NUMERICAL SIMULATION OF THE BLAST IMPACT PROBLEM USING THE DIRECT SIMULATION MONTE CARLO (DSMC) METHOD

Anupam Sharma [1] and Lyle N. Long [2]

Department of Aerospace Engineering,
233 Hammond Building
University Park, PA - 16802

Abstract

A particle approach using the Direct Simulation Monte Carlo (DSMC) method is used to solve the problem of blast impact with structures. A novel approach to model the solid boundary condition for particle methods is presented. The solver is validated against an analytical solution of the Riemann shocktube problem and against experiments on interaction of a planar shock with a square cavity. Blast impact simulations are performed for two model shapes, a box and an I-shaped beam, assuming that the solid body does not deform. The solver uses domain decomposition technique to run in parallel. The parallel performance of the solver on two Beowulf clusters is also presented.

Key words: Direct Simulation Monte Carlo (DSMC), Blast Impact, Object Oriented Programming (OOP)

1 Introduction

The protection of civilians and military personnel from an attack by another country or by terrorists is of primary importance for a nation. In recent times of heightened terrorist activities and alarming threats of future attacks, it has become utmost important to develop safe housing and barracks respectively

Email address: lnl@psu.edu (Lyle N. Long).
[1] Graduate Research Assistant, Aerospace Engineering
[2] Professor, Aerospace Engineering

for innocent civilians and military personnel. This problem requires the scientific community to research the general area of design and development of protective structures. Recent tragic mishaps such as the attacks on the World Trade Center and the USS Cole have brought this problem to the foreground.

Research in this area has been ongoing for several years but has been focused primarily on open-air, field blasts. Such experiments involve detonation of real explosives and investigation of the impact of the explosion on nearby structures constructed specifically for this purpose. These experiments are both hazardous and expensive. Also, such large-scale experiments involving substantial amount of explosives can only be performed by the military because of the federal regulations.

A different approach to the problem is to use computers to simulate the impact of a blast wave generated from an explosion with solid structures. This approach is more feasible for universities and other private research groups and is free of hazards associated with the handling of explosives.

This study focuses on numerical simulations of the interaction of blast waves with structures and the prediction of resulting pressure loading. This is also known as the "load definition" problem. The pressure loading may then be used as an input to a commercially available structural dynamics solver, such as ANSYS to determine if the structure would fail in such circumstances. In this manner the structural mechanics problem (of solving the strain and deformation) is uncoupled from the fluid dynamics problem (of predicting the pressure / stress loading). The uncoupled fluid dynamics problem is solved assuming that the structure does not deform because of the impact. Hence, the assumption of uncoupling is only accurate if the structure remains intact. Once the structure deforms it will alter the blast wave and the two problems will become coupled. In the present study we are dealing only with the uncoupled problem.

The key requirements of a numerical solver for this problem are:

- It should be able to handle complicated geometries such as buildings (for external blasts), and cubicles and rooms (for internal blasts).
- This mission is time critical and hence the simulations should be relatively fast. One cannot afford to spend months on a single simulation. The speed also determines the cost of the computation. The faster the results are obtained, the more economical it is.
- Since the goal is to design for threats that cannot be accurately determined, slight inaccuracy in the results is acceptable if it significantly reduces the turnaround time.

The Direct Simulation Monte Carlo (DSMC) method meets the above conditions and is chosen for the study. The following section discusses in detail

the advantages of DSMC over conventional Computational Fluid Dynamics (CFD) methods for the problem of interest.

A parallel, object oriented DSMC solver is developed for this problem. The solver is written in C++ and is designed using the object oriented features of the language. The parallelization is achieved by domain decomposition and using the Message Passing Interface (MPI) library for inter-processor communications. There are two reasons for making the code parallel: firstly, to solve large-size problems, and secondly, to reduce the turnaround time.

In the following sections the implementation of the solid and inflow boundary conditions, and diatomic gas modeling are described. The object oriented approach for the solver and the parallelization technique are also discussed. The solver is verified against the analytical solution of the Riemann shocktube problem. The solid and inflow boundary conditions are separately validated against analytical results. The problem of a normal shock interaction with a square cavity is simulated and compared against the experiments by Igra *et al.* [1]. Two model shapes, a box and a I-shaped beam are studied for blast impact.

2 DSMC

The Direct Simulation Monte Carlo (DSMC) method has gained a lot of popularity since its development by Bird [2] in the 1960s. The method has been thoroughly tested for high Knudsen-number (> 0.2) flows over the past 25 years and found to be in excellent agreement with both experimental data [3] and Molecular-Dynamics computations [4]. The method was developed primarily for aerospace applications in rarefied gas dynamics and has been extensively applied in that area [3,5,6,7,8]. DSMC has also been successfully used for hypersonic flows [6,9,10] and modeling detonations [11]. In fact, DSMC has become, *de facto*, the principal tool to investigate high Knudsen number flows. DSMC is also very useful for modeling flows involving chemical reactions [12,13,14,15].

DSMC is a direct particle simulation method based on the kinetic theory of gases. The fundamental idea is to track a large number of statistically representative particles. The particles move according to Newton's laws and collide with each other conserving mass, momentum and energy. The particles' motion is calculated deterministically but the collisions are treated statistically. The direction of a molecule's post-collision velocity is calculated by performing *random walks* (a random process consisting of a sequence of discrete steps of fixed length) while conserving momentum and energy. Hence, the collisions in DSMC are only statistically correct. The treatment of intermolecular collisions

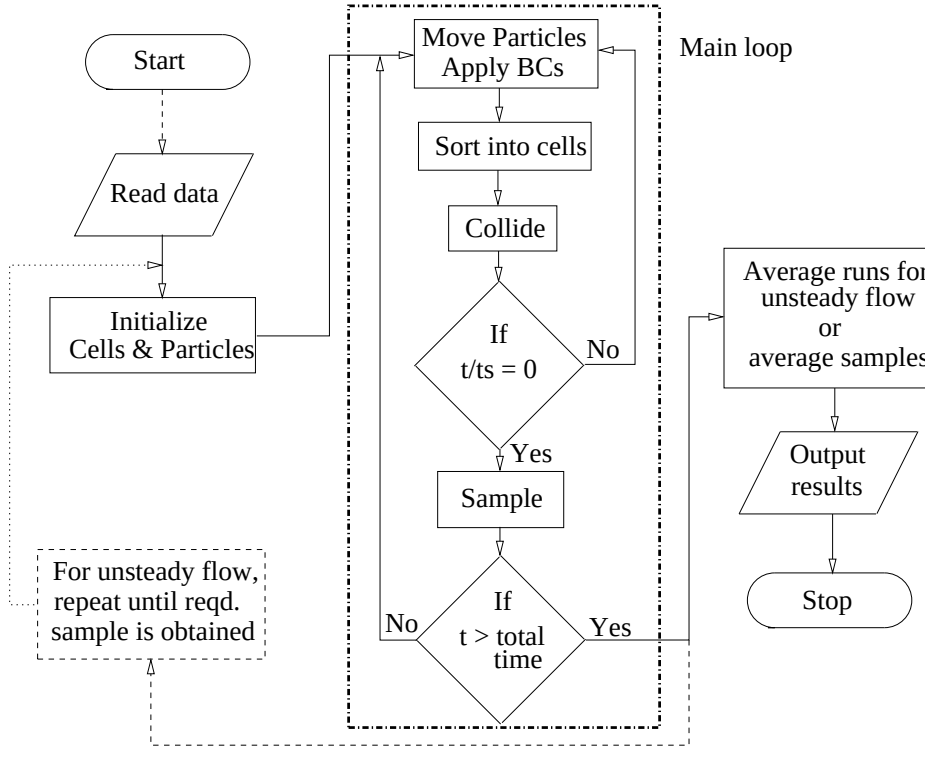


Fig. 1. A Flowchart of a typical DSMC simulation.

is the key difference between Molecular Dynamics and DSMC. In the Molecular Dynamics method the particle interactions are calculated using potentials such as the Lennard-Jones potential.

Figure 1 is a flowchart of a typical DSMC simulation. There are essentially four routines in a DSMC simulation: (a) move the particles for time Δt assuming no intermolecular collisions but treat the collisions with the domain boundaries, (b) sort the particles in different cells, (c) perform intermolecular collisions, and (d) sample macroscopic properties of interest.

DSMC simulations become more exact as the time step and the cell size tend to zero. Wagner [16] has proved that the DSMC method converges to the solution of the Boltzmann equation in the limit of the time step and the cell size equal to zero. Since the motion of the particles is independent of the cell structure, the disturbances can propagate at the sound or shock wave speed even when the ratio of cell size to the time step is relatively very small. Therefore, DSMC is not limited by a stability criterion such as the Courant-Fredrichs-Lewy (CFL) condition.

There have been few attempts [17,18] to use DSMC for continuum problems ($Kn < 0.1$). This is because DSMC inherently assumes that the cell size and the time step in a simulation are of the order of magnitude of the mean free path and the mean collision time respectively of the gas. Typical magnitudes

of the mean free path and the mean collision time of air in standard atmospheric conditions are of the order 10^{-8} m and 10^{-10} seconds respectively. A simulation of flow over large bodies (typical length O(m)) in such conditions requires enormous time and memory. But, in a rarefied atmosphere, the values of the mean free path and the mean collision time can be quite large. This makes possible the simulation of flows over re-entry vehicles using DSMC in reasonable time. When the body of interest is comparable to the size of the mean free path, e.g., in micro-devices, DSMC can also be inexpensively used at standard atmospheric conditions.

Pullin [17] suggested an algorithm to extend the application of DSMC to the continuum regime for inviscid, perfect-gas flows. The assumption of perfect-gas inviscid flow is equivalent to assuming local thermal equilibrium ($Kn = 0$) at every point and at all times [17]. This is analogous to solving the Euler equations. In his algorithm, Pullin reinitializes the velocity of every molecule to the local Maxwellian after every iteration thus insuring local equilibrium. Pullin [17] suggested that for such idealized cases, the cell size and the time step could be chosen to be a practically infinite number of times of the molecular mean free path and the mean collision time since the fluid properties are assumed to be constant over each cell. The error in making such an approximation is that the flow properties are smeared over a cell width. For example, if we simulate a shock wave, the minimum thickness of the wave by a DSMC simulation will be equal to the cell size.

Interestingly, Pullin [17] ruled out the possibility of using his approach because of the time penalty associated with the calculation of equilibrium molecular velocities at each time step. Recently, Sharma and Long [19] have proposed an inexpensive procedure to achieve local thermodynamic equilibrium. In this procedure, one allows enough collisions per cell per time step to relax the gas to the local Maxwellian. The maximum number of collisions is a function of the number of particles in the cell but we fix it to about three-fourths of that number. This is a conservative approximate as can be inferred from Figs. 2 and 3. These figures plot the results of a simple experiment of rotational relaxation in air. The gas is initially perturbed from Maxwellian by assigning zero rotational temperature. Figure 2 plots the rotational and translational temperatures as a function of number of collisions for three different numbers of particles per cell. After a sufficient number of collisions, the gas returns to the local Maxwellian. This limit increases with increasing number of particles, but is always less that three-fourths the value as can be seen from Fig. 3.

In order to do one collision, approximately 60 floating point operations are required as opposed to about 360 required to assign a Maxwellian velocity to one particle. Hence, the procedure of relaxing the gas to a Maxwellian using collisions is at least about six times more economical. A significant reduction in the computation time by this procedure makes DSMC a practical Euler

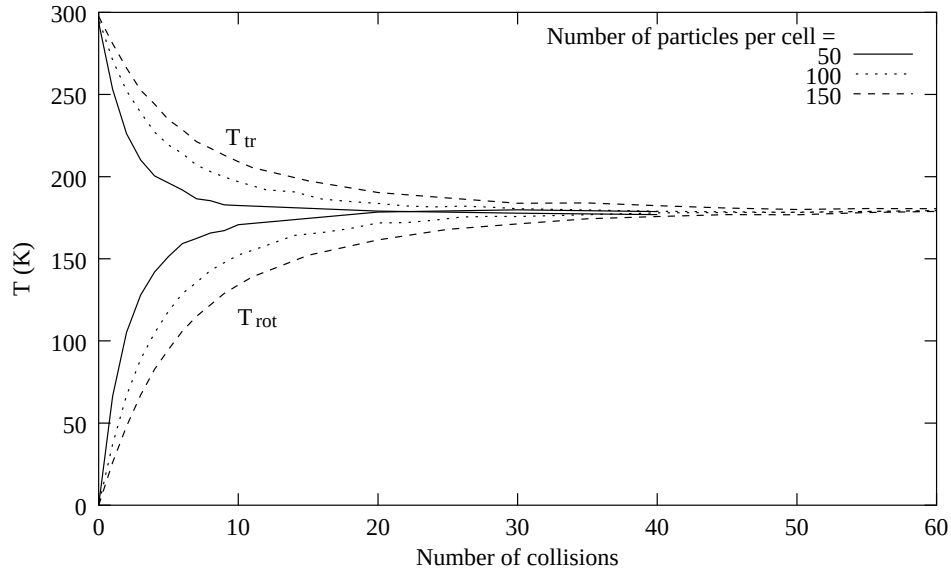


Fig. 2. Rotational relaxation in air.

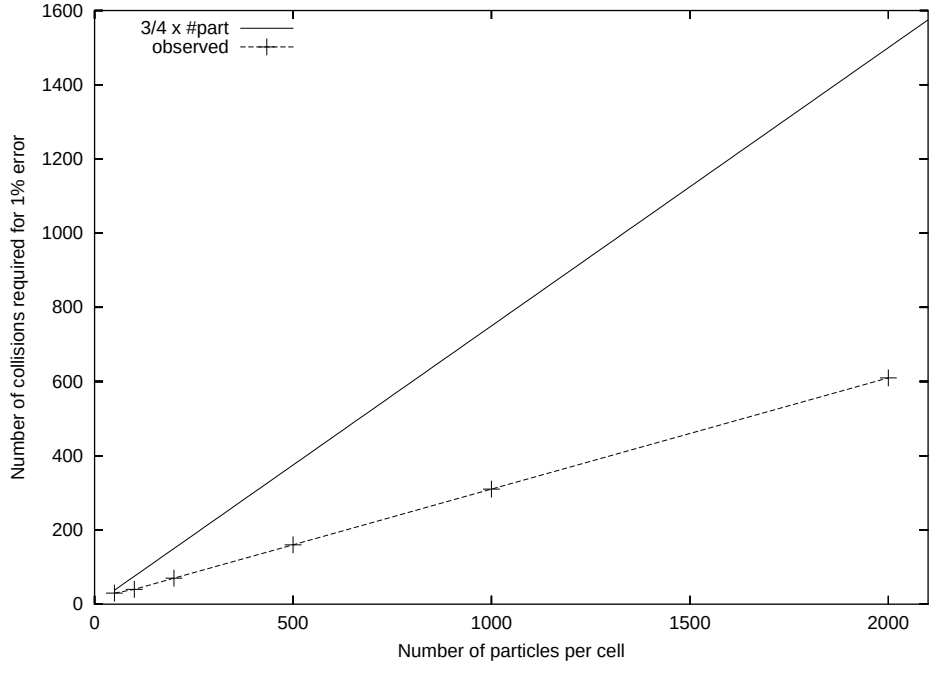


Fig. 3. Minimum number of collisions per cell per time step required to reduce the difference between rotational and translational temperatures to within 1%.

solver.

It is essentially the robustness of DSMC that has made this method widely popular. Research is going on to apply this method to other areas such as acoustics [20,21]. The interested reader is referred to Bird [6] for detailed description of DSMC and its applications.

3 DSMC Solver

A parallel, object oriented DSMC solver is developed to study blast wave structure interactions. A uniform, Cartesian grid is used with embedded surfaces to model complex geometries. The particles are distributed uniformly across the domain and their velocities are given by a Maxwellian distribution at the start of each run. All the particles inside the solid body (closed surface) are removed. This is done to avoid any information exchange into the solid body because of intermolecular collision.

The implementation of different aspects of the solver are described in the following subsections.

3.1 Solid Boundary Condition

The ultimate goal of a blast-impact simulation is to couple the shock-wave simulation with a structural dynamics model. This would allow simultaneous deformation of the solid body in the simulation as the stress on the body exceeds the yield stress. Although the solid body is not allowed to deform in this study, a very general approach to model the solid boundary is developed. This will allow the same code to be used with little modification when the body is simultaneously deformed.

Since we are only interested in Euler calculations, inviscid boundary conditions are imposed on the solid surfaces. Inviscid boundary condition in the solver is implemented by imposing specular reflection of the particles when they hit a solid surface. The solid surfaces are made of triangular patches to incorporate arbitrary-shaped bodies. The patches may be arbitrarily oriented in 3-dimensions and the particles are also moving in a 3-dimensional space. We need to find the post-collision position and velocity of the particles reflected off the solid surface. The probability of a collision of a particle with a given triangle is very small and therefore it is imperative to devise a computationally efficient algorithm to reject the particles that do not collide with the triangle.

A quick way to eliminate the particles which will definitely not collide with the solid body is by checking if the particle crosses into a box bounding the solid body. It is much cheaper to check the intersection with a bounding box (a cuboid) than an arbitrary solid body since it only requires six logical statements in a subroutine to compare the three Cartesian coordinates (position) with the dimensions of the box. An easy way to implement this is by using the Sutherland line clipping algorithm [22] in three dimensions. This algorithm is widely used in polygon clipping in computer graphics applications. The idea is to have an efficient way to ignore the line segments that lie completely outside

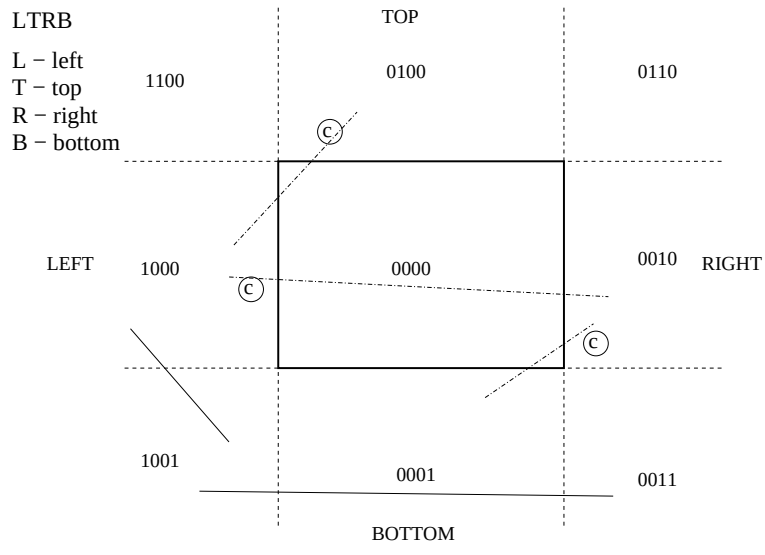


Fig. 4. Sutherland line clipping algorithm for a 2-D case. The lines marked with © should be clipped.

the canvas. If we extend the canvas to 3 dimensions, we get a box and instead of 2-D line segments we can deal with 3D line segments. Correlating line segments with the motion of particles in one time step, and the 3D box with a box bounding our solid object, we can directly use the Sutherland algorithm for our problem.

Figure 4 illustrates the Sutherland algorithm in 2D. The 2D space is divided by the square (canvas) into 4 regions with respect to the canvas - LEFT, TOP, RIGHT and BOTTOM. Each point in the 2D space can now be located by using a binary representation in four bits one each for LEFT, TOP, RIGHT and BOTTOM. The appropriate bits are turned on depending on the location of the point (i.e. if the point is to the left and top of the square then the LEFT and TOP bits are 1 and the rest are 0). It is intuitive that if we get a nonzero value when we perform an "AND" operation on the binary representation of the end points of a line segment, the segment will never intersect the canvas.

This concept is easily extended to three dimensions by adding two more bits - OUT and IN for the third dimension. Figure 5 illustrates this point. The "AND" operation is now performed over the 6-bit binary representation but there is no increase in cost for this check. The cost increase is only in identifying the (binary) location of the particle in the 3D space.

Once we are through the bounding box test and the line segment is found to intersect the bounding box, we do further tests to eliminate the particles that may not intersect the solid body. If there is no collision (with the solid body), the particle would go from position A to position B in the time interval Δt

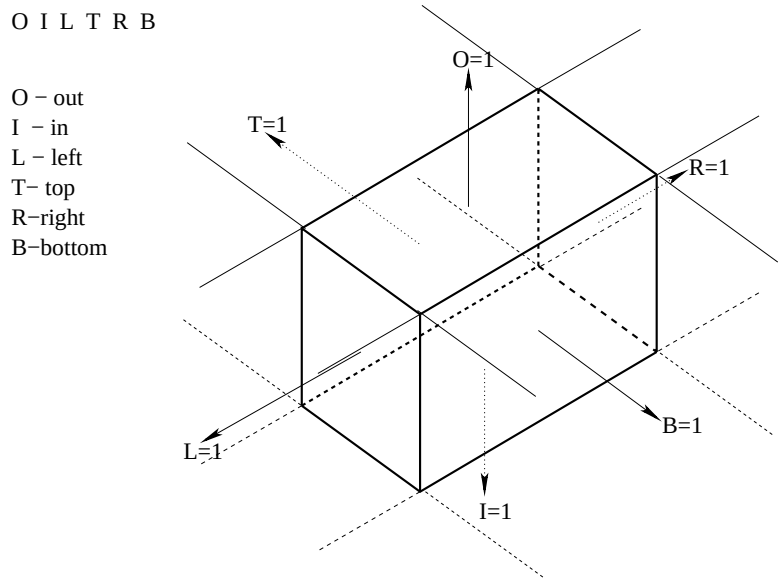


Fig. 5. Sutherland line clipping algorithm extended to 3 dimensions.

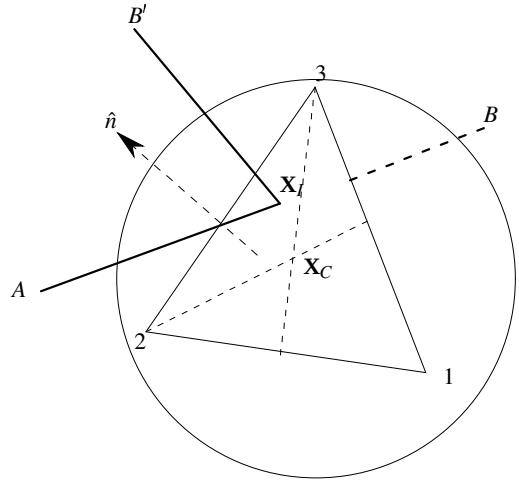


Fig. 6. A schematic of a particle reflecting off a solid triangular surface.

(refer Fig. 6). We use a parametric representation of the line segment AB.

$$\mathbf{X} = \mathbf{X}_A + \mathbf{V}t \tag{1}$$

where $\mathbf{V} = \mathbf{X}_B - \mathbf{X}_A$ is the fictitious velocity of the particle such that the particle will reach B from A in $t = 1$ unit. For finite AB, t can range only between 0 and 1.

The equation of an infinite plane formed by the three vertices of a triangle,

$\mathbf{X}_1, \mathbf{X}_2$ and $\mathbf{X}_3$ is

$$(\mathbf{X} - \mathbf{X}_1).\hat{\mathbf{n}} = 0 \tag{2}$$

where, $\mathbf{X}$ is the vector defining the plane and $\hat{\mathbf{n}}$ is a unit normal to the plane calculated using Eq. 3.

$$\hat{\mathbf{n}} = \frac{(\mathbf{X}_1 - \mathbf{X}_2) \times (\mathbf{X}_3 - \mathbf{X}_2)}{|(\mathbf{X}_1 - \mathbf{X}_2) \times (\mathbf{X}_3 - \mathbf{X}_2)|} \tag{3}$$

Note that the normal to the plane need not be calculated at every time step. It may be calculated during the first iteration and stored for the rest of the simulation. The intersection point, $\mathbf{X}_I$ of the segment with the infinite plane can be obtained by simultaneously solving Eqs. 4 and 5 for t^* and $\mathbf{X}_I$.

$$(\mathbf{X}_I - \mathbf{X}_1).\hat{\mathbf{n}} = 0 \tag{4}$$
$$\mathbf{X}_I = \mathbf{X}_A + \mathbf{V}t^* \tag{5}$$

The solution of the above equations is

$$t^* = \frac{(\mathbf{X}_A - \mathbf{X}_1).\hat{\mathbf{n}}}{\mathbf{V}.\hat{\mathbf{n}}} \tag{6}$$

and $\mathbf{X}_I$ may be obtained from Eq. 5 using t^* from Eq. 6. The finite segment, AB intersects the infinite plane only if $0 \leq t^* \leq 1$ The equality on either side meaning that $\mathbf{X}_I$ is the same as $\mathbf{X}_A$ or $\mathbf{X}_B$. If t^* does not satisfy the above condition, then we can say that the particle will not hit the triangle. If, however, t^* satisfies the above condition then we need to find out if $\mathbf{X}_I$ lies inside the triangle.

A quick elimination of a number of particles not intersecting the triangle may be performed by checking if $\mathbf{X}_I$ lies outside a circle enclosing the triangle. An obvious choice for this is the circumcircle, but we do not choose the circumcircle because it is computationally involving to obtain the center of a circumcircle in three dimensions. We choose a circle with a radius, $r = 2/3\times$ the largest edge of the triangle, and its center at the centroid. This circle will completely enclose the triangle but will not exactly circumscribe it. The reasoning behind this is - the vertex of a triangle farthest from its centroid is at a distance of $2/3\times$ the maximum of the three medians. Since the largest edge of a triangle is always greater than the largest median, the circle with radius r and center at the centroid of the triangle will completely enclose the triangle.

We calculate the centroid of the triangle and the radius of the enclosing circle described above during the first iteration. All $\mathbf{X}_I$s with $|\mathbf{X}_I - \mathbf{X}_C| > r$ will hit

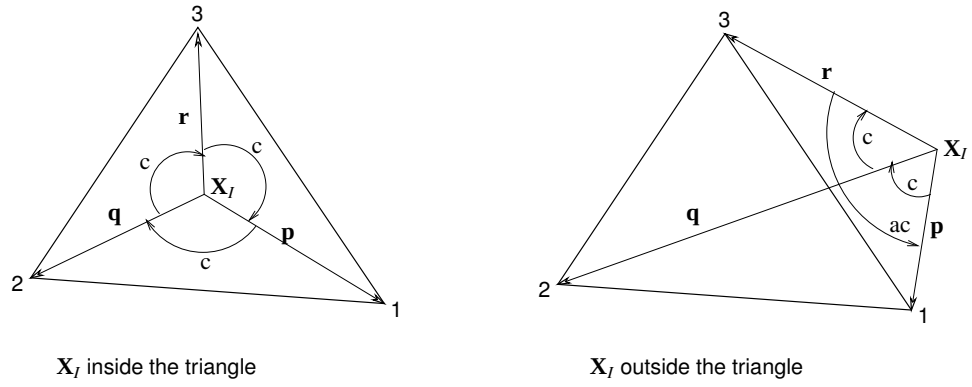


Fig. 7. An illustration showing that the angles subtended by all the edges have the same sign if $\mathbf{X}_I$ is inside the triangle but have different signs if $\mathbf{X}_I$ is outside. The subtended angle is considered positive if it is clockwise (represented by 'c'), and negative if it is anti-clockwise (represented by 'ac').

the infinite plane outside the circle and hence outside the triangle. Here, $\mathbf{X}_C$ is the centroid of the triangle.

If $\mathbf{X}_I$ lies inside the circle then there is no shortcut to figure out if it is inside the triangle or outside. Since the vertices of all the triangles are stored in a specific order (cyclic or anti-cyclic), the angles subtended by each edge (viz.. 1-2, 2-3 and 3-1) of the triangle at the point $\mathbf{X}_I$ will all be either positive or negative if $\mathbf{X}_I$ lies inside the triangle. If it lies outside the triangle then there will be a change of sign. Figure 7 illustrates the above point. In fact, the sum of the subtended angles is equal to zero if $\mathbf{X}_I$ is outside the triangle, but we don't want to do an extra summation.

The change in the sign of the subtended angles can be perceived by looking at the directions of the cross products - $\mathbf{p}\times\mathbf{q}$, $\mathbf{q}\times\mathbf{r}$ and $\mathbf{r}\times\mathbf{p}$, where $\mathbf{p} = \mathbf{X}_1 - \mathbf{X}_I$, $\mathbf{q} = \mathbf{X}_2 - \mathbf{X}_I$ and $\mathbf{r} = \mathbf{X}_3 - \mathbf{X}_I$. If all of them point in one direction then $\mathbf{X}_I$ lies inside the triangle otherwise it lies outside the triangle. If two vectors point in the same direction their dot product is positive; if they point in opposite directions then their dot product is negative. Note that $\mathbf{p} \times \mathbf{q}$, $\mathbf{q} \times \mathbf{r}$ and $\mathbf{r} \times \mathbf{p}$ are either parallel or anti-parallel depending on whether $\mathbf{X}_I$ lies inside or outside the triangle since $\mathbf{p}$, $\mathbf{q}$ and $\mathbf{r}$ are co-planar. We check the dot products $(\mathbf{p} \times \mathbf{q}).(\mathbf{q} \times \mathbf{r})$ and $(\mathbf{q} \times \mathbf{r}).(\mathbf{r} \times \mathbf{p})$; if both are positive then we conclude that $\mathbf{X}_I$ lies inside the triangle otherwise it lies outside the triangle.

If a particle hits the solid surface then it is bounced off the surface specularly just like a light ray reflects off a mirror. This is fairly easy to implement if we write the velocity vector as a sum of the normal velocity (normal to the plane) and tangential velocity. The tangential direction is given by $\mathbf{V}_t = \mathbf{V} - \mathbf{V}_n$. For the reflected velocity, we just change the sign of the normal velocity while the tangential component remains unchanged. Hence, the reflected velocity is $\mathbf{V}_t - \mathbf{V}_n$. The final position is calculated by moving the particle with the

reflected velocity for the remaining time taking into consideration that there might be further collisions if the solid body is concave.

The solid boundary condition implementation is summarized in a step-by-step procedure below:

(1) Ignore all particles that lie outside the bounding box. These are found using the Sutherland algorithm.
(2) For each particle, calculate the intersection of its trajectory with the infinite planes formed by each triangle of the solid body. Ignore the particles that do not intersect with any of the planes.
(3) Ignore the collision if the intersection point (found in step 2) lies outside a circle that encloses the corresponding triangle.
(4) Check if the intersection point lies inside the triangle. If yes, then the molecule collides with the solid body and is reflected back specularly from the surface.

3.2 Inflow Boundary Condition

An inflow boundary condition is used for shocktube type simulations where a planar shockwave is desired. For a specified shock strength and atmospheric conditions ahead of the shock wave, the conditions behind the shock wave may be calculated by using normal shock relations (c.f., ref. [23]). The velocity, density and temperature behind the shock wave are then applied at the inflow boundary. This approach can be used with conventional CFD schemes as well as with particle methods such as DSMC.

The treatment of the inflow boundary condition in DSMC is different than in the CFD schemes. This is because, in conventional CFD schemes, the boundary condition is known in terms of flow variables whereas in DSMC we need the number, position and velocity of the entering particles. A simple way to implement the inflow boundary condition is to delete all the particles that go out of the inflow boundary and those that are in the first layer of cells near the boundary at each time step. The correct number of particles (obtained by dividing the density by the product of the volume of a cell and the ratio of real to simulated particles) are then inserted in each cell in this layer. These are distributed uniformly as we assume local thermodynamic equilibrium at the boundary. The velocities of these particles are obtained using a Maxwellian distribution based on the inflow velocity and temperature calculated using normal shock relations.

The air is considered to be essentially a diatomic gas. The DSMC solver is required to model the diatomic nature of a gas as the present study concentrates on shock waves propagating through air. A diatomic molecule has five degrees of freedom while a monoatomic molecule has only three degrees of freedom. The additional degrees of freedom are from the rotation of molecules about their center of mass. The rotation allows for additional storage of energy in the molecule, namely, the rotational kinetic energy. Vibrational modes are not included here.

A monatomic gas may be represented by hard spheres. However, an internal energy model is required to truly represent a diatomic gas. Such a model is required to incorporate the internal energy associated with the rotation of the molecule. It is important to note that our only concern is to handle the internal energy exchange between the molecules during the collisions; we are not worried about the structure of the molecule and how the collisions actually occur considering the structure of the molecules.

Larsen and Borgnakke [24] model a diatomic molecule as a sphere with a variable internal energy. The internal energy and the translational energy of the colliding molecules are redistributed during the collision but the total energy is conserved. This is a phenomenological model which may be used for polyatomic gases with multiple degrees of freedom. The following mathematically describes the model for a diatomic gas.

The internal energy of each molecule is initialized to be $e_i = mRT_r$ where m is the mass of the molecule, R is the gas constant and T_r is the rotational temperature. T_r is initially the same as the total temperature since the gas is in thermal equilibrium. Note again that each molecule is diatomic but it is considered to be spherical for calculating the collision frequency. The internal energy is considered only when the energy is redistributed during the collision process.

Two molecules are randomly chosen from a cell to be considered for collision. Let us name these molecules a and b for referencing. The relative velocity of the molecules is calculated as $\mathbf{g}_{ab} = \mathbf{u}_b - \mathbf{u}_a$. The collision pair (a, b) is then accepted if the following is satisfied.

$$g_{ab}^{(\nu-5)/(\nu-1)} / [g_{ab}^{(\nu-5)/(\nu-1)}]_{\max} > \mathcal{R}_1 \tag{7}$$

where, $\mathcal{R}_1$ is a random number from the series $\mathcal{R}_1, \mathcal{R}_2, \ldots$ having a rectangular distribution in $[0; 1]$. $\nu = \infty$ for a hard sphere molecule which reduces the above expression to $g_{ab}/[g_{ab}]_{\max} > \mathcal{R}_1$. Random pairs are chosen until the

above condition is satisfied.

Once a pair satisfies the inequality in Eq. 7, the total energy of the molecules $e = e_t + e_i$ is calculated. The translational energy is $e_t = \frac{1}{2}\mu g_{ab}^2$ where $\mu = m_a m_b/(m_a + m_b)$ is the reduced mass, and the internal energy is $e_i = e_{i_a} + e_{i_b}$.

The ratio of the probability to the maximum probability of a particular value of the translational energy for a diatomic molecule is

$$P/P_{\max} = 4(e_t/e)(1 - e_t/e) \tag{8}$$

Two sets of random numbers $(\mathcal{R}_2, \mathcal{R}_3)$ are drawn until the following inequality is satisfied.

$$\{P/P_{\max} = 4(\mathcal{R}_2)(1 - \mathcal{R}_2)\} \geq \mathcal{R}_3$$

The post-collision kinetic energy, $e_t' = \mathcal{R}_2 e$ and the internal energy $e_i' = e - e_t'$ (conservation of energy) are then redistributed between the two molecules. The internal energy is divided randomly: $e_{i_a}' = \mathcal{R}_4 e_i'$ and $e_{i_b}' = e_i' - e_{i_a}'$. The post-collision velocities of the molecules are obtained in the same manner as in the case of hard sphere molecules with an updated relative velocity magnitude, $g_{ab} = \sqrt{2e_t'/\mu}$.

We use this model because of its simplicity and generality. It is easy to program and still gives excellent results.

3.4 Object-Oriented Approach

An Object-Oriented (OO) approach is used in the development of a DSMC solver for this study. It is most suitable for a particle method solver such as DSMC because the particles and cells are physical objects that have a defined set of attributes and functions. There are, of course, other benefits of using an object-oriented approach, namely, the code is reusable, easy to maintain and more organized.

The different classes used in the solver and their relationships are shown by a Unified Modeling Language (UML) diagram in Fig. 8. A very natural choice of classes is adopted: The particle class defines the properties and the actions of a particle. Since the selection of collision pairs and the sampling of properties are performed using only the particles in each cell, a cell class is defined.

The solver itself is a class which is derived from an abstract generic solver. The abstract solver simply provides the declaration of the required actions

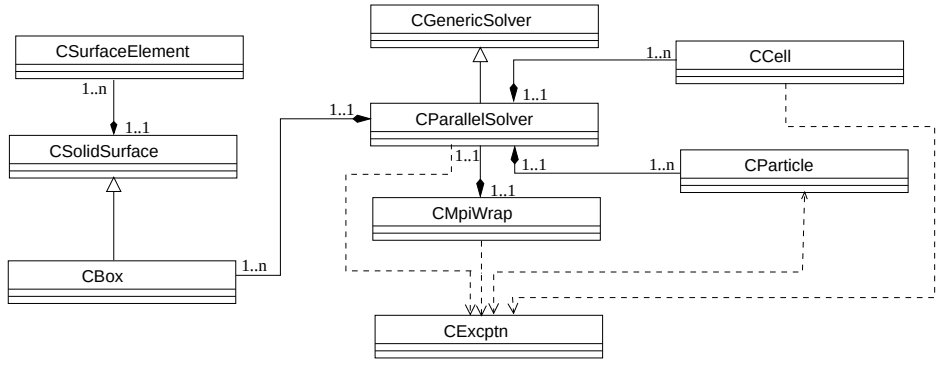


Fig. 8. A Unified Modeling Language diagram of the parallel DSMC solver.

(methods) which should be defined by any solver. The abstract class is defined to set a guideline for another user who would like to write a new solver for a different purpose. It defines the actions that are necessarily required in any DSMC solver. The solver class contains all the particles and all the cells in the domain. Each cell contains an array of pointers that point to the particles (in the solver class) that are currently in the cell. This needs to be updated every time step before the collision and sampling can be performed.

A separate class is defined for the solid body which encapsulates the collision and the bouncing of particle from the body. A solid body is stored as a collection of surface triangles. A surface element class is developed that performs the collision and the bouncing of a particle with a triangle. An array of surface elements (objects) is a private data member of the solid body class. Hence, all the details of the solid-body collision are hidden inside the solid body class which can be essentially used as a black box by the solver.

The actions of a particle such as moving and colliding with other particles are handled by appropriate member functions in the particle class whereas sorting and sampling are handled by the cell class. The solver class contains arrays of cells and particles as private members and defines global functions such as move, collide and sample which call the respective functions of either the particle or the cell class. Hence the details of how collision, sampling and move are actually performed are hidden from the user. The boundary conditions are domain dependent and hence their application is defined by the solver class directly.

A Message Passing Interface (MPI) wrapper class is developed to encapsulate the details of MPI communications. This class is a friend of the solver class to allow easy access to its private data members. This is required because MPI needs a lot of information which belongs to the solver class but is needed for inter-processor communication. The wrapper is very useful as it isolates the MPI communication calls and leaves the solver class with tidy computational routines.

It is extremely important to provide a safe exit mechanism to handle exceptional situations in a parallel program. If a runtime error develops and there is no rectification possible then all the processors running the job need to be notified to abort the job and release the computer resources they were using. An exception class is developed to deal with such situations. All the classes can throw an object of this class as an exception at any point during the program execution. The exception is caught in the MPI wrapper class which safely aborts the parallel program and generates a log of the error. The use of exceptions makes the code more robust.

The C++ solver prepared using these classes is well organized, easy to read and use, maintainable and adaptable for specific problems. It is also well documented using standard *doc++* [25] comments.

3.5 Parallelization

There are two main reasons why a parallel DSMC solver is desired: (1) The size of the problem may easily become larger than the available memory of a machine, so there is a memory constraint, and (2) a quick solution to the problem is desired, so there is also a time constraint. A very easy and obvious way to parallelize any Monte Carlo problem is to simultaneously run different ensembles on different computers and then average them. Such a program is called an 'embarrassingly parallel' program and it should ideally give a 100% speedup [26]. However, this approach is not viable for this problem because of the memory constraints.

Another approach to parallelize a program is to distribute the different functions of the program to different processors. All the processors work on the same data but perform different actions. This is called functional decomposition. This approach is also not useful for a DSMC solver because the functions in DSMC cannot be performed in parallel; they have to be executed one after the other, and again, because of the memory constraint it is imperative to distribute the data among different processors.

A better approach to parallelize the DSMC solver is the domain decomposition technique. In this procedure the domain is decomposed into sections that are processed by different processors. Each processor is given its share of cells (and corresponding particles) and this processor always works on the particles in these cells. The communication is required at the end of each time step when the particles are exchanged between neighboring processors. This approach may be combined with the embarrassingly parallel approach to perform multiple ensembles simultaneously.

Figure 9 presents a 2-D domain decomposition among 9 processors. The pro-

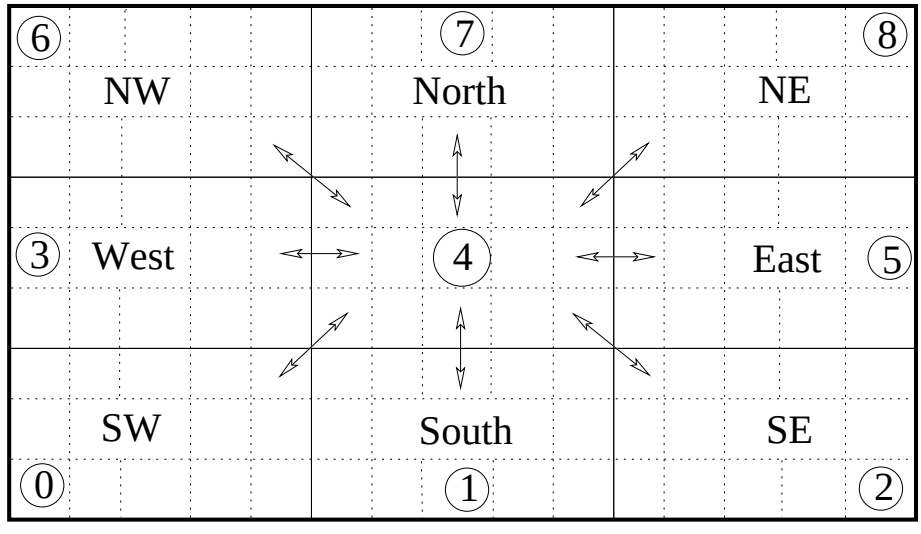


Fig. 9. A schematic showing the communication neighbors of processor 4.

cessors can communicate not only with the East, West, North and South processors but also with the North-East, South-East, North-West and South-West processors. This is illustrated in Fig. 9 for processor number 4 by dashed lines with arrows. The directions of the arrows indicate where the data is sent. Each processor can send and receive data from eight neighboring processors except when it lies on the boundary of the domain in which case it has less than eight neighbors.

A 2-D domain decomposition is useful for problems where the domain size in one direction is relatively small. However, when the problem is truly 3-D, the 2-D domain decomposition can be very inefficient. The parallel DSMC solver is therefore designed to handle 3-D domain decomposition. The user can specify the number of processors in each direction, say nx, ny and nz, then the total number of processors for the problem is equal to $nx \times ny \times nz$. In a 3-D decomposed domain, the central processor has 26 neighbors and it may have to communicate with all of them.

All the communication among the processors is performed between the move and the sort routine. There is no direct means of communicating objects of C++ in Message Passing Interface (MPI), hence an indirect approach is chosen: the object data is converted into a structure and arrays of structures are communicated. After these structures are received, they are again converted back to objects for processing. Since we are dealing with only one gas species (air), the only properties of each particle we need to communicate are: the position and velocity (3 variables each) and the internal energy (7 variables in all). The structure used for communication has seven variables (of type "double") and the conversion from object to structure and vice-versa is trivial.

A few important things to keep in mind when parallelizing a particle method solver are: Firstly, the number of particles to be communicated changes each time step, so the size of the array that is transferred has to be communicated to the corresponding processor before the particles are exchanged. Secondly, when all the communication is performed the array of objects of particles has to be rearranged to get rid of the empty spots left by the outgoing particles. The second point appears to be a minor issue but may yield frustrating segmentation faults if not done properly.

4 Results

The DSMC solver developed for this study is validated against analytical and experimental results. The Riemann shocktube problem is analytically solved for validation, and experimental results from reference [1] on a planar shock interaction with a square cavity are compared. The different aspects of the solver such as the solid and inflow boundary conditions are verified against analytical solutions of shock reflection from a solid wall, and normal shock relations.

4.1 The Riemann Problem

For the Riemann problem, the shocktube is initially divided into two chambers separated by a diaphragm. One chamber contains a hot gas at high pressure and density and the other contains a cold gas at low pressure and density. The gases in the two chambers may be different but in the present case they are the same. The diaphragm is then burst instantaneously and the hot gas (driver) is allowed to expand into the cold (driven) section. A shock wave and an expansion wave are formed which travel in opposite directions. The shock wave moves at supersonic speed into the driven section and the expansion wave travels into the driver section. Although the expansion wave travels into the driver section, the motion of the gas is always in the direction of the shock wave. A schematic of the shocktube problem with the pressure distribution both before and after the diaphragm is burst is sketched in Fig. 10. There are four distinct regions marked '1','2','3' and '4' in Fig. 10. Region '1' is the cold gas which is undisturbed by the shock wave. Region '2' contains the gas immediately behind the shock traveling at a constant speed. The 'contact surface' across which the density and temperature are discontinuous lies in this region. If two different gases are used in the driver and the driven section then the contact surface is the interface between the two gases. The region between the head and the tail of the expansion fan is marked '3'. In '3' the flow properties change gradually since the expansion process is isentropic. Region

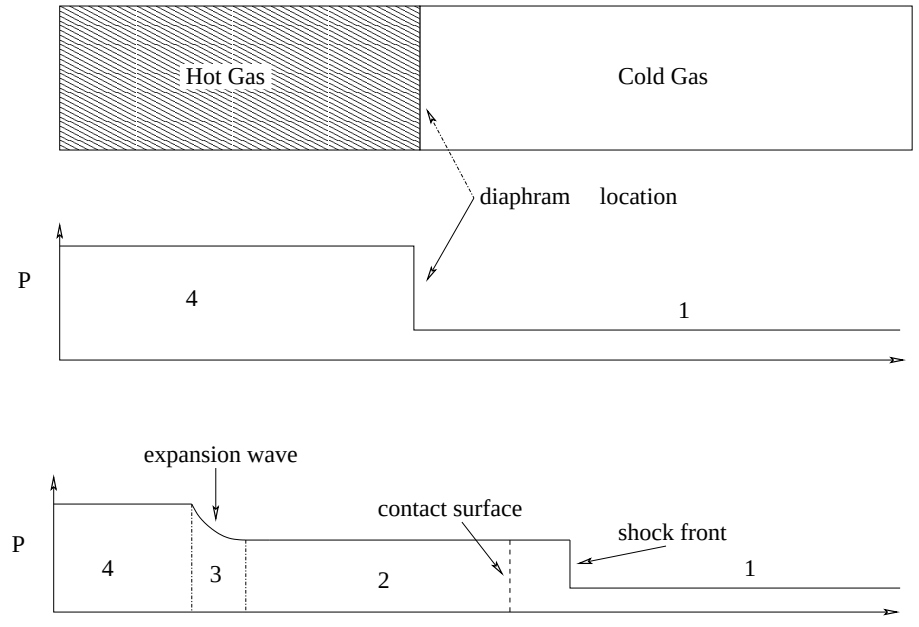


Fig. 10. A schematic of a shocktube experiment; the pressure distribution at $t = 0$ and some time after the diaphragm is burst.

'4' denotes the undisturbed hot gas. The interested reader may refer to the monologue on shock tubes by Wright [27] for further reading.

An analytical solution to the Riemann problem is available (c.f., ref. [23]). It is assumed that the ratio of the specific heats of the gas, γ, does not change with temperature (which is valid for low temperatures). The simplified equations (Eqs. 10) in region '2' are obtained using the normal shock relations.

$$
\begin{aligned}
\frac{p_4}{p_1} &= \frac{p_2}{p_1}\left\{1 - \frac{(\gamma-1)(a_1/a_4)(p_2/p_1-1)}{\sqrt{2\gamma\left[2\gamma+(\gamma+1)(p_2/p_1-1)\right]}}\right\}^{\frac{-2\gamma}{\gamma-1}} \\
u_2 &= \frac{a_1}{\gamma}\left(\frac{p_2}{p_1}-1\right)\left(\frac{(2\gamma/(\gamma+1)}{p_2/p_1+(\gamma-1)/(\gamma+1)}\right)^{1/2} \\
\frac{T_2}{T_1} &= \frac{p_2}{p_1}\left(\frac{(\gamma+1)/(\gamma-1)+p_2/p_1}{1+(p_2/p_1)(\gamma+1)/(\gamma-1)}\right) \\
p_2 &= \rho R T_2
\end{aligned}
\tag{9}
$$

The velocity of the gas in region '2' is constant throughout and is equal to u_2. The method of characteristics yields the solution in region '3' ($-a_4 \leq x/t \leq u_3 - a_3$).

$$
u_3 = \frac{2}{\gamma+1}\left(a_4 + \frac{x}{t}\right)
$$

$$\frac{p_3}{p_4} = \left[1 - \frac{\gamma - 1}{2}(u_3/a_4)\right]^{2/(\gamma-1)} \qquad (10)$$
$$\frac{p_3}{p_4} = (\rho_3/\rho_4)^{\gamma} = (T_3/T_4)^{\gamma/\gamma-1}$$

A setup similar to that used in an experiment is used for the numerical calculations. The flow properties are averaged in the 'y' direction as this is a one dimensional problem. A sample comparison is provided in Fig. 11 for the following set of values in SI units:

$$\rho_1 = 1.226 \qquad \rho_4 = 4.226 \qquad T_1 = 300 \qquad T_4 = 900$$

Figure 11 plots the density and pressure profiles in the shocktube at some instant of time after the diaphragm explodes. The DSMC calculations capture the sharp gradients and give an excellent overall match with the analytical solution. The DSMC results show a slight smoothing at the contact surface (refer Fig. 11 (a)) but it is not of critical importance for the problem of interest and is observed in conventional CFD schemes as well.

4.2 Inflow Boundary Condition

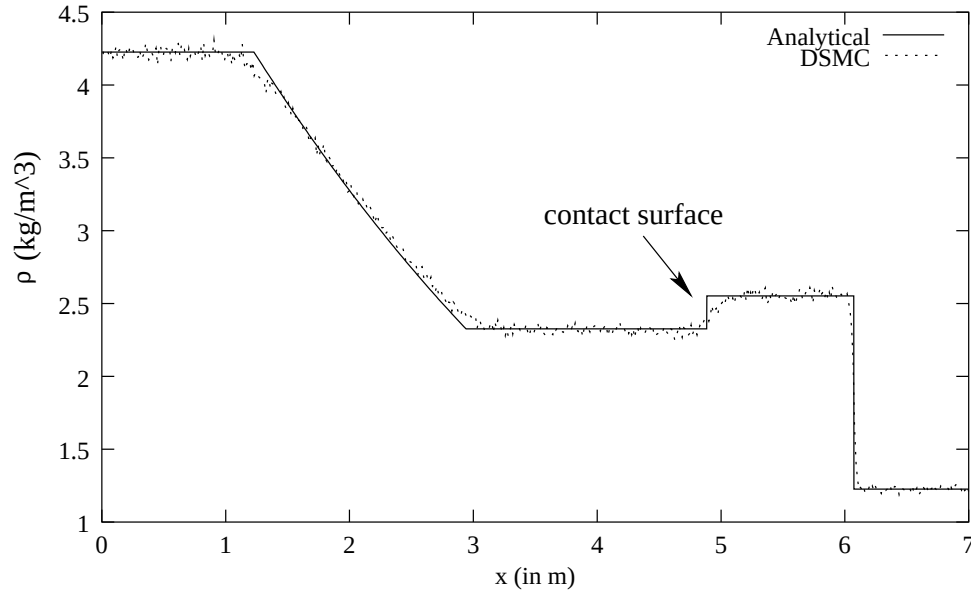
The simple approach to model the inflow boundary condition used in this study gives excellent results. The Riemann problem is used with inflow boundary conditions imposed on the left 'x' boundary. The simulation starts with the shock wave on the left boundary that moves into the domain with time. A comparison with the analytical normal shock relations is provided in Fig. 12. The following set of conditions are used for the comparison.

$$\rho_1 = 1.0091 \qquad T_1 = 293.15 \qquad M_s = 1.43$$

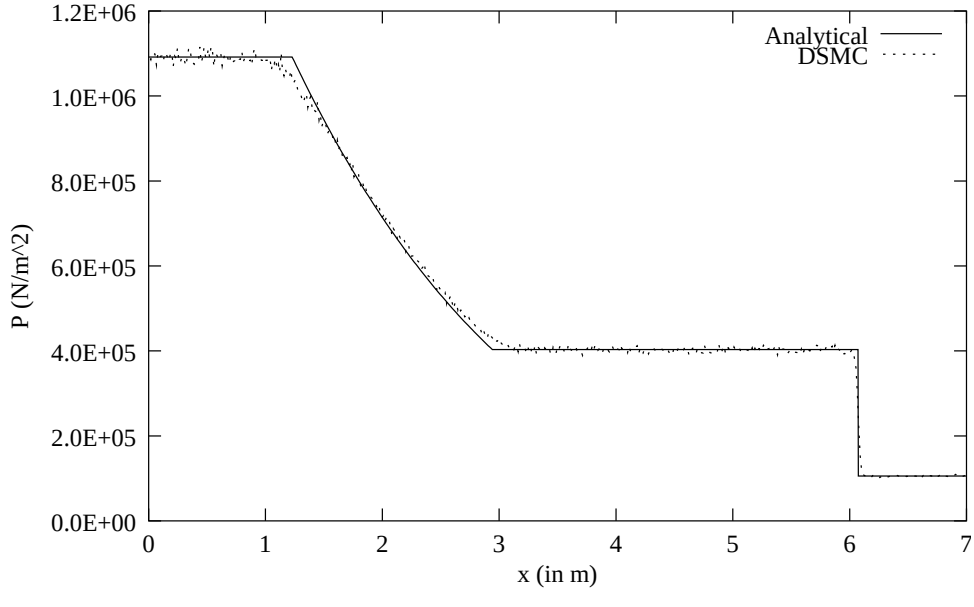
The inflow boundary should not be placed arbitrarily close to the body in a blast-impact study as the reflected shock may change the conditions on the boundary.

4.3 Solid Boundary Condition

The solid boundary condition implementation is validated against the analytical solution for a normal shock wave reflecting off a solid wall. A solid box is placed at the end of the shocktube such that one face of the box is inside and



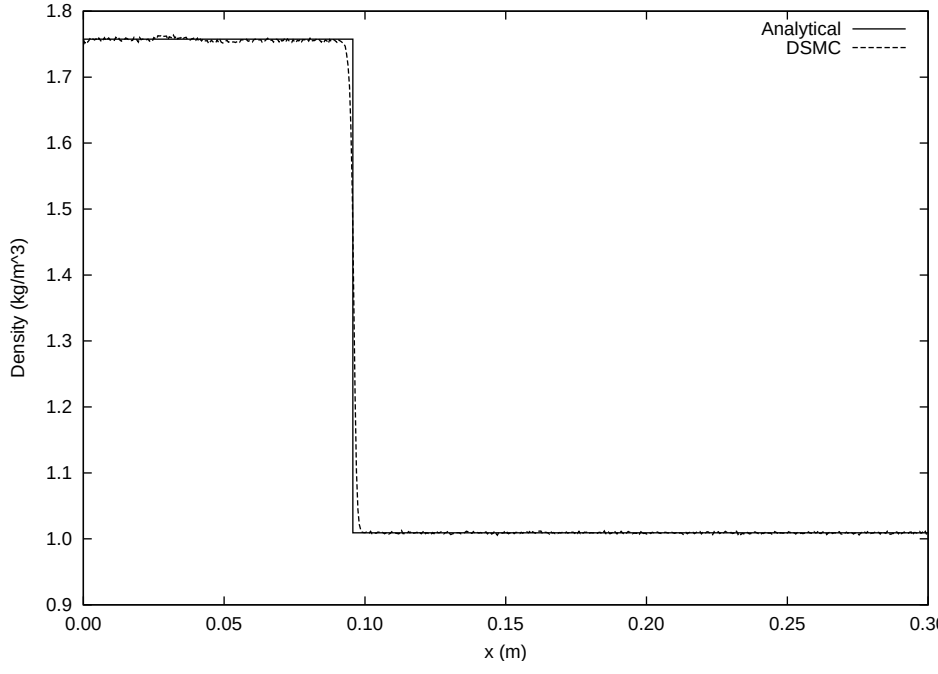
(a) Density



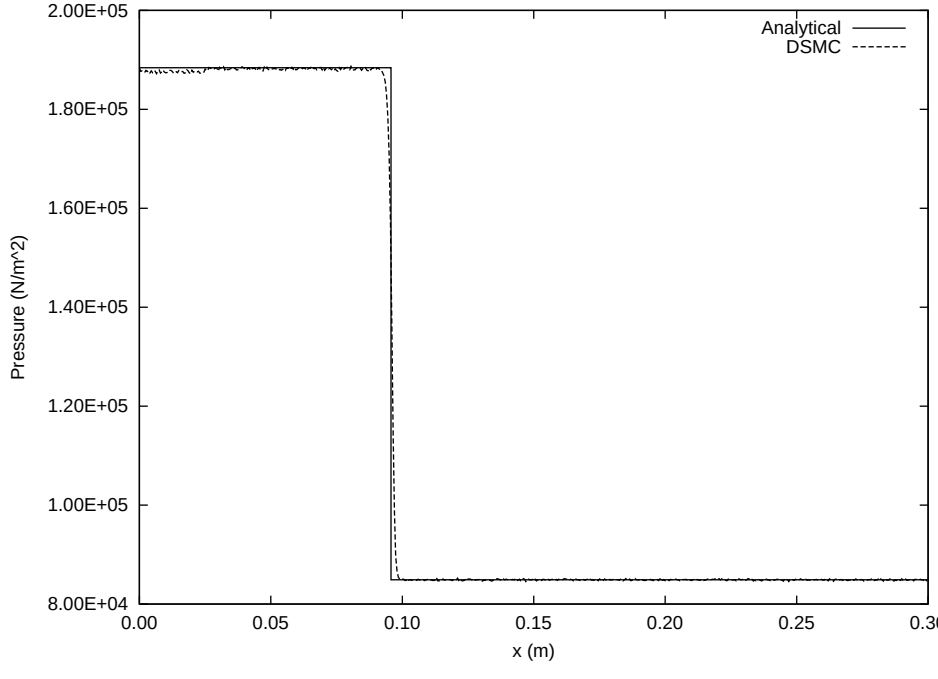
(b) Pressure

Fig. 11. DSMC and analytical results for the shocktube problem at t=3.6 ms. (a) density and (b) pressure.

blocks the entire cross-section of the shocktube (see Fig. 4.3). Each wall of the solid box is made of two triangles made by a diagonal and remaining edges. If fact, just one wall could be used to do this numerical experiment. Inflow boundary condition is used on the left boundary of the domain. The comparison of the results against the analytical solution is presented in Fig. 14. The



(a) Density



(b) Pressure

Fig. 12. Verification of the inflow boundary condition. Comparison with the analytical solution of the shock-tube problem. (a) density and (b) pressure.

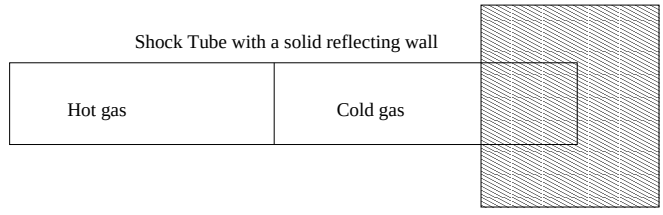


Fig. 13. A schematic of the setup of the numerical experiment to test the solid boundary condition implementation.

results match very closely with the analytical solution. The following set of conditions are used for the comparison.

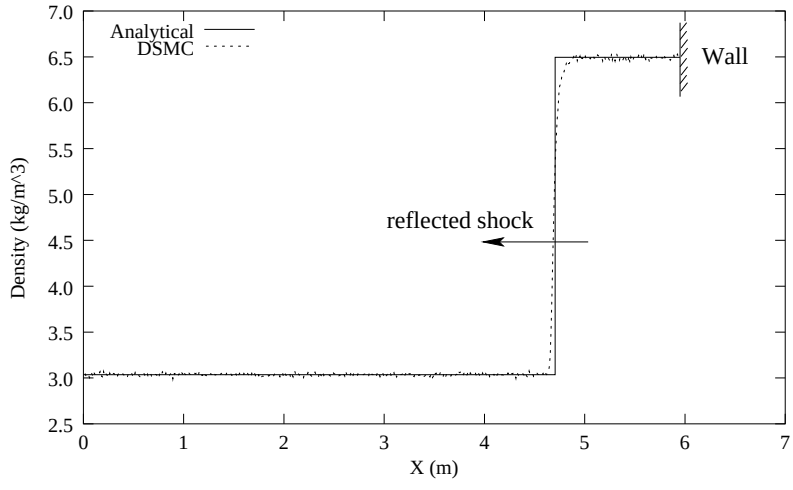
$$\rho_1 = 1.226 \qquad T_1 = 300.0 \qquad M_s = 1.87$$

4.4 Planar Shock Interaction with a Square Cavity

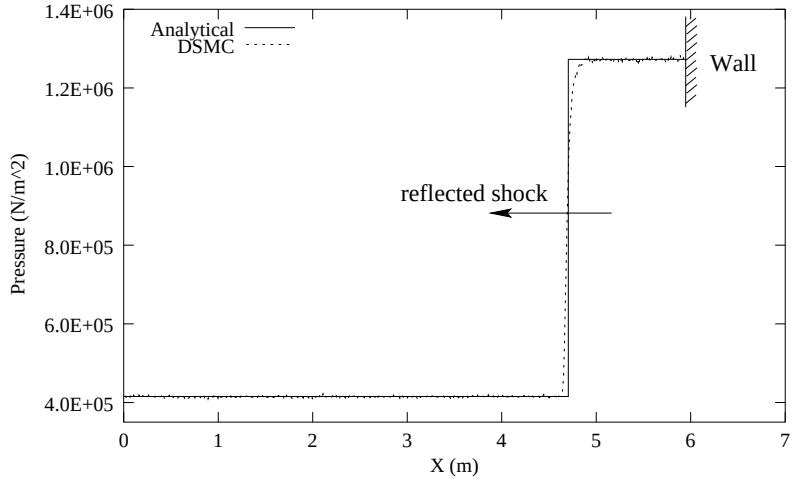
Reichenbach *et al.* [1] experimentally and numerically investigated the interaction of a planar shock wave with a square cavity. They performed experiments in a shocktube designed for using 'two-dimensional' models [1]. Reference [1] presented Shadowgraphs at different times during the experiments for two incident shock wave Mach numbers, 1.3 and 2.032. They also presented a quantitative comparison of their numerical result with the experiments for first peak overpressures on the cavity walls for different shock wave Mach numbers. A schematic description of the flowfield considered prior to the arrival of the incident shock front at the cavity is provided in Fig. 15.

DSMC simulations were performed to compare against the experimental results of reference [1]. A qualitative comparison is made in Fig. 16 for incident shock wave Mach number, $M_s = 1.43$ and Fig. 17 for $M_s = 2.032$. A very good qualitative match is obtained in both cases. In Fig. 16 the flow behind the shock wave is subsonic and hence a vortex is formed near the cavity's upper-left corner. However, in the second case (Fig. 17), the post-shock flow is sonic, and hence an expansion fan centered at the corner is observed. Figure 18 compares the DSMC predictions against the experimental measurements of the first peak overpressures at the three tap locations on the cavity walls. The simulation results accurately match the experimental data.

The grid used for the cavity simulations is made of 800×440 cells with 40 particles in each cell at the start of the simulation. Around 1200 samples were collected for the results.



(a) Density



(b) Pressure

Fig. 14. Comparison of the DSMC results against the analytical solution of the reflection of a normal shock wave from a solid wall (a) density and (b) pressure.

4.5 Blast Impact Simulations

Two model shapes are tested for blast impact using the DSMC solver. These are a box and an 'I' shaped beam (I-beam). The I-beam is chosen because it is a concave geometry which is challenging to model. Figures 19 and 20 are schematics to show the domain size and the locations at which pressure history is recorded, for the box and I-beam simulations. Inflow boundary conditions

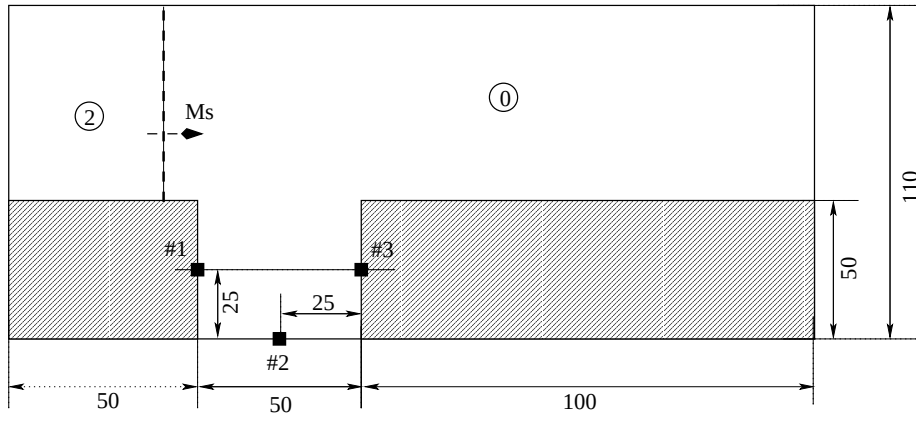


Fig. 15. A schematic showing the dimensions of the cavity and the pressure transducer locations. Shaded parts are modeled as boxes using the solid boundary condition implementation. All dimensions are in millimeters.

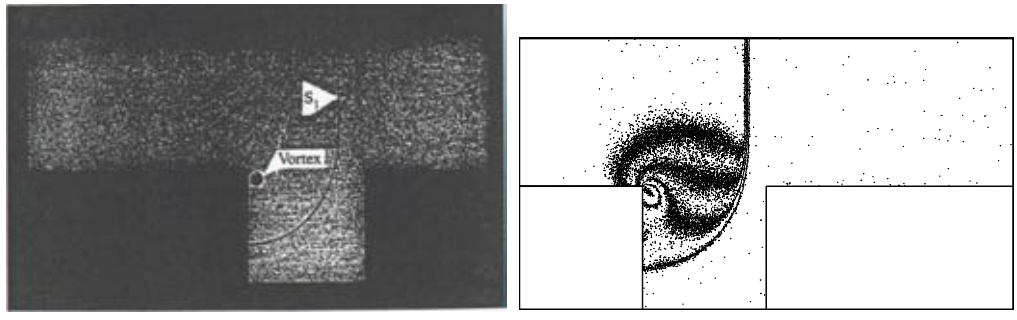
are applied on the left wall and the shock wave is placed close to the solid body at the start of the simulations as shown in Figs. 19 and 20. Wall boundary condition is used on the top and bottom walls. The following set of initial conditions (in SI units) are used in both simulations.

$$\rho_1 = 1.14 \qquad T_1 = 196.45 \qquad M_s = 1.98$$

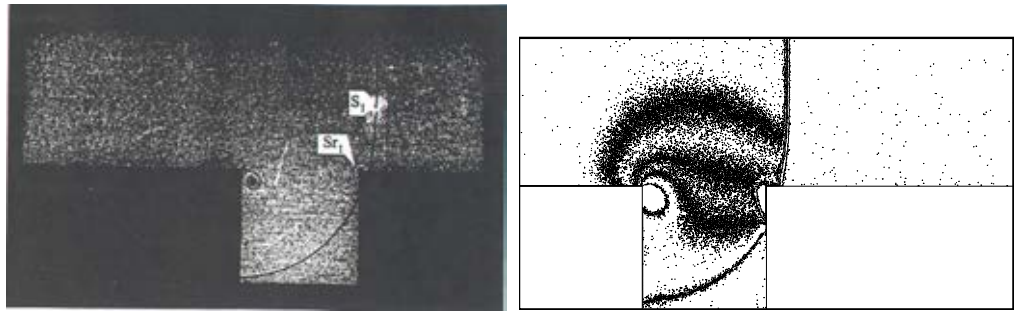
Figures 21 are isopycnic plots of the simulation at different times during the blast impact simulation of a box. The part of the incident shock front that hits the front face of the box is reflected back. The reflected shock is normal in the center and curves near the top and bottom edges of the box as it diffracts into 2-D space. The reflected shock later hits the top and bottom walls. The shock fronts above and below the box travel along the edges and diffract at the rear edge of the box. The overpressure history at the four points shown in Fig. 19 is plotted in Fig. 22. Since the problem is symmetric about the centerline ($x = 38.5$), the locations 3 and 4 have identical pressure signatures.

In the I-beam case, the shock front reflects from the two flanges and from the main spar of the I-beam resulting in a little more complicated shock pattern. Figures 23 are isopycnics showing the interaction of a planar shock with the I-beam at different times. Figure 24 plot the overpressure history at the eight locations marked in Fig. 20. The symmetry of the problem results in overlapping pressure signatures for location pairs 3 and 4, 5 and 7, and, 6 and 8.

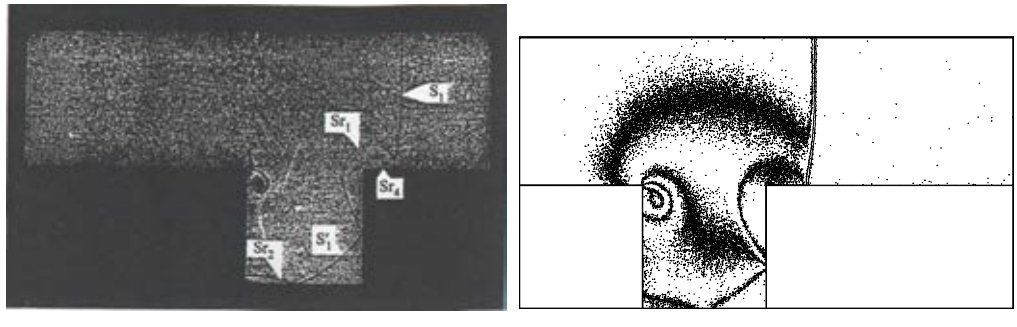
The grid used for the blast impact simulations has 600×300 cells with 60 particles per cell at the start of the simulation. The results presented are obtained using 1800 samples.



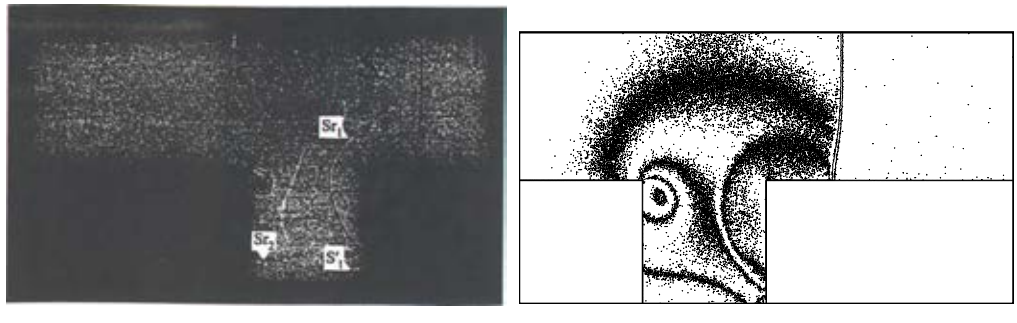
(a) t = 100 μs



(b) t = 140 μs

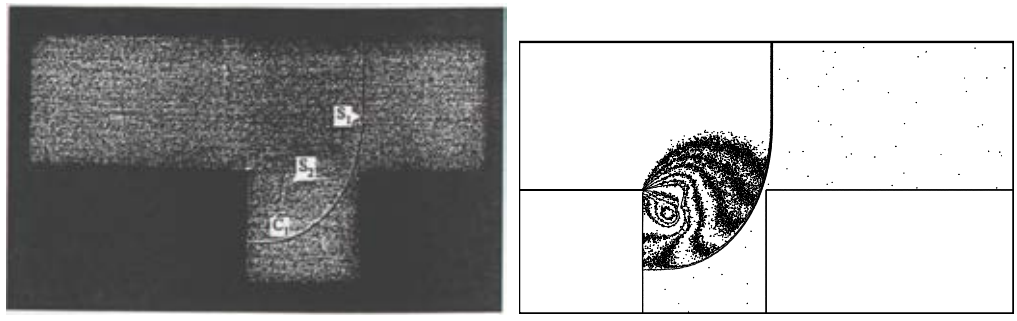


(c) t = 160 μs

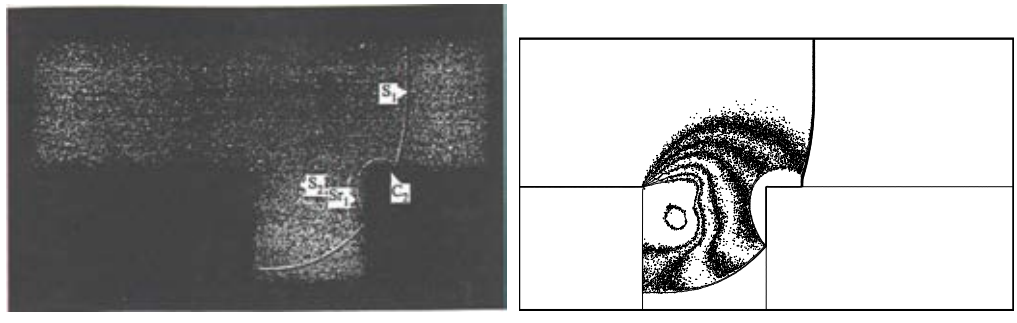


(d) t = 180 μs

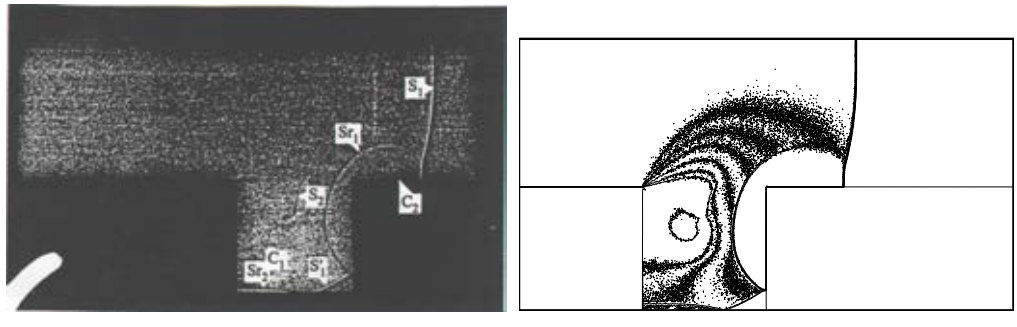
Fig. 16. The interaction of a traveling planar shock wave (Mach number 1.43) with a square cavity at different times. On the left are Shadowgraphs from Ref. [1], and on the right are isopycnic plots from DSMC simulations.



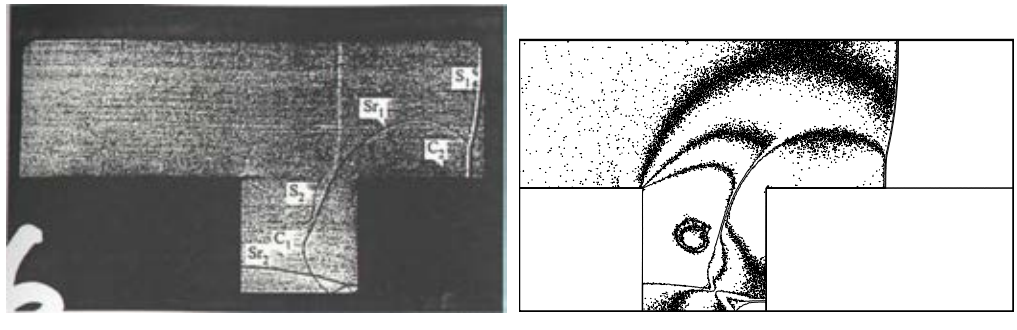
(a) t = 75 μs



(b) t = 100 μs



(c) t = 125 μs



(d) t = 150 μs

Fig. 17. The interaction of a traveling planar shock wave (Mach number 2.032) with a square cavity at different times. On the left are Shadowgraphs from Ref. [1], and on the right are isopycnic plots from DSMC simulations.

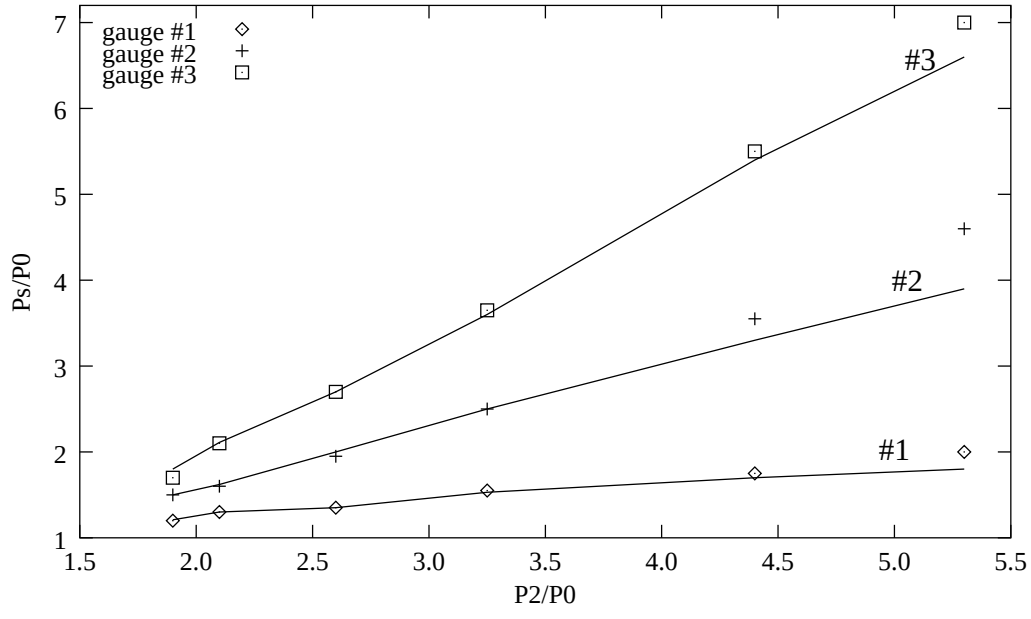


Fig. 18. First pressure peaks on each of the cavity walls. Solid lines represent DSMC results and symbols represent experimental data.

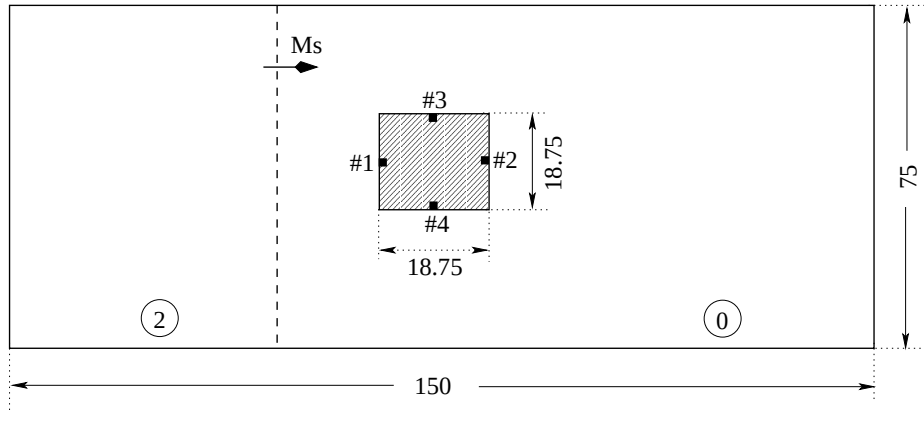


Fig. 19. A schematic of the blast impact simulation on a box. The box is located in the center of the domain. Pressure history is collected at four locations marked by 1, 2, 3 and 4 in the figure. All dimensions are in mm.

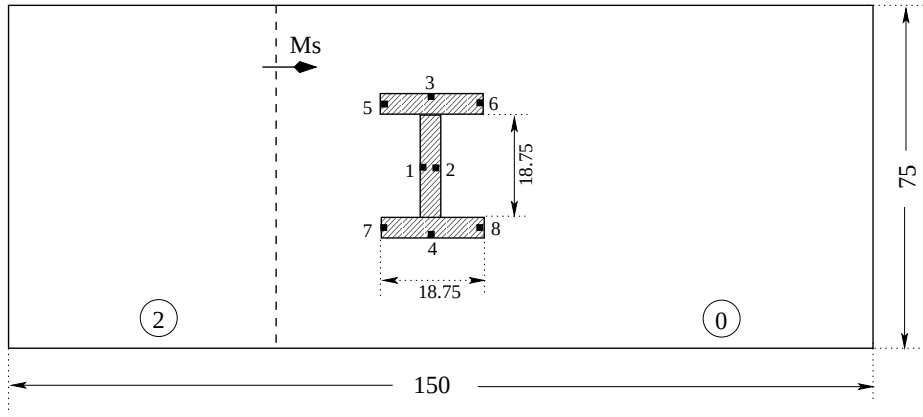
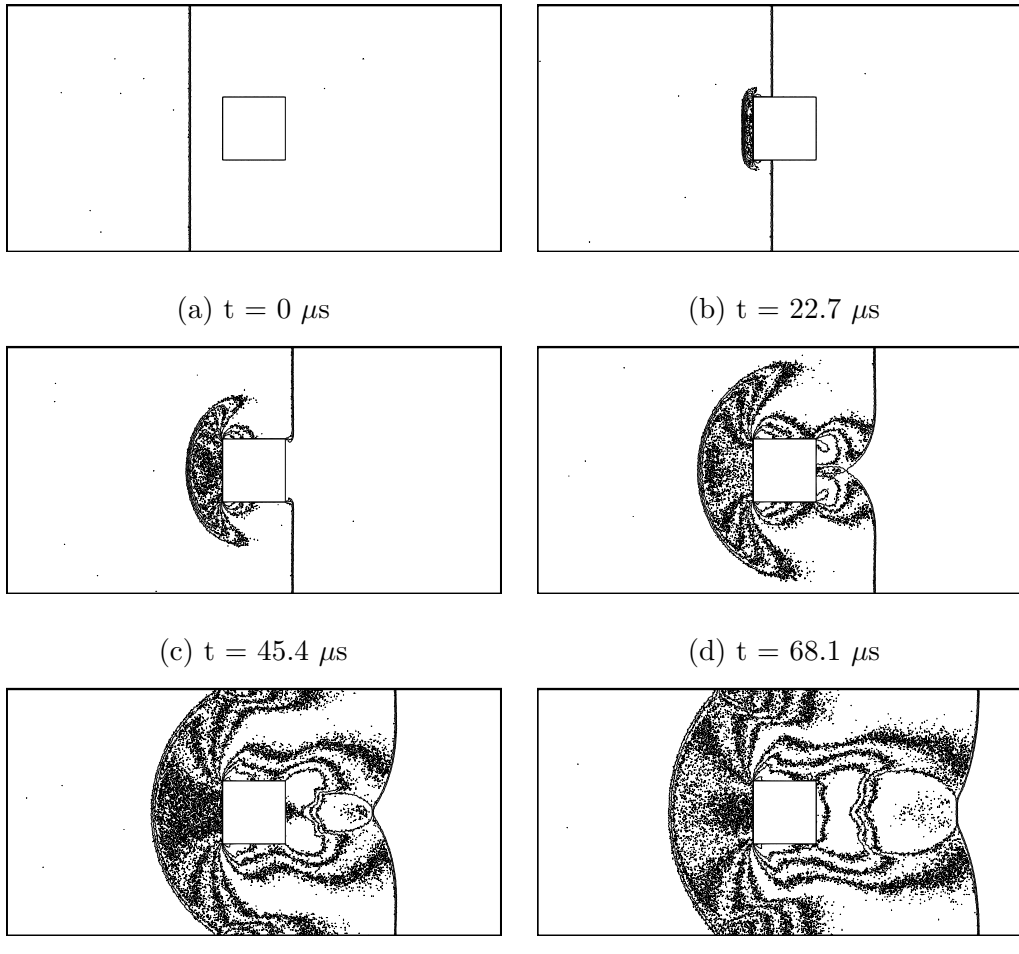


Fig. 20. A schematic of the blast impact simulation on an I-beam. The I-beam is located in the center of the domain. Pressure history is collected at 8 locations as shown. The I-beam has a uniform thickness of 3.75 mm. All dimensions are in mm.



(a) t = 0 μs (b) t = 22.7 μs

(c) t = 45.4 μs (d) t = 68.1 μs

(e) t = 90.8 μs (f) t = 113.5 μs

Fig. 21. Isopycnic plots at different times for the blast impact simulation of a box.

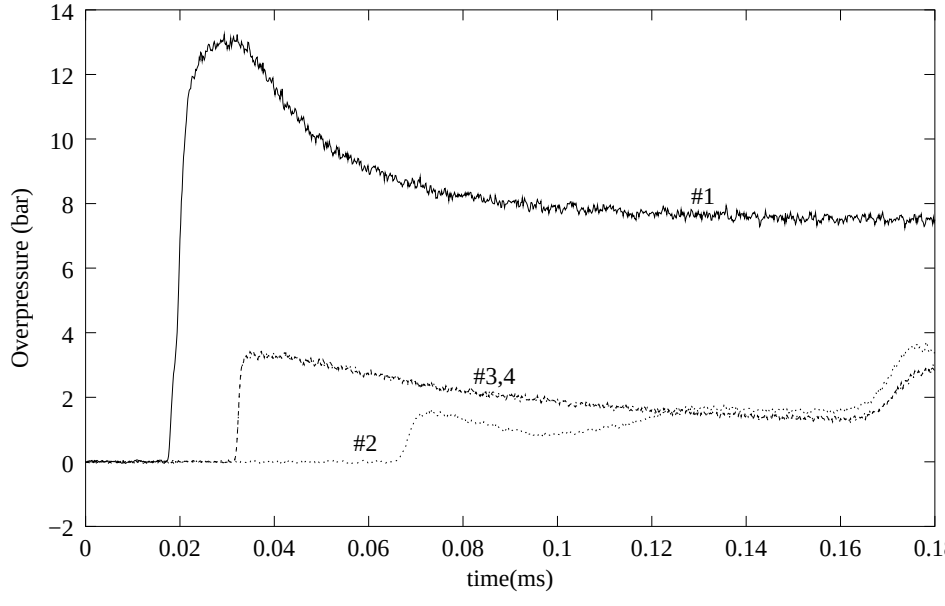
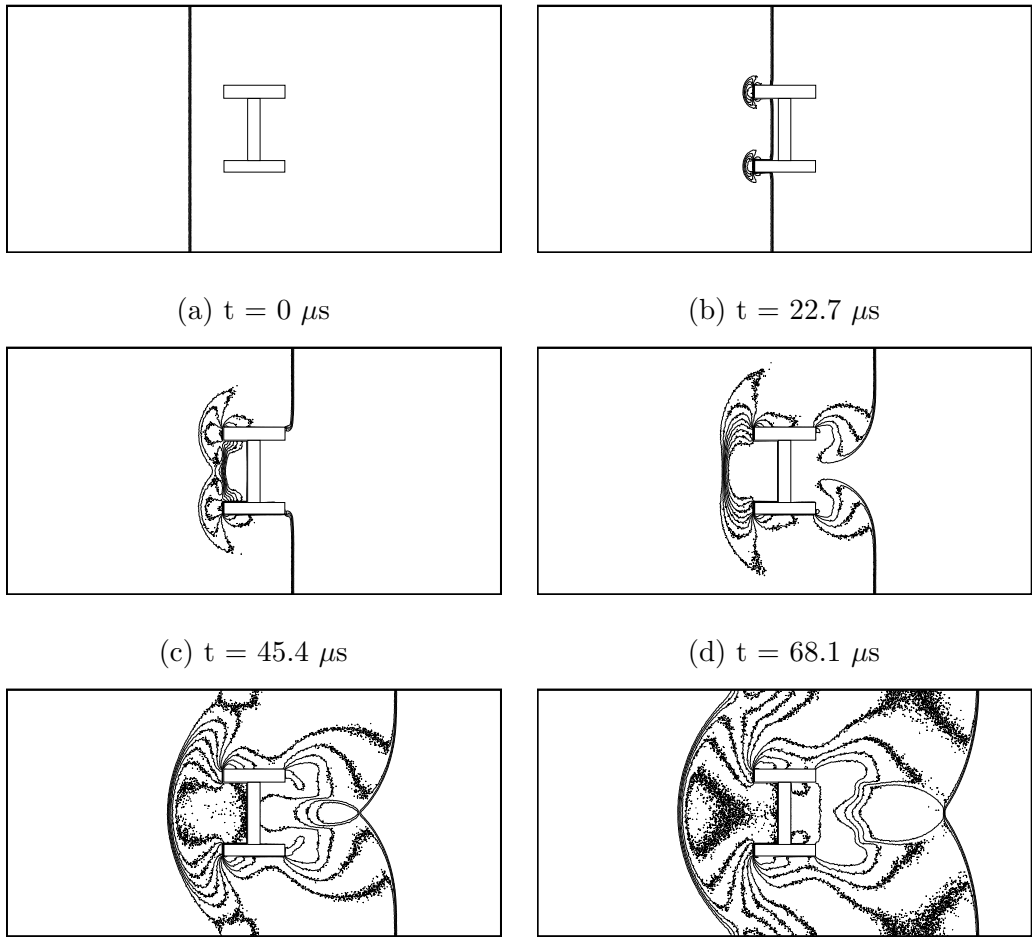


Fig. 22. Overpressure history at the four locations marked in Fig. 19. Data collected at every 0.15 μs.



(a) t = 0 μs (b) t = 22.7 μs

(c) t = 45.4 μs (d) t = 68.1 μs

(e) t = 90.8 μs (f) t = 113.5 μs

Fig. 23. Isopycnic plots at different times for the blast impact simulation of an I-beam. All times are in μs.

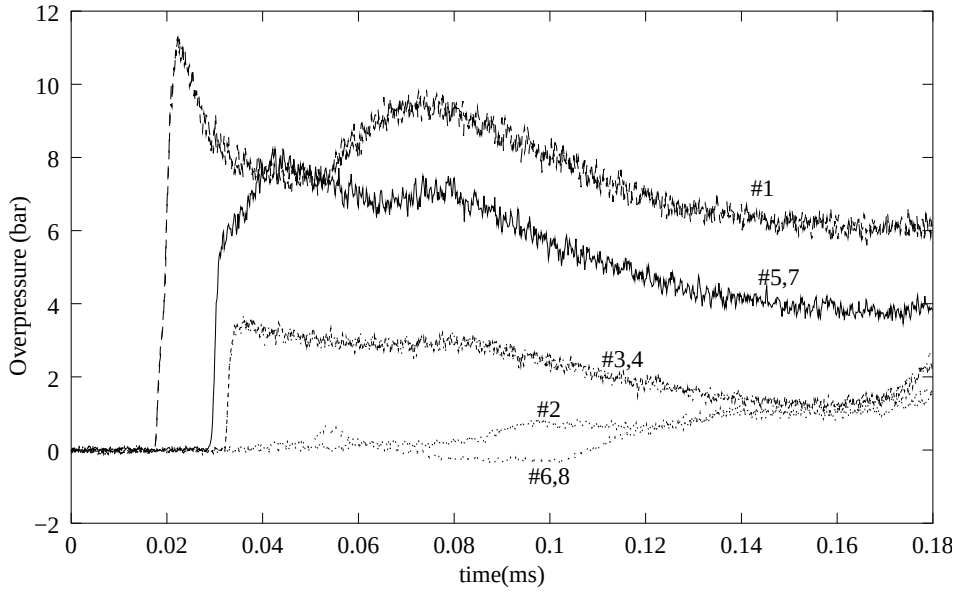


Fig. 24. Overpressure history at the eight locations marked in Fig. 20. Data collected at every 0.15 μs.

4.6 Parallel Performance

The Riemann shocktube, 'shock interactioni with a cavity' and the 'blast impact on a box' problems were solved for sample sizes to measure the parallel performance of the solver. The performance measurements are conducted on two Beowulf clusters, COCOA2 and COCOA3. COCOA2 is a 20 node dual 800 MHz Pentium III processor cluster with two 100 Mbit Network Interface Cards (NICs) per node bonded together for interprocessor communication. COCOA3 is a 60 node dual 2.4GHz Xeon processor cluster with one 1 Gbit NIC per node. Both the clusters run open-source Redhat linux operating system. The solver was compiled using the open-source GNU compiler, *g++* with the optimization flag *-O*.

The time taken to perform the key functions in the solver are tabulated in Table 1. The size of the problem is deliberately reduced to allow solution by a single processor. The performance may differ if the domain is decomposed in a different fashion. It is optimized for maximum load balancing for the cases presented here. Figure 25 plots the speedup and the parallel efficiencies obtained for the three problems in Table 1. Appreciable speedup is obtained even with a small problem size. Better performance is expected for larger problem sizes.

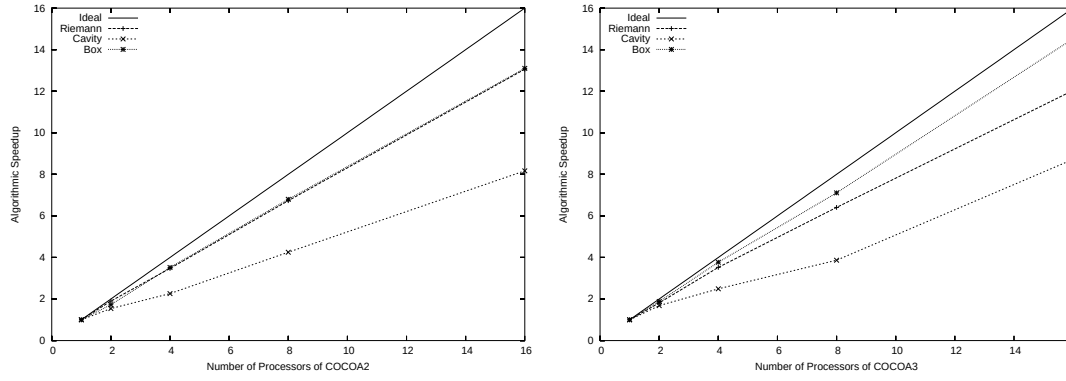
Table 1
Listing of the three sample problems solved on COCOA3 for performance testing.

Case	Timing(s)					
	Procs	***Move***	***Collide***	***Sample***	***Comm.***	***Total***
Riemann	1	999.17	6923.57	30.16	0.00	11278.3
Shocktube	2	612.58	3404.70	17.32	34.23	6213.3
Cells: 600*100*1	4	327.00	1675.02	8.88	31.83	3201.3
Part: 3 mill.	8	182.96	896.70	4.63	99.82	1760.2
	16	85.30	422.84	2.33	158.05	936.7
Shock Interaction	1	1780.77	4442.40	21.09	0.00	8640.9
With a Cavity	2	1057.07	2543.21	12.69	8.32	5137.4
Cells: 400*220*1	4	279.11	828.78	3.19	1968.94	3470.2
Part: 4.4 mill.	8	43.97	75.95	0.45	2062.24	2233.9
	16	0.31	0.40	0.03	985.49	990.8
Blast Impact	1	1243.73	6251.45	20.48	0.00	9781.5
With a Box	2	701.84	3117.28	11.62	29.25	5204.4
Cells: 400*220*1	4	318.60	1499.78	5.58	133.32	2592.4
Part: 4.4 mill.	8	159.68	774.33	2.79	120.22	1375.1
	16	76.99	381.08	1.27	60.05	673.7

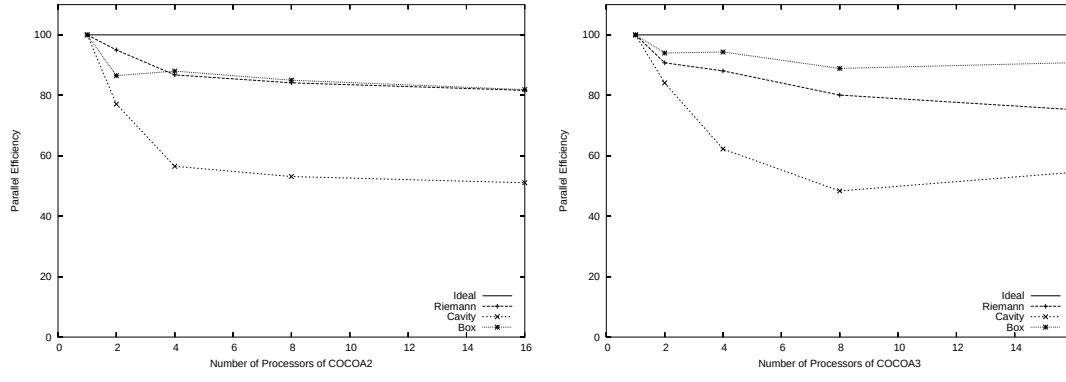
5 Conclusions

The problem of blast wave interaction with structures is studied using the Direct Simulation Monte Carlo approach. A new approach to model the solid boundary condition for complex geometries is described and implemented with success. The Object Oriented approach used for the development of the solver is described. The code is made parallel using the domain decomposition technique to solve large, complex problems. Different aspects of the solver are individually validated against analytical results for benchmark problems. Reflection of a normal shock from a wall is used to validate the solid boundary condition and normal shock relations are used to validate the inflow boundary condition. The Riemann shocktube problem and experimental results from an experiment on planar shock interaction with a cavity (ref. [1]) are used to validate the solver. Both qualitative and quantitative comparisons are made for the shock-cavity interaction problem. The numerical results are found to be in excellent agreement with the experiments.

The problem of blast impact is studied on two model shapes: a box and an I-beam. Pressure signatures at a few locations are plotted to provide a quantitative measure of the loading. Isopycnic plots showing the flow structure are



(a) Algorithmic Speedup



(b) Parallel Efficiency

Fig. 25. The performance of the parallel program for three test cases tabulated in Table 1 on COCOA2 (left) and COCOA3 (right).

presented at different times. Lastly, the parallel performance of the solver is estimated on two Beowulf clusters for three problems.

References

[1] O. Igra, J. Falcovitz, H. Reichenbach, W. Heilig, Experimental and Numerical Study of the Interaction between a Planar Shock Wave and a Square Cavity, Journal of Fluid Mechanics 313 (1996) 105–130.

[2] G. A. Bird, Molecular Gas Dynamics, Clarendon Press, Clarendon, Oxford, 1976.

[3] E. P. Muntz, Rarefied Gas Dynamics, Annual Review of Fluid Mechanics 21 (1989) 387–417.

[4] E. Salomons, M. Mareschal, Usefulness of the Burnett Description of Strong Shock Waves, Physical Review Letters 69 (2) (1992) 269–272.

[5] G. A. Bird, Monte Carlo Simulation of Gas Flows, Annual Review of Fluid Mechanics 10 (1978) 11–31.

[6] G. A. Bird, Molecular Gas Dynamics and the Direct Simulation of Gas Flows, 2nd Edition, Oxford Science Publications, Clarendon, Oxford, 1994.

[7] D. I. Pullin, J. Davis, J. K. Harvey, Monte Carlo Calculations of the Rarefied Transition Flow Past a Bluff Faced Cylinder, in: 10th International Symposium on Rarefied Gas Dynamics, Aspen, Colorado, 1976.

[8] E. S. Oran, C. K. Oh, B. Z. Cybyk, Direct Simulation Monte Carlo : Recent Advances and Application, Annual Review of Fluid Mechanics 30 (1998) 403–441.

[9] G. A. Bird, Direct Simulation of the Boltzmann Equation, Physics of Fluids 13.

[10] L. N. Long, Navier Stokes and Monte Carlo Results for Hypersonic Flows, AIAA Journal 29 (2).

[11] J. B. Anderson, L. N. Long, Direct Monte Carlo Simulation of Chemical Reaction Systems: Prediction of Ultrafast Detonations, The Journal of Chemical Physics 118 (7) (2003) 3102–3110.

[12] L. N. Long, J. B. Anderson, The Simulation of Detonations using a Monte Carlo Method, in: Rarefied Gas Dynamics Conference, Sydney, Australia, 2000.

[13] J. B. Anderson, L. N. Long, Direct simulation of pathological detonations, in: 18th International Symposium on Rarefied Gas Dynamics, Vancouver, Canada, 2002.

[14] S. M. Dunn, J. B. Anderson, Direct Monte Carlo Simulation of Chemical Reaction Systems: Internal Energy Transfer and an Energy-Dependent Unimolecular Reaction, The Journal of Chemical Physics 99 (9) (1993) 6607–6612.

[15] S. M. Dunn, J. B. Anderson, Direct Monte Carlo Simulation of Chemical Reaction Systems: Dissociation and Recombination, The Journal of Chemical Physics 102 (7) (1995) 2812–2815.

[16] W. Wagner, A Convergence Proof for Bird's Direct Simulation Monte Carlo Method for the Boltzmann Equation, Journal of Statistical Mechanics 66 (3/4).

[17] D. I. Pullin, Direct Simulation Methods for Ideal-Gas Flow, Journal of Computational Physics 34 (1980) 231–244.

[18] C. L. Merkle, H. William Behrens, Robert D. Hughes, Application of the Monte-Carlo Simulation Procedure in the Near Continuum Regime, in: Twelfth Symposium on Rarefied Gas Dynamics, 1980.

[19] A. Sharma, L. N. Long, T. Krauthammer, Using the Direct Simulation Monte Carlo Approach for the Blast-Impact Problem, in: The 17th International Symposium on Military Aspects of Blast Simulations, Las Vegas, Nevada, 2002.

[20] A. L. Danforth, L. N. Long, Acoustic Propagation using the Direct Simulation Monte Carlo Method, The Journal of the Acoustical Society of America 114 (4) (2003) 2356–2357.

[21] N. G. Hadjiconstantinou, A. L. Garcia, Molecular Simulations of Sound Wave Propagation in Simple Gases, Physics of Fluids 13 (2001) 1040–1046.

[22] I. E. Sutherland, R. F. Sproull, R. Shumacker, A Charaterizarion of Ten Hidden Surface Algorithms, ACM Computing Surveys 6 (1) (1974) 1–55.

[23] J. D. Anderson Jr., Modern Compressible Flow with Historical Perspective, 2nd Edition, McGraw Hill Professional Publishing, 1989.

[24] Claus Borgnakke, Poul S. Larsen, Statistical Collision Model for Monte Carlo Simulation of Polyatomic Gas Mixture, Journal of Computational Physics 18 (1975) 405–420.

[25] Doc++ homepage:, http://docpp.sourceforge.net.

[26] L. N. Long, K. S. Brentner, Self-Scheduling Parallel Methods for Multiple Serial Codes with Applications to wopwop, in: 38th Aerospace Sciences and Meeting Exhibit, Paper 0346, Reno, Nevada, 2000.

[27] J. K. Wright, Shock Tubes, John Wiley & Sons INC, 1961.