Stokes solution of the nozzle flow. The nozzle wall was modelled as a fully accomodated diffuse reflector with the local wall temperature again obtained from

TABLE I

Conditions for Axisymmetric Nozzle Flow

Gas type	CO_2
Stagnation temperature	1000 K
Stagnation pressure	7700 Pa
Throat Reynolds number	1000
Nozzle throat diameter	3.175×10^{-3} m
Nozzle exit angle	20°
Nozzle area ratio	100

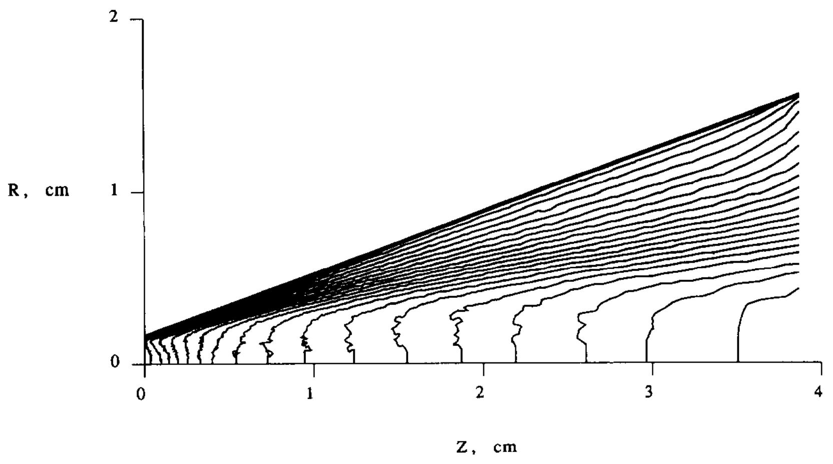

FIG. 8. Mach number contours for a small axisymmetric nozzle.

the continuum solution. The axisymmetric flow is modelled in two dimensions in the usual manner, as described in Ref. [1]. Hence the molecular velocities and radial position must be rotated to ensure the axisymmetric nature of the flow. The nonequilibrium relaxation of the rotational energy mode is modelled using the variable exchange probability developed by Boyd [17]. Due to the low temperatures involved in the simulation, the vibrational mode is assumed to be frozen. The Mach number contours for the flow are shown in Fig. 8 and range from 0.75 near the wall to 5.5 in the isentropic core. As found in previous studies, the sonic line intersects the nozzle lip. This phenomenon is not predicted by the continuum solution. The temperature profiles in the nozzle exit plane are given in Fig. 9 for

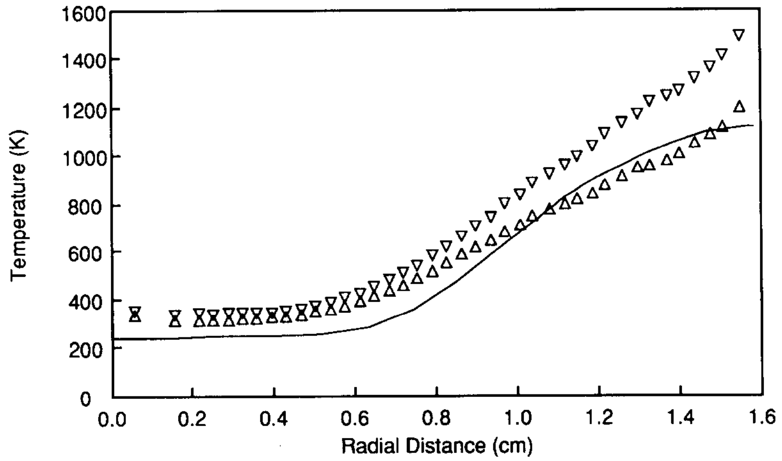

FIG. 9. Temperature profiles obtained with DSMC and Navier–Stokes codes in the exit plane of a small axisymmetric nozzle: —, total (continuum); Δ, translational (DSMC); ∇, rotational (DSMC).

both the continuum and DSMC solutions. It may be seen that significant non-equilibrium exists between the translational and rotational modes, while the temperatures obtained with the DSMC solution are larger than that obtained from the Navier–Stokes computations. A similar trend is observed for the density solutions at the exit plane. A total of 15,000 cells and 1 million simulated particles were employed in the flowfield. Due to the very large change in density experienced in the flowfield as the gas expands, the cell sizes and decoupling time steps were varied with geometric location. During the sampling phase of these computations, the vectorized code marched the 1 million particles through 1750 time steps and processed over 140 million collisions every CPU hour on a CRAY-2. The overall performance is slightly slower than that for the shock wave and it is generally found that the efficiency of the code is largely governed by the complexity of the computational geometry employed. In this application, an optimized version of Bird's adjacent cell procedure was used in the determination of the location of each molecule.

The final example considered is the hypersonic flow of argon over a wedge with an apex half angle of 40°. The freestream conditions are listed in Table II and have been chosen to correspond to one of the experimental cases investigated by Hornung and Smith [18] except that the density of the flow has been reduced by a factor of 1000. One of the concerns in Ref. [18] was the three-dimensionality of the flow. Therefore, in the present application of the vectorized code, this flow is treated in three dimensions. The computational grid was divided into four separate regions to allow proper simulation of the interaction with the wedge and the surface was modelled as a diffuse reflector fully accomodated to a wall temperature of 300 K which was chosen to be representative of the experimental investigation [19]. Temperature contours for the flow are shown in Fig. 10 in the XY plane for $Z = 0$, which corresponds to the plane through the center of the wedge. The molecules enter the computational domain across the plane $X = 0$ and are removed after crossing the plane specified by the X coordinate corresponding to the back end of the wedge. All other boundaries specularly reflect the molecules back into the flowfield. Due to the large angle of the wedge, the shock is detached from the leading edge and this phenomenon is clearly observed in Fig. 10. Contours of density ratio are shown in Fig. 11 in the XZ plane for $Y = 0$, which again represents the plane of symmetry. It is seen that the flow in front of the wedge becomes only

TABLE II

Conditions for Three-Dimensional Wedge Flow

Gas type	Ar
Freestream temperature	57 K
Freestream density	2.2×10^{-6} kg m^{-3}
Freestream Mach number	16
Wedge length	0.051 m
Wedge extent	0.152 m
Wedge angle	40°

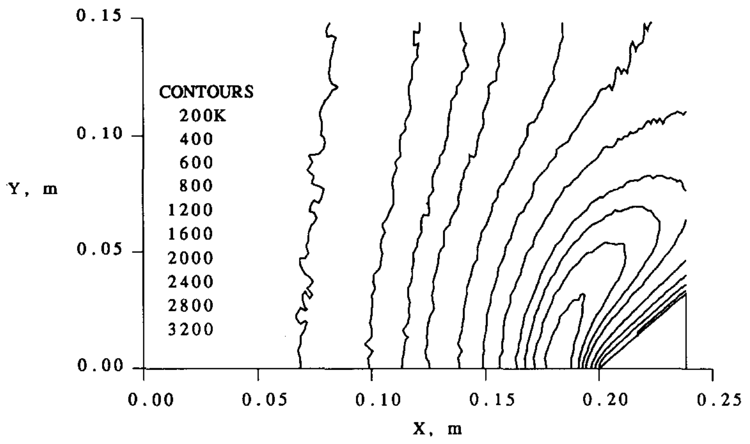

FIG. 10. Temperature contours for hypersonic flow over a wedge in the XY plane for $Z=0$.

slightly three-dimensional as the corner is approached. The shock standoff distance attained from the calculation is 0.0057 m which compares very favorably with the experimental result of 0.005 m [18]. The flowfield was computed using over 375,000 cells and 10 million simulated molecules. This number of molecules matches the largest total previously reported in the literature in Ref. [5] and is by far the largest achieved for Bird's DSMC method. The performance of the vectorized code for this application is such that the 10 million particles are advanced through 132 time steps in 1 CPU h on the CRAY-2.

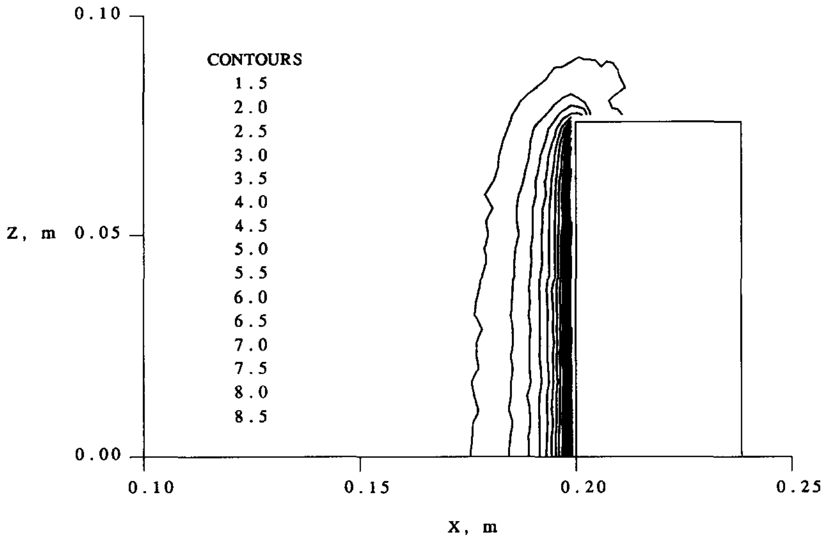

FIG. 11. Contours of density ratio (ρ/ρ_∞) for hypersonic flow over a wedge in the XZ plane for $Y=0$.

CONCLUSIONS

Substantial improvements in the numerical performance of the Direct Simulation Monte Carlo method have been attained by the application of vector programming techniques to an algorithm which employs the No Time Counter collision scheme. The new computational possibilities of the vectorized code have been demonstrated. It is found that the geometric complexity of the computational grid employed is the dominating aspect in terms of numerical performance. While the use of uniform and regular shaped cells gives the greatest efficiency, it has not been proven that such procedures provide accurate solutions for flowfields involving interaction with solid surfaces. The most important conclusion to be drawn from the present study is that the new vectorized implementation allows the DSMC method to take good advantage of the hardware of supercomputers. It is therefore possible to run a larger number of parametric studies for moderately sized problems, or alternatively model problems on a scale which has not been previously possible. All through its development, the DSMC method has suffered from its notoriously poor numerical efficiency. It is therefore proposed that the advances reported here will go some way to reducing this obstacle, thereby encouraging wider use of this important computational technique.

ACKNOWLEDGMENTS

Support (for IDB) has been provided by NASA Grant NCC2-582. Helpful discussions with the research group of Professor D. Baganoff and Dr. J. D. McDonald are gratefully acknowledged. Thanks are extended to Dr. P. F. Penko of NASA Lewis Research Center for providing the continuum solution of the axisymmetric nozzle flow.

REFERENCES

1. G. A. BIRD, *Molecular Gas Dynamics* (Univ. of Oxford Press, Clarendon, Oxford, 1976).
2. G. A. BIRD, *Commun. Appl. Numer. Methods* **4**, 165 (1989).
3. E. P. MUNTZ, *Annu. Rev. Fluid Mech.* **21**, 387 (1989).
4. M. C. CELENLIGIL, G. A. BIRD, AND J. N. MOSS, *AIAA J.* **27**, 1536 (1989).
5. D. BAGANOFF AND J. D. MCDONALD, *Phys. Fluids A* **2**, 1248 (1990).
6. T. ABE, *J. Comput. Phys.* **83**, 424 (1989).
7. J. D. MCDONALD, Ph.D. thesis, Stanford University, December 1989.
8. K. NANBU, in *Proceedings, 15th International Symposium on Rarefied Gas Dynamics, Grado, Italy*, 1986 (Teubner, Stuttgart, 1987), p. 369.
9. H. PLOSS, *Computing* **38**, 101 (1987).
10. I. D. BOYD AND J. P. W. STARK, *J. Comput. Phys.* **80**, 374 (1989).
11. I. D. BOYD, *Comput. Phys.* **3**, 73 (1989).
12. G. A. BIRD, in *Rarefied Gas Dynamics*, Progress in Astronautics and Aeronautics Series, Vol. 118 (AIAA, Washington, DC, 1989), p. 211.

13. M. Usami, T. Fujimoto, and S. Kato, in *Rarefied Gas Dynamics*, Progress in Astronautics and Aeronautics Series, Vol. 116 (AIAA, Washington, DC, 1989), p. 283.
14. I. D. Boyd, "Rotational and Vibrational Nonequilibrium Effects in Rarefied Hypersonic Flows," *J. Thermophys. Heat Transf.* **4**, 478 (1990).
15. H. Alsmeyer, *J. Fluid Mech.* **74**, 497 (1976).
16. F. E. Lumpkin, Ph.D. thesis, Stanford University, March 1990.
17. I. D. Boyd, *Phys. Fluids A* **2**, 447 (1990).
18. H. G. Hornung and G. H. Smith, *J. Fluid Mech.* **93**, 225 (1979).
19. H. G. Hornung, private communication.