16th AIAA Computational Fluid Dynamics Conference
23-26 June 2003, Orlando, Florida

AIAA 2003-4234

# A PARALLEL, OBJECT-ORIENTED DIRECT SIMULATION MONTE CARLO METHOD FOR BLAST-IMPACT SIMULATIONS

Anupam Sharma* and Lyle N. Long†
*Department of Aerospace Engineering*
*The Pennsylvania State University*
*University Park, PA 16802, U.S.A.*



**A parallel, object oriented Direct Simulation Monte Carlo (DSMC) solver is developed to study the impact of blast waves on arbitrary-shaped bodies. A very natural choice of classes is adopted to design the solver. A 3-D domain decomposition is used to parallalize the code. The Message Passing Interface (MPI) library is used to perform interprocessor communications. Preliminary results of two simulations using the parallel solver are presented. A test case to validate the solid boundary condition on the solid bodies is successfuly tested. A significant speedup is observed for three sample problems tested for parallel performance.**

## 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 "havens" and barracks respectively 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.

The 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 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 which cannot be accurately determined anyway, slight inaccuracy in the results is acceptable if it significantly

*Ph.D. Candidate, axs455@psu.edu
†**Professor, Assoc Fellow, AIAA.** lnl@psu.edu

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, the size of the problem may become larger than the memory on a single PC, and secondly, the parallelization is expected to reduce the turnaround time.

The solver has been verified against the analytical solution of the Riemann Shocktube problem and is found to give excellent results.' This paper presents the Object Oriented approach and the parallel a p proach used to develop the solver. The solid boundary condition used on the solid bodies tested for blast impact is validated. The problem of a normal shock wave reflecting off a wall is solved for the validation. Preliminary results from two numerical simulations are presented and compared against the experiments: a shock wave traveling over a square cavity (experiments conducted by Igra *et al.*[2]), and a balloon blast inside a box (experiments conducted by Settles *et al*[3]).

## DSMC

The Direct Simulation Monte Carlo (DSMC) method has gained a lot of popularity since its development by Bird[4] 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[5] and Molecular-Dynamics computations.[6] The method was developed primarily for aerospace applications in rarefied gas dynamics and has been extensively applied in that area.[5,7,8,9,10] DSMC has also been successfully used for hypersonic flows.[11,8,12] 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.[13,14]

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 are moved 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

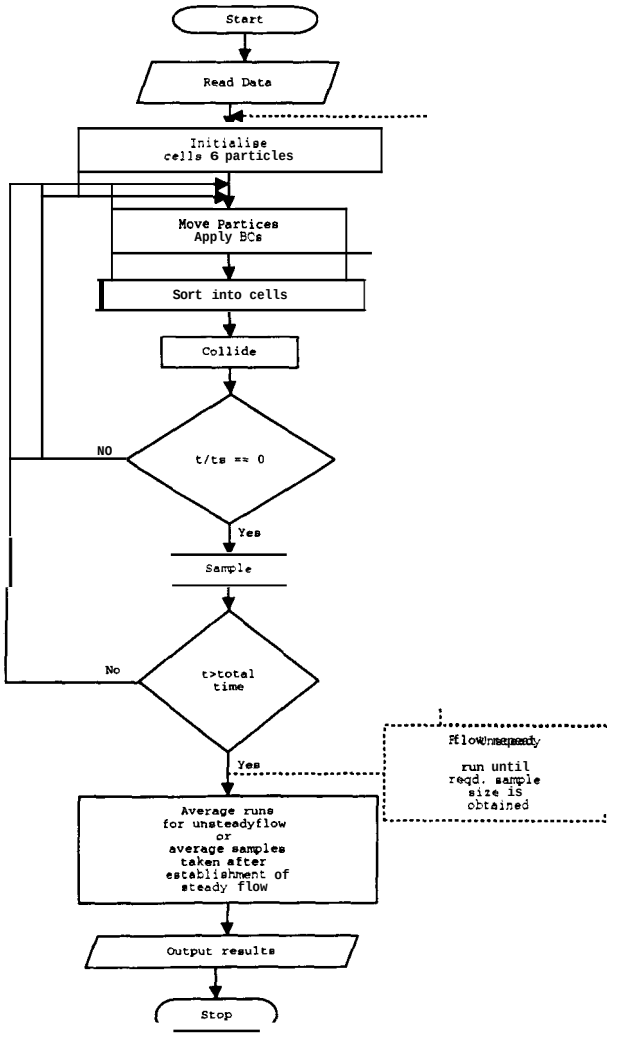



**Fig. 1 A Flowchart of a typical DSMC simulation.**

of discrete steps of fixed length) while conserving the momentum and energy. Hence, the collisions in DSMC are only statistically correct. The treatment of intermolecular collisions is the key difference between the Molecular Dynamics and the 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 ***At*** assuming no intermolecular collisions but treat the collisions with the domain boundaries, (b) sort the particles in the different cells, (c) perform intermolecular collisions, and (d) sample macroscopic properties of interest.

The DSMC simulations become more exact **as** the time step and the cell size tend to zero. In fact, Wagner[15] 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 go 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-F'redrichs-Lewy (CFL) condition.

There have been few attempts[16,17] 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[16] 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.[16] This is analogous to solving the Euler equations. In his algorithm, Pullin reinitializes the velocities of the molecule to the local Maxwellian after every iteration thus insuring local equilibrium. Pullin[16] 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[16] 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[1] have proposed an inexpensive procedure to achieve local thermodynamic equilibrium. A significant reduction in the computation time by this procedure makes DSMC a practical Euler 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.'' The interested reader is referred to Bird' for detailed description of DSMC and its applications.

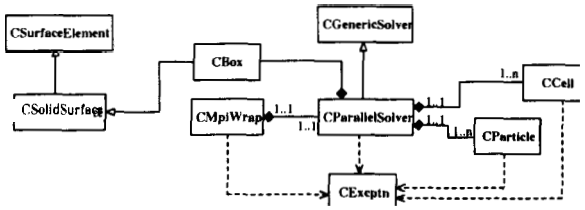



**Fig. 2 A Unified Modeling Language diagram of the parallel DSMC solver.**

## Object-Oriented Approach

An Object-Oriented (OO) approach is used in the development of the 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 which have a defined set of attributes and functions. On the contrary, a conventional CFD scheme deals with relatively abstract quantities such as density and pressure that are not physical objects and hence such schemes are difficult to code in an object-oriented language. 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 relationship are shown by a Unified Modeling Language (UML) diagram in Fig. 2. 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 simp ] provides the declaration of the required actions (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 points 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 the molecules from the body. A solid body is stored as a collection of surface triangles. A surface element class is written for the triangles that performs the collision and the bouncing of a particle with a triangle. The elements were chosen to be triangles because the solver is required to deal with arbitrary geometrical shapes and an unstructured grid of triangles easily achieves that purpose. 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 particles 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 is actually performed is 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 $doc++$[19] comments.

## Parallelization

There are two main reasons why the parallelization of the DSMC solver is desired: (1) The size of the problem may easily become larger than the available memory (Random Access Memory (RAM)) 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% speed up.[20] 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.

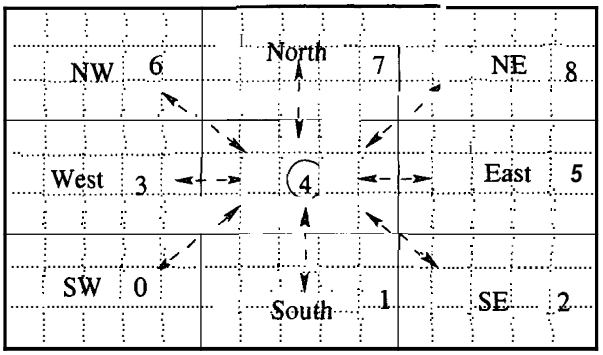



**Fig. 3** A schematic showing the communication neighbors of processor 4.

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 molecules) and this processor always works on the molecules in these cells. The communication is required at the end of each time step when the molecules are exchanged between neighboring processors. This approach may be combined with the embarrassingly parallel approach to perform simultaneous ensemble averages.

Figure 3 presents a 2-D domain decomposition among 9 processors. The processors 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. 3 for processor number 4 by dashed lines with arrows. The direction of the arrows indicates 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 per-

formed 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 double variables in all). The structure used for communication has seven double variables 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.

## Model for Diatomic Gases

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 in the kinetic theory of gases. 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" 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.

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

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. 1, 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

$$P/P_{\max} = 4(e_t/e)(1 - e_t/e) \qquad (2)$$

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

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

Now having obtained the post-collision kinetic energy, $e_t' = \mathcal{R}_2 e$ and the internal energy $e_i' = e - e_t'$ (conservation of energy), they are 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 just 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.

## Code Validation and Preliminary Results

The serial version of the parallel DSMC solver developed by the authors was validated against the Riemann shocktube problem.' Another test case is designed to validate the solid boundary condition implementation in the solver. The implementation of the solid boundary condition is described in detail in Ref.'

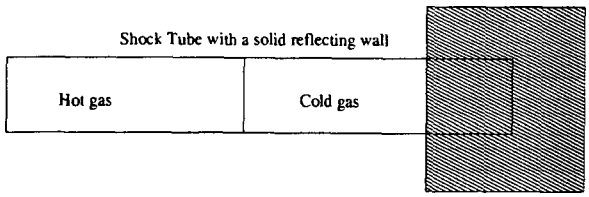



Fig. 4 A schematic of the setup of the numerical experiment to test the solid boundary condition implementation.

Preliminary results from parallel simulations of two experiments are presented. The first experiment is on the interaction of a planar shock wave with a square cavity. The interest is in the pressure loading on the walls of the cavity. The second experiment has been recently performed by Settles *et al.*[3] They blast an oxygen-acetylene gas mixture in a balloon inside a box and capture the shock wave interaction with the walls using Schlieren photography. It is extremely costly to simulate this experiment because a lot of ensemble averages are required to distinguish the shock wave from the statistical scatter.

### Solid Boundary Condition

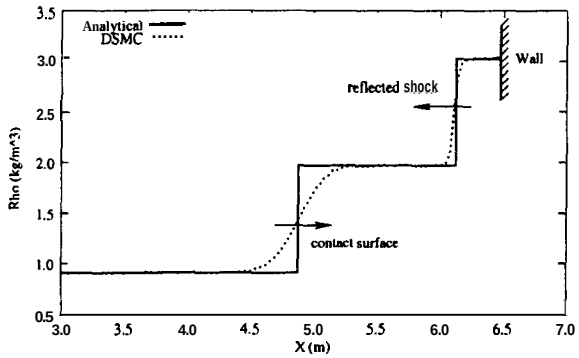
The solid boundary condition implementation is validated against the analytical solution for a normal shock wave reflecting from a solid wall. A solid box is placed at the end of the shock tube such that one face of the box is inside and blocks the entire cross-section of the shocktube (see Fig. 4). 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.

The comparison of the results against the analytical solution is presented in Fig. 5. The results match very closely with the analytical solution.

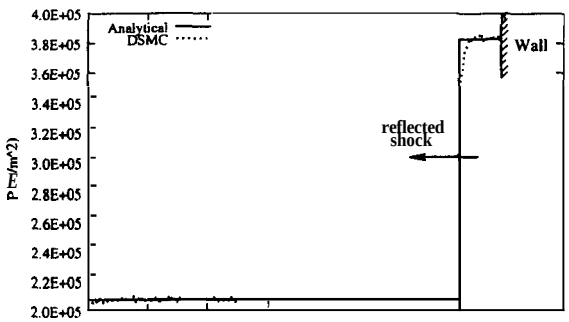
### Balloon Blast

Settles *et al.*[3] conducted an experiment on detonation of an oxygen-acetylene mixture in a balloon kept inside a box. They are interested in studying indoor explosions to safeguard aircraft against on-board explosions.[22] The dimensions of the box were 1.78 m $\times$1.78 m $\times$0.46 m, and the radius of the balloon was 7.5 cm. The mixture in the balloon was ignited by a nichrome-wire glow plug. The speed of the shock wave just after explosion was observed to be of Mach number 1.2.

The initial conditions for the simulation require specification of temperature and density inside and outside the balloon immediately after the explosion. It is assumed that all of the mixture ignites instantaneously and the temperature inside the balloon is uniformly escalated at the instant of explosion. The temperature inside the balloon is then calculated using normal shock relations across a planar shock wave of Mach 1.2 and assuming that the density inside the balloon is equal to the atmospheric density. The simulation is performed using the actual-size of the box.



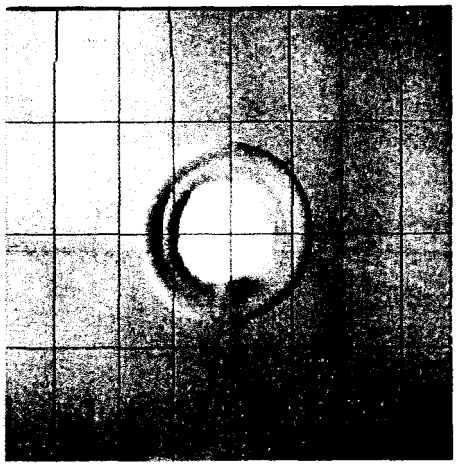

a) Density



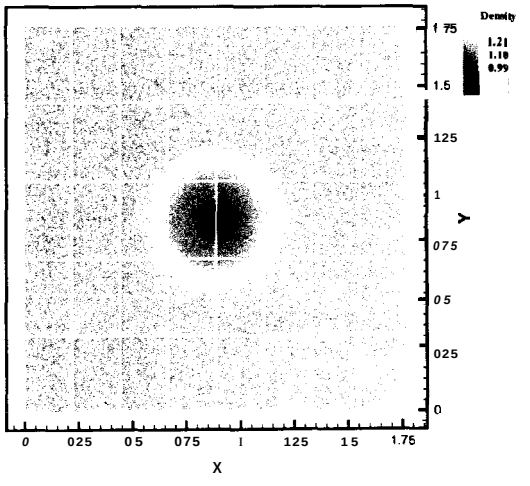

b) Pressure

Fig. 5 Comparison of the DSMC results against the analytical solution of the reflection of a normal shock wave from a solid wall.

Since the shock wave (Mach 1.2) is very weak, and it further weakens as it expands in the box, a number of ensemble averages are required to delineate the shock wave from the statistical fluctuations. Also, a very fine grid of cells is required to compare against sharp Schlieren images of the shock waves. This makes the problem very computationally expensive. A grid of 200 x 200 x 52 cells is used. Each cell is initially assigned 40 particles, which makes a total of 83 million particles. The simulation is performed on 40 2.8 GHz Athlon (AMD) processors of Mufasa,[23] so each processor has approximately 2 millions particles. The results are obtained by averaging over 90 ensembles. The simulations are performed for a total time of 1.6 milliseconds. The shock waves do not reach the walls in this time but the simulation is kept short due to limited resources. Figures 6 compares the results against the experiments by Settles *et al.*[3] Figure 7 shows the density and pressure contours on a X-Z plane at the center of the box.



a) Experiment




b) DSMC

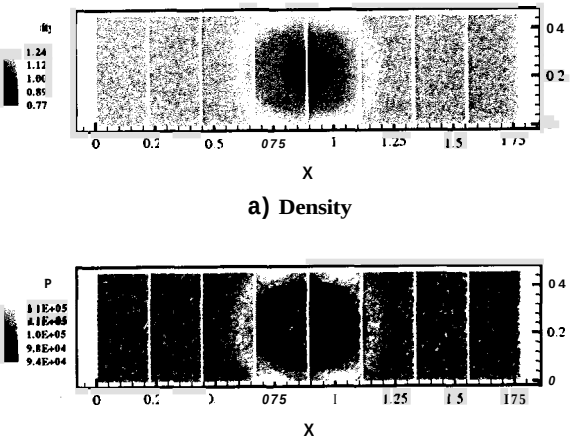
**Fig. 6 A comparison of the experimental and numerical results on an *X-Y* plane at the center of the box at t=0.54 ms.**

### Planar Shock Interaction with a Square Cavity

As a final test, a simulation of a planar shock wave traveling over a cavity is performed. This was experimentally investigated by Reichenbach *et al.*[2] A few snapshots from a preliminary run are compared against the experimental and simulation results of Reichenbach *et al.*[2] The DSMC simulation is a little different from the experimental case as there is a wall on the left instead of an inflow boundary. Figures 8 and **9** compare the DSMC results with the results from Ref.[2] A very encouraging match can be observed.

### Parallel Performance

The Riemann, balloon blast, and the 'shock interaction with a cavity' problems were solved for sample sizes to measure the parallel performance of the solver. 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 10 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 up to the maximum number of processors used for the solution. **A** better performance is expected for larger problem sizes.




a) Density




b) Pressure

**Fig. 7 The pressure and density plots on a X-Z plane at the center of the box at t=0.54 ms.**

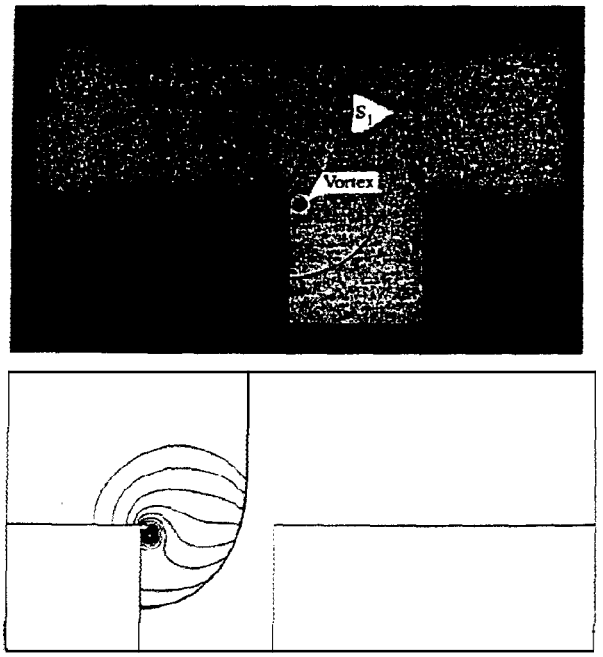
**Table 1 Listing of the three sample problems solved on COCOA-$2^{24}$ for performance testing.**

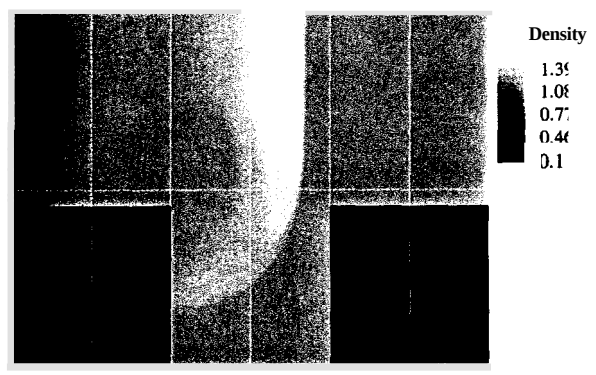
| Case | Timing(s) | | | | | |
|---|---|---|---|---|---|---|
| | *Procs* | *Move* | *Collide* | *Sample* | *Comm.* | *Total* |
| **Riemann Shocktube** Cells: 500*400*1 Part: 6 mill. | 1 | 2462.8 | 20181.2 | 31.1 | 0.35 | 27644.7 |
| | 2 | 1327.2 | 10019.8 | 20.37 | 937.5 | 14722.9 |
| | 4 | 702 | 5010.74 | 9.96 | 531.91 | 7562.35 |
| | 8 | 307.63 | 2499.12 | 4.1 | 428.87 | 4025.99 |
| | 16 | 153.39 | 1249.25 | 1.97 | 246.99 | 2226.88 |
| **Balloon Blast** Cells: 44*44*44 Part: 2.55 mill. | 1 | 1056.3 | 8734.77 | 14.81 | 0.01 | 11772.9 |
| | 2 | 532.79 | 4325.29 | 8.61 | 485.54 | 6460.18 |
| | 4 | 266.48 | 2152.67 | 4.06 | 288.46 | 3326.12 |
| | 8 | 126.05 | 1069.94 | 1.83 | 198.3 | 1775.32 |
| | 16 | 759.77 | 542.15 | 0.93 | 107.69 | 1672.4 |
| **Shock over a cavity** Cells: 500*400*1 Part: 6 mill. | 1 | 3568.3 | 13861.5 | 22.11 | 0.01 | 20445.1 |
| | 2 | 1765.3 | 6997.8 | 12.88 | 706.42 | 11125.6 |
| | 4 | 540.35 | 1992.65 | 4.38 | 4845.49 | 8040.67 |
| | 8 | 61.59 | 243.28 | 0.92 | 3723.5 | 4333.99 |
| | 16 | 0.55 | 2.44 | 0.36 | 2048.9 | 2326.07 |

The results obtained for the benchmark problems and for other test cases prove that the solver is capable of accurately solving complex blast-interaction problems. The parallel program is scalable and can efficiently handle large problem sizes.

## Conclusions

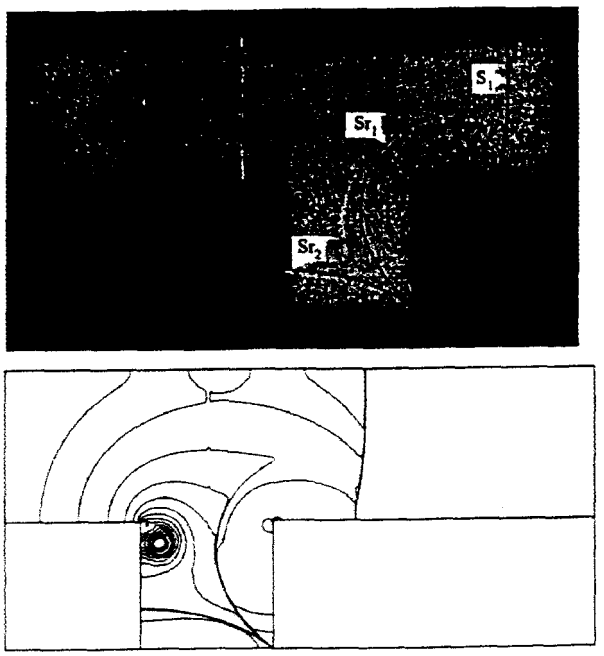
A parallel, object oriented DSMC solver has been developed to numerically solve the problem of blast-impact on arbitrary-shaped structures. The solver comprises of a very natural choice of classes that



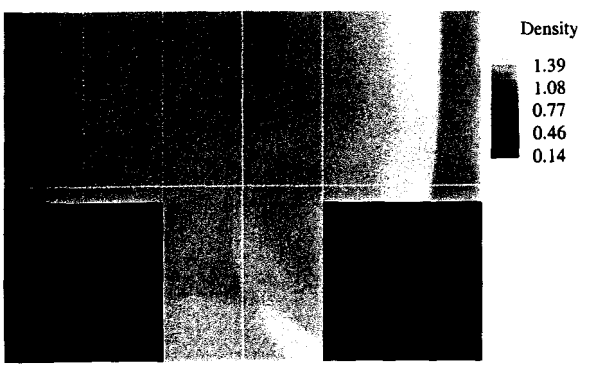




b) DSMC results

**Fig. 8 The interaction of a planar shock wave with a square cavity at an instant.**




a) Results from Ref.[2]




b) DSMC results

**Fig. 9 The interaction of a planar shock wave with a square cavity at another instant.**

makes the code very organized. The parallelization is achieved by domain decomposition and the communication between the neighbors is handled by a few MPI library calls.

The solid boundary condition (interaction of molecules with a solid body) implementation is validated against the analytical solution of a normal shock reflecting from a solid wall. A numerical simulation on the problem of a planar shock wave interaction with a square cavity is in progress. A similar study was experimentally performed by Igra *et al.*[2] The preliminary results from the simulation appear promising. It should be noted that for the blast-impact problem, the interest is in predicting the loading on the body, not capturing the aerodynamics. It takes a lot more time to precisely capture the aerodynamics than it does to obtain a reasonably accurate pressure loading.

## Acknowledgement

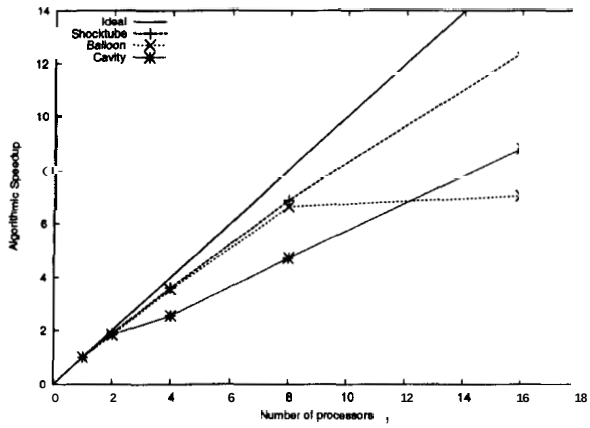
The authors would like to acknowledge the Protective Technology Center[25] at the Pennsylvania State University for sponsoring this research.

## References

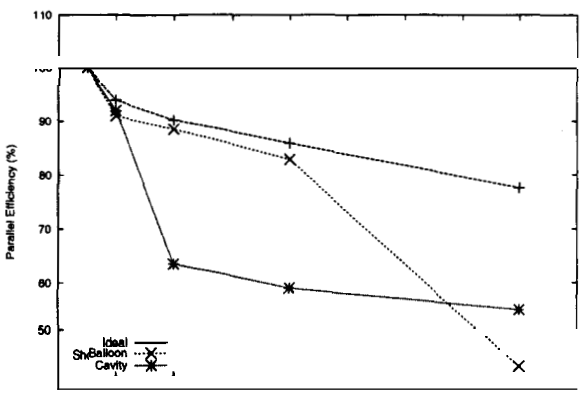
[1] A. Sharma, L. N. Long, and T. Krauthammer, "Using the Direct Simulation Monte Carlo Approach for the Blast-Impact Problem," *The 17th International Symposium on Military Aspects of Blast Simulations*, Las Vegas, Nevada, 2002.

[2] O. Igra, J. Falcovitz, H. Reichenbach, and W. Heilig, "Experimental and Numerical Study of the Interaction Between a Planar Shock Wave and a Square Cavity," *Journal of Fluid Mechanics*, Vol. 313, 1996, pp. 105–130.

[3] G. S. Settles, J. R. Benwood, and J. A. Gatto, "Optical Shock Wave Imaging for Aviation Security," *Proceedings of 4th ASME Fluids Engineering Conf.*, Hawaii, July 6-11, July 2003.




**a)** Algorithmic Speedup




[14] J. B. Anderson and L. N. Long, "Direct Simulation of Pathological Detonations," 18th International Symp. on *Rarefied* Gas Dynamics, Vancouver, Canada, Jun. 2002.

[15] W. Wagner, "A Convergence Proof for Bird's Direct Simulation Monte Carlo Method for the Boltzmann Equation," Journal of Statistical Mechanics, Vol. 66, No. 3/4, 1992.

[16] D. I. Pullin, "Direct Simulation Methods for Ideal-Gas Flow," Journal of Computational Physics, Vol. 34, 1980, pp. 231–244.

[17] C. L. Merkle, H. William Behrens, and Robert D. Hughes, "Application of the Monte-Carlo Simulation Procedure in the Near Continuum Regime," Presented at the *12th* International Symp. on Rarefied Gas Dynamics, University of Virginia., Jul. 1980. Twelfth

[18] N. G. Hadjiconstantinou and A. L. Garcia, "Molecular Simulations *of* Sound Wave Propagation in Simple **Gases,"** Physics of Fluids, Vol. 13, 2001, pp. 1040-1046.

[19] "DOC++ Homepage:," `http://docpp.sourceforge.net.`

[20] L. N. Long and K. S. Brentner, "Self-Scheduling Parallel Methods for Multiple Serial Codes with Applications to WOP-WOP," AIAA Paper *2000-0346,* 38th Aerospace Sciences and Meeting Ezhibit, Reno, Nevada, Jan. 2000.

[21] Claus Borgnakke and Poul S. Larsen, "Statistical Collision Model for Monte Carlo Simulation of Polyatomic Gas Mixture," Journal of Computational Physics, Vol. 18, 1975, pp. 405–420.

[22] G. S. Settles, B. T. Keane, B. W. Anderson, and J. A. Gatto, "High-speed Imaging of Shock-Wave Motion in Aviation Security Research," FAA 3rd International *Symposium* on *Explosive* Detection and Aviation Security, Altlantic City, New Jersey, Nov. 2001.

[23] "Mufasa Homepage:," http://vvv.cse .psu.edu/mufasa/.

[24] "COst effective Computing Array-2 (COCOA2):," http://cocoa2.ihpca.psu.edu.

[25] "Protective Technology Center, The Pennsylvania State University," http://vvv.ptc.psu.edu.