Optimizing the Direct Simulation Monte Carlo Algorithm for Multi-Core Processors

Table of Contents

1. Executive Summary

The goal of this project is to create a physically realistic Direct Simulation Monte Carlo (DSMC) model and optimize its performance on multi-core processors.

Objects moving through low-density atmospheres, such as a spacecraft passing through the upper atmosphere, are difficult to experiment with directly, making accurate simulations of these objects especially important. DSMC is an appropriate model for such simulations because it models low-density airflow efficiently and has the capability to simulate large regions. Despite this efficiency, DSMC requires large amounts of compute power in order to ensure the accuracy of any substantially sized simulation.

An original DSMC program, written in C++, has been tested for accuracy through a series of sample problems and then used to simulate objects moving through the upper atmosphere. Special attention has been given to the shockwaves formed by objects, such as a space shuttle, moving at supersonic speeds.

Although multi-core processors are a relatively new architecture, they are powerful enough to compute reasonably sized DSMC simulations. Since their architecture is different from supercomputers, the traditional platform for DSMC, executing this simulation on a multi-core processor required research and experimentation. In this project, the DSMC algorithm has been optimized for efficiency on multi-core processors.

2. Overview of Direct Simulation Monte Carlo Methods

Direct Simulation Monte Carlo, commonly referred to as DSMC, is a 3-D simulation method in which the simulation region is broken up into many subsections, known as cells. Each cell contains particles that have both velocity and position vectors that define their movement. However, the interactions between particles are not dependent on the position vector, which is used purely to define the particle's trajectory. Particle-particle interactions may only occur if two particles are within the same cell. When simulating a region, the probability for a collision is

applied to all particles in a cell. Within that cell, a certain number of collisions will be performed and new velocity vectors will be assigned to the particles that collide.

The Knudsen number is defined as the ratio between the mean free path, (average travel distance between successive collisions), and the representative length of the simulated object. The Knudsen number controls the appropriate model and parameters for a model because it gives the relationship between the density of the region and the size of the simulated object. Choosing a smaller Knudsen number for the simulation results in a more accurate simulation, but the compute intensity involved in the simulation would increase drastically. On the other hand, a large Knudsen number would have greater inaccuracies. The Knudsen number 0.1 is commonly used because it gives a balance between accuracy and compute intensity (Gallis, 2008).

When compared to other standard modeling methods such as fluid and molecular dynamics, DSMC has several advantages in areas involving a large simulation region containing low densities. The area modeled by DSMC is usually rarefied gas, such as the gas found in the upper atmosphere. The collision by two gas molecules does not always affect other molecules because the average distance between molecules is so large, meaning the gas does not act completely like a fluid. This leads to an advantage for DSMC over fluid dynamic models in upper atmosphere. DSMC also allows for a much larger area to be simulated. Since only the cell positioning of a particle affects collisions in DSMC, it is much more efficient to simulate large numbers of particles. This gives DSMC a significant advantage over the compute intensive molecular dynamics models for large regions containing many particles.

3. Physical Mechanisms of Simulation

3.1 Finding Probability of Collisions

We can estimate probability for a collision to happen between a pair of particles within a cell by comparing the volume available for collisions to the volume of the entire cell. First, we find the total distance covered by one particle in one time step, or $V_{max}\,\Delta t$. In this equation, V_{max} represents the maximum relative velocity a particle can have. V_{max} is found by testing the velocity of the sampled particles at different times, and multiplying it by three to ensure maximum velocity (Matthias, 1999).

In order for two particles to collide, the cross-sectional area for the first particle must intersect with the cross-section of the second particle.

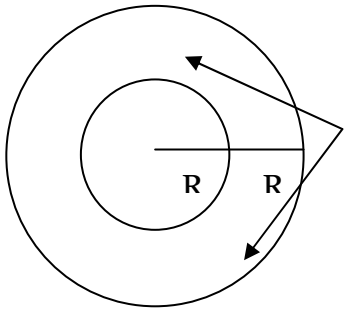


Figure 1

The center of the second particle must be within this disk. We can see that the possible volume for a collision to take place has a cross section with a radius of 2R. Since the maximum distance a particle can travel is $V_{max}\,\Delta t$, the maximum volume available for collisions is $V_{max}\,\Delta t(4\pi R^2)$.

If we proceed to divide the volume available for collisions by the total volume of the cell, we get an estimated probability of a collision between a pair of particles in a cell.

$$P(c) = \frac{V_{\max}\Delta t(4\pi R^2)}{V_c} \qquad (1)$$

We multiply this rough probability estimate by the number of pairs of particles in a particular cell to determine the number of potential collisions in that cell. Since there are Nc choose 2 (NcCr2) pairs of particles in a cell, the maximum number of collisions in a cell is

$$M_c = \frac{V_{\max} \Delta t (4\pi R^2)}{V_c} [\frac{N_c (N_c - 1)}{2}] \quad (2)$$

3.2 Choosing Colliding Particles

After we derive the maximum number of collisions in a cell within a time step, we must choose which collisions to perform since not every pair of particles will collide. The actual probability of a pair of particles colliding is dependent on the relative velocity of the two particles, given by $V_{rel} = |v_i - v_j|$. If the relative velocity between two particles *i* and *j* is very small, then the two velocity vectors are almost identical, resulting in smaller chances that these two particles will collide. On the other hand, two particles with a large relative velocity are more likely to collide. We can now introduce an acceptance-rejection method to randomly collide particles with a probability depending on their relative velocity (Matthias, 1999).

$$\frac{|v_i - v_j|}{v_{\max}} > Z \quad (3)$$

Two particles *i* and *j* will collide if this equation is satisfied. *Z* is a randomly distributed real number between 0 and 1. We roll a new *Z* for every new pair of particles. Z is independent of any other variables in the simulation.

3.3 Deriving Post-Collision Velocities

For two particles that have been chosen to collide, we can calculate their post-collision velocities from their pre-collision velocities. The calculation of the post-collision velocities is based on the fact that collisions will be elastic in the DSMC model due to no thermal diffusion. This allows us to use the following two physical properties:

I. Conservation of Total Momentum

$$m_1 v_1 + m_2 v_2 = m_1 v_1' + m_2 v_2' \quad (4)$$

II. Elastic Collisions (Conservation of Total Kinetic Energy)

$$\frac{1}{2}m_1{v_1}^2+\frac{1}{2}m_2{v_2}^2=\frac{1}{2}m_1{v_1}'^2+\frac{1}{2}m_2{v_2}'^2 \tag{5}$$

We rearrange and perform numerical operations on these two equations to find one resultant velocity:

$$v_1'=\frac{(m_1v_1+m_2v_2)}{(m_2+m_1)}+\frac{m_2}{(m_2+m_1)}(v_2-v_1) \tag{6}$$

Notice that this equation includes the velocity at center of mass for a two-body collision. Then, we assign random unit sphere direction vectors *e* to the two resultant velocities, netting two final equations for resultant velocities of two colliding particles.

$$\vec{v}_1'=\vec{v}_{cm}+\frac{m_2}{m_1+m_2}\left|\vec{v}_2-\vec{v}_1\right|\vec{e} \tag{7}$$

$$\vec{v}_2'=\vec{v}_{cm}-\frac{m_2}{m_1+m_2}\left|\vec{v}_2-\vec{v}_1\right|\vec{e} \tag{8}$$

3.4 Determining Particle-Object Collisions

When a particle collides with a macro-sized object, the equations governing the collision are different. In this case, the mass of the object can be considered to be infinite due to the insignificant mass of the particles. The conservation of momentum along the surface of the object requires the angle of incidence for the particle will be equal to the angle of reflection. The basis for the determination of the angle is the normal to the barrier. The speed of the particle does not change from incidence to reflection. However, the particle will take on a new direction.

3.5 Finding Thermal Velocity of Particles

All particles have a thermal velocity due to finite temperature of atmospheric gas. The thermal velocity for classical equilibrium is assigned according to the Maxwell-Boltzmann speed

distribution. The distribution will be dependent on several factors in the atmosphere, including the molar mass of the particles and the temperature of the system. The distribution is as follows (Nave):

$$f(v) = 4\pi\left(\frac{M}{2\pi RT}\right)^{\frac{3}{2}} v^2 e^{\frac{-Mv^2}{2RT}} \quad (9)$$

In this equation, f(v) is the distribution of the velocities (v). M represents the molar mass of the simulators (kg/mol) and T defines the temperature of the gas (K). Finally, R is the gas constant (8.3145 J/mol K).

In the simulation, we initially assign speeds of particles based on the equilibrium Boltzmann distribution but with random directions to speed up convergence.

3.6 Determining Mean-Free Path and Cell Size

In our simulation, we assign our cell size according to the mean free path. It is commonly accepted that the size of the cells should be smaller than the mean free path in order to gain accuracy when counting collisions. The mean free path is equivalent to the ratio between average velocity and collision rate of particles (Matthias, 1999). The collision rate can be found by multiplying the average number of particles per unit volume (n_v) by the amount of space available for collisions.

$$\Gamma = n_v(v_{rel} 4\pi R^2) \quad (10)$$

We use the relative velocity to calculate the volume swept out in a time step because other particles are also moving in the cell. The relative velocity can be calculated to be the square root of 2 times the average molecular velocity (Nave). Now, we can define the mean free path as the ratio of average velocity to the collision rate:

$$\lambda(M.F.P.) = \frac{\bar{v}}{\sqrt{2}\pi 4R^2\bar{v}(n_v)} = \frac{1}{4\sqrt{2}\pi R^2(n_v)} \quad (11)$$

From this equation, we can see that the mean free path of a particle is dependent on temperature, pressure, and the size of the particles since those factors will affect n_v. The cell size

can be chosen to best accommodate the application. For cell sizes, there is a trade-off between the accuracy and the compute time required. However, the cell size should always be less than the mean free path. In our simulation, the cell size is approximately one-third of the mean free path. From the mean-free path, we can then calculate average collision time T, given by the mean free path divided by the average molecular velocity. The step Δt should be much smaller than the average collision time. The optimal ratio between time step and collision time is one tenth, which we use in our simulations (Gallis, 2008).

4. Program Description

The DSMC program for this project is original code written in C++. The program is designed to run on a multi-core platform and uses dynamic array allocation to accommodate large-scale simulations.

Simulator data is stored in a primary data array by cell number. All cells begin the simulation with room to hold zero simulators and continuously allocate more room to accommodate increasing numbers of simulators. When running the program on multiple cores, each processor is assigned a range of cells to calculate. Simulators that move between cells are put on a list of scheduled particle movements that are executed at the end of each time step.

This memory grouping method stops cells that continuously have low density from wasting memory, making the code memory more efficient. The computer will also spend less time accessing memory in this configuration because many particles that must be calculated in order will be in the cache line if they are physically grouped together, greatly decreasing computational time. Each simulator requires 60 bytes, allowing thousands of particles to be stored in the cache at one time without accessing main memory. If simulators were not stored in

order of their cell, processors would have to fetch particle data for collisions from a list that was completely unordered, requiring a main memory access almost every time.

Simulators are initialized with velocity dependent on the thermal energy of a gas at a specific altitude and temperature as dependent on the Maxwell-Boltzmann speed distribution. These particles are then used to set up the primary data arrays.

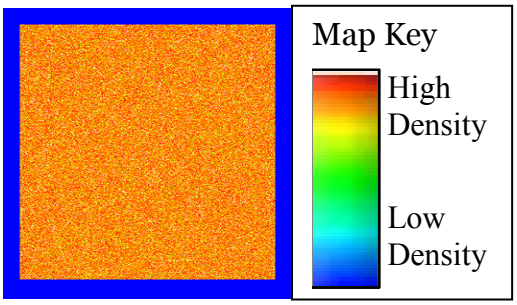
There are two primary functions for calculating the movement and collisions of simulators. One calculates collisions between molecules using the methods described. The other calculates collisions between the simulators and the boundaries. Each boundary is defined as a collection of line segments in the simulation region. By converting each simulator's movement into a line segment, simple algebra may be used to see if a simulator collides with a boundary and then reflect the particle off the boundary. On each timestep, the number of simulators in each cell is added to an array that is used to create a gif image that shows the density and velocity map over the simulation region. At the end of the simulation some information parameters are displayed, such as the total number of particle collisions, the number of simulator-boundary collisions, and a histogram of the size of each cell's memory.

5. Calibrating the Simulation

In order to ensure that the simulation is physically realistic, it is necessary to validate the model through some form of test. The goal is a realistic representation of gas behavior at some atmospheric altitude and therefore must be calibrated for the temperature, velocity, pressure, and density at that altitude. The calculation of gas movement in the simulation must also be realistic to gas behavior and must not gain or lose energy through collisions between simulators or with objects in the simulation region. Accurately assembling a DSMC program is further complicated by the calibration of the Knudsen number, cell size, and simulation region.

The "Box Test" verified these aspects of our model because this test requires correct molecule-molecule and molecule-object collisions, a balance between particles and cell size, and an efficient and accurate data structure. In the box test, the gas is put in a bounded area, initialized with the Boltzmann distribution. The simulators only perform elastic collisions with each other and the bounds of the region. There should be no change in total energy of the system, making this a very good test of the physical accuracy of the collision equations. Figure 2 shows the density map and parameters for this simulation.

Figure 2



Parameters:
Altitude: 90 km
Temperature: 180 K
Pressure is 0.0019 mmHg
Mean free path: 0.0245 m
Simulation region: $5m^2$ with 500 cells on a side (about .4 the mean free path)
Radius of average air particle: 3×10^{-10}m
Mass: 4.81727×10^{-26}kg
Number of simulators: 255,000, each representing 600,000,000 particles
Simulated time: .001 seconds
Runtime: 2 minutes 30 seconds to run on the dual core laptop
Memory Requirement: 323,800 KB

According to calculated values, there should be 14800 collisions per second for each molecule at this altitude, which corresponds to one collision every .000068 seconds (Nave). Since we want only a tenth of the particles to collide each time step (Gallis), the time step is .0000068 seconds, or one tenth of the theoretical collision time. There were 25,100 collisions per time step, which meets the $1/10^{th}$ goal.

The total number of collisions was also validated. The theoretical number of total collisions is 3,766,733. There were 3,757,507 collisions, a mere 9,000 off the theoretical value with a .2% error.

Data about air mass and radius as well as their relation to the mean free path is from R. Nave's "Kinetic Theory Concepts."

The Box Test is also a good test of the efficiency of the data structures and the multi-core operation of the code. If a simulator moves between cells calculated on different processors it takes extra time. In many simulations, the simulators will tend to stay in some region of the area,

which reduces the time cost of re-ordering particles. In this test, particles will move throughout the box, causing them to change processors, and testing the efficiency of memory movement.

This test was also used to calibrate the parameters of the program to an actual altitude by taking into account the pressure, temperature, and mean free path at 90km. This also helped integrate realistic values for mass and radius of the average air atom.

The simulation 'passed' the Box Test because the energy varies by one part in 10^{13}, which corresponds to machine accuracy. This means the collisions equations are not losing or gaining energy. In addition, this validated the number of collisions, both per timestep and over the duration of the simulation.

6. Applications

A calibrated DSMC model allows us to simulate complex rarified gas flows in the upper atmosphere with confidence that these simulations are accurate. The existing model allows us the opportunity to observe and analyze many familiar objects and gas phenomenon in the upper atmosphere. The simulation creates density and velocity vector maps of the simulation region as well as calculate the force exerted on each section of the object.

6.1 Shockwave of a Supersonic Box

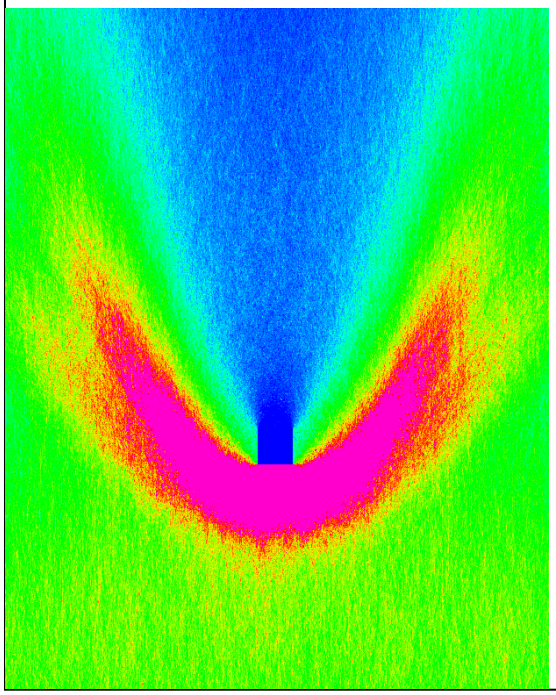
The program was used to calculate the shockwave of a box traveling at supersonic speeds through the atmosphere. Shockwaves are formed due to disturbances in a medium. Under supersonic conditions, the simulated object creating the disturbance travels faster than the particles that bounce off it. This results in a sudden change in temperature, pressure, and density, causing a shockwave. In rarefied gas, the shockwaves formed are generally thicker, ranging up to ten mean-free paths thick (Gallis, 2008). Figure 3 shows the simulation setup for a standard shockwave simulation.

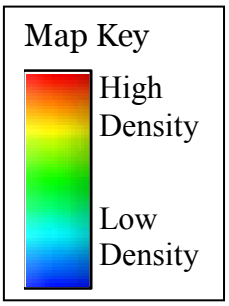
The shape of the shockwave is dependent on the speed of the propagating disturbance compared to the speed of sound, which will decrease with decreasing density. Any supersonic disturbance at a particular density will cause a "cone shape" in the particle density distribution. The angle that the cone forms with the normal to the object, or the semi-cone angle, is given by the equation ($\sin(\theta)=c/v$) where c is the speed of sound in the gas and v is the speed of the disturbance (Elert). At subsonic speeds, $\sin(\theta)$ is greater than one, resulting in no real solutions (i.e. no shockwave). With a larger v, the angle between the shockwave and the axis direction of the cone becomes increasingly small.

Figure 3

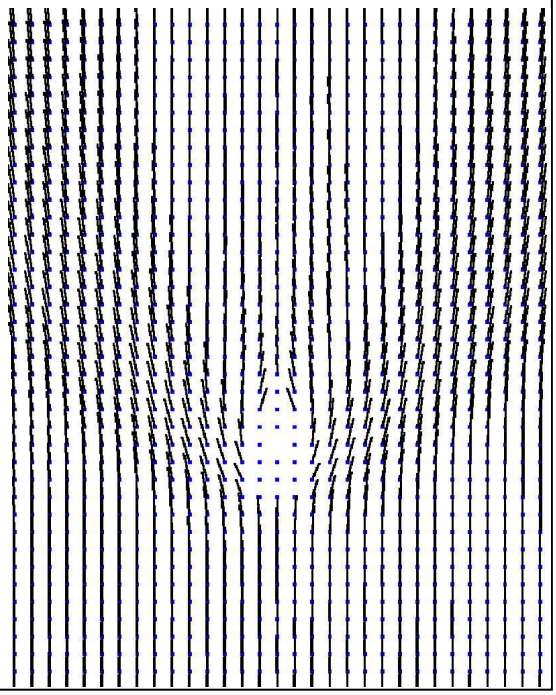
Parameters:

Runtime: 27 minutes on the quad-core system
Memory: 1.4 MB
Collisions per time step: 21,000
Total Simulation time: 0.04 s
Time step: .0000068 s
Simulation area: 10 m^2, 1000 cells on a side.
Simulators: 250,000, each representing 7,000,000,000 molecules
The actual size of the box was .4 meters and its speed was 1.2 times the speed of sound, or Mach .95.
Note that the speed of sound is 268 m/s at this altitude.





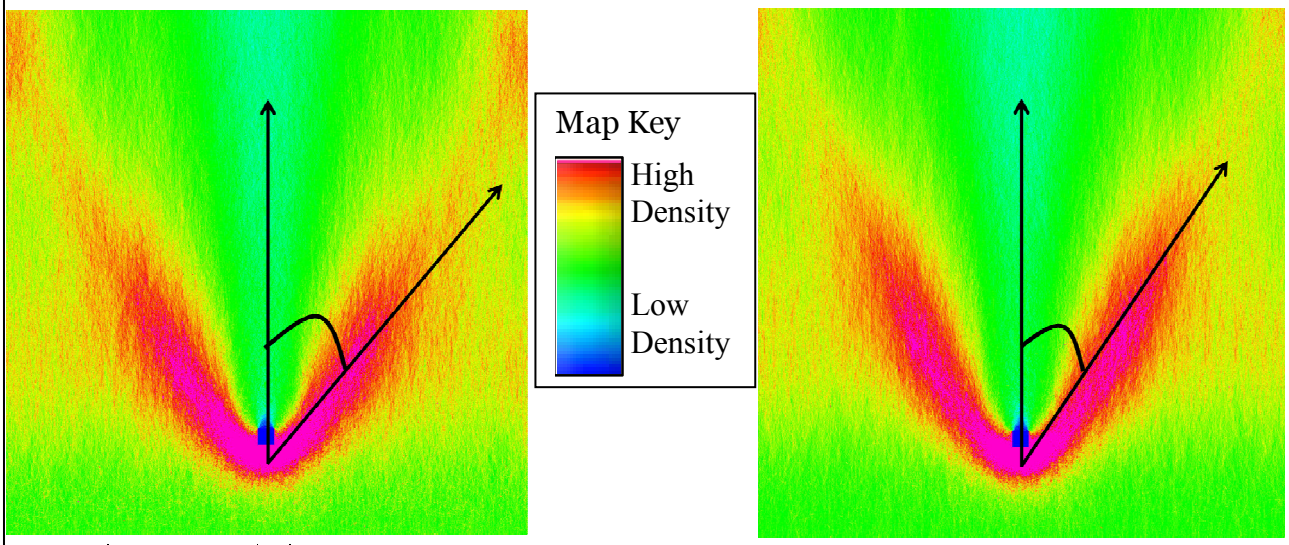
At right is a velocity map of the simulation region, which shows direction of flow. The blue dots signify the beginning of each vector.



We simulated the shockwave shape of a box at different mach numbers. As predicted, the shockwave is thicker than at lower altitudes and the angle increases with Mach number. In the following pictures, the average direction and speed of particles in each area is shown by the lines. The blue dots show the starting point for the line, and white lines are supersonic while black lines are not. This simulation generated the following density and velocity map:

We also ran several simulations to show how Mach number changes the form of the shockwave. We looked at shockwaves formed at many different Mach numbers and compared their density and velocity maps. Figure 4 is one example of such a simulation.

Figure 4
The deformation of a shockwave can be seen in these two pictures. The shape of the shockwave correctly matches predicted values.



6.2 Simulation of the Space Shuttle

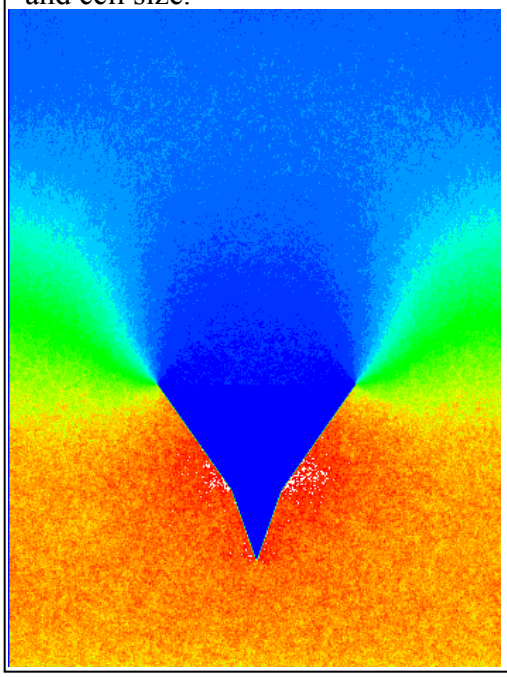
Using the same simulation region at 90 km, the space shuttle shape was simulated. This simulation was one of the most compute intensive, taking over 11 hours to run. Figure 5 shows

the density and velocity maps for the simulation.

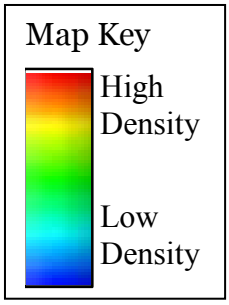
Figure 5

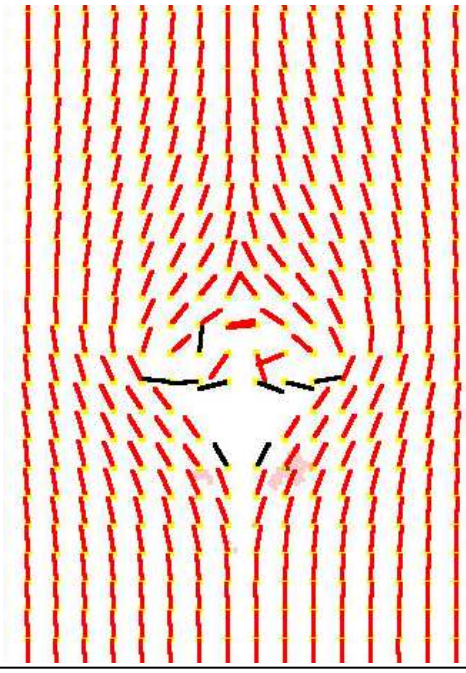
Parameters

The simulation had the same parameters as the box simulation regarding altitude, temperature, and cell size.



Speed of Object: Mach 1.58.
Simulated time: 2 s
Runtime: 11.5 hours in dynamic mode on the quad-core system.





7. *Performance Analysis*

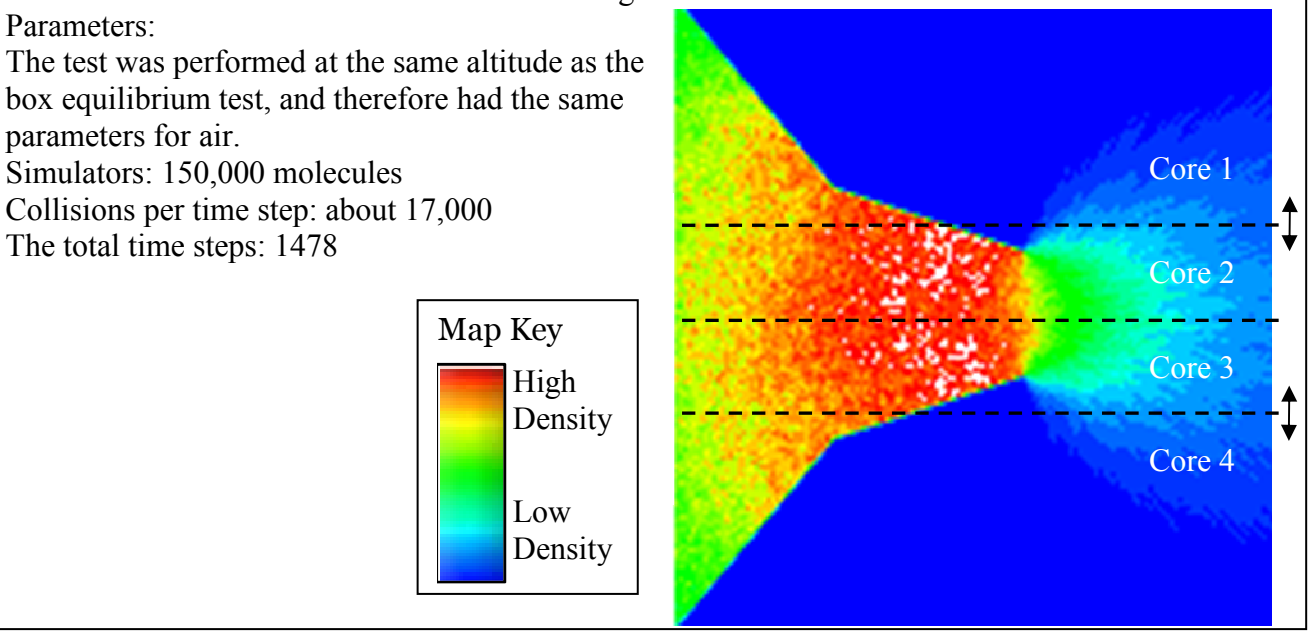
A standard compression nozzle simulation was used as a baseline for computer performance tests. Compression and expansion nozzle flows are interesting problems for DSMC because they involve both density variation and particle movement. Two computers were used in the performance test. One is a laptop with two 1.7 GHz cores and 256 KB L2 cache per core. The other is a quad-core desktop with 2.4 GHz processors and 2 MB cache per core.

When using the quad-core system, the program has the option of dynamically balancing the number of cells assigned to each processor. For two cores, the simulation region is divided in half so that both cores have the same number of cells. When using four cores, the simulation

region is not symmetric over all four, and so the intermediate boundary lines between cores 1, 2 and 3, 4 change dynamically, as shown in Figure 6.

Figure 6

Parameters:
The test was performed at the same altitude as the box equilibrium test, and therefore had the same parameters for air.
Simulators: 150,000 molecules
Collisions per time step: about 17,000
The total time steps: 1478



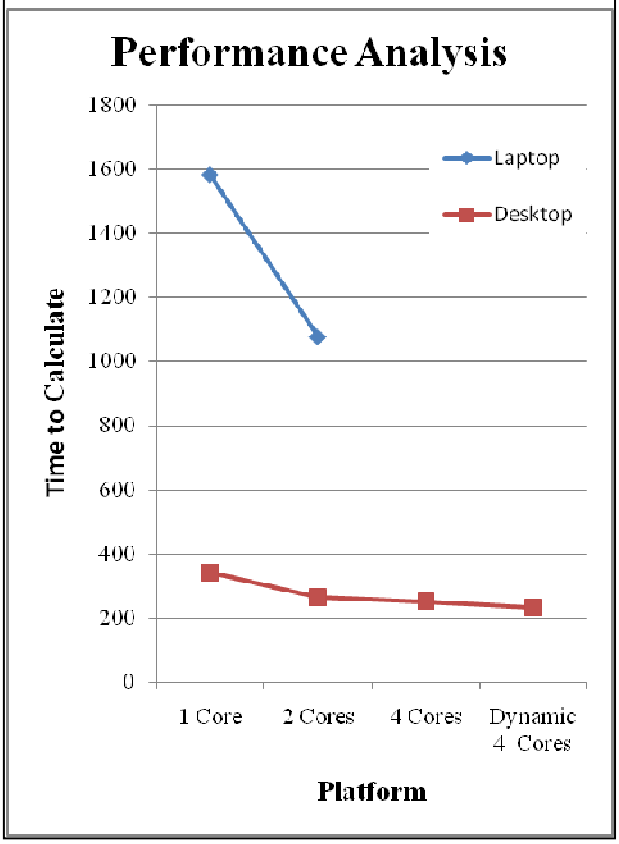
Each core computes how long it takes to calculate its section, and then the boundaries are moved so that cores with short calculation times have more cells on the next time step. Although the proportion of cells on each processor can be manually set at the start, dynamically changing this parameter throughout the simulation is more effective because the distribution of simulators may change over the duration of the simulation.

The nozzle simulation was run on different numbers of processors on each computer to analyze performance. Figure 7 shows the time it took for these computers to run the simulation. As shown, the laptop benefited most from running multi-core, while the times on the desktop did not improve as dramatically.

In fact, further analysis of the desktop's performance showed it was not achieving the expected performance benefits. As shown in Figure 9 (page 18), line "With Dynamic Balancing," running

the program on four cores only make it run twice as fast as it did on one core, not four times faster. Even though the benefit of adding a processor usually decreases for many processors, the benefit of four cores should still be higher.

Figure 7



A likely culprit for decreasing the benefits of many cores was wasting CPU time having only one core reallocate particle memory. The program's original memory configuration involved creating a list of particle movements for all particles that move between cells, and then after the calculations are completed for one timestep, all the particles scheduled for movement are moved by one core. Since this method involved wasting CPU time having one core operating alone, this method did not produce satisfactory speedup. A second method was devised to correct for this CPU wastage by having molecules that move within the region of a processor be moved by that processor. Only the molecules that jumped the boundary of calculation by one processor were scheduled for movement in a serial phase. This new method succeeded in utilizing all the CPUs; average CPU usage jumped from around 70-80% to a solid 90%. The Desktop's performance with the new memory configuration is shown in Figure 9, line "Reduction of Serial Portion."

7.1 Memory Access Bottleneck on Quad-Cores

Why is the desktop's performance not higher even with efficient CPU usage? To answer

this question, we ran several tests on memory access time. Using high-precision timers, we found the time it took for one core to execute a molecule-molecule collision, which involves obtaining the memory for each molecule. The results are shown in Figure 8. As shown in the table, four cores took longer to run the same calculation in parallel because they shared one memory bus.

Figure 8

The analysis of the time it took for different numbers of cores to run the same calculation reveals bottleneck in memory access.

# of Cores	Time (µs)	% of base time
1	2.114	100%
2	2.397	113%
4	3.532	167%

Quad-cores with only one memory bus will experience a memory bottleneck.

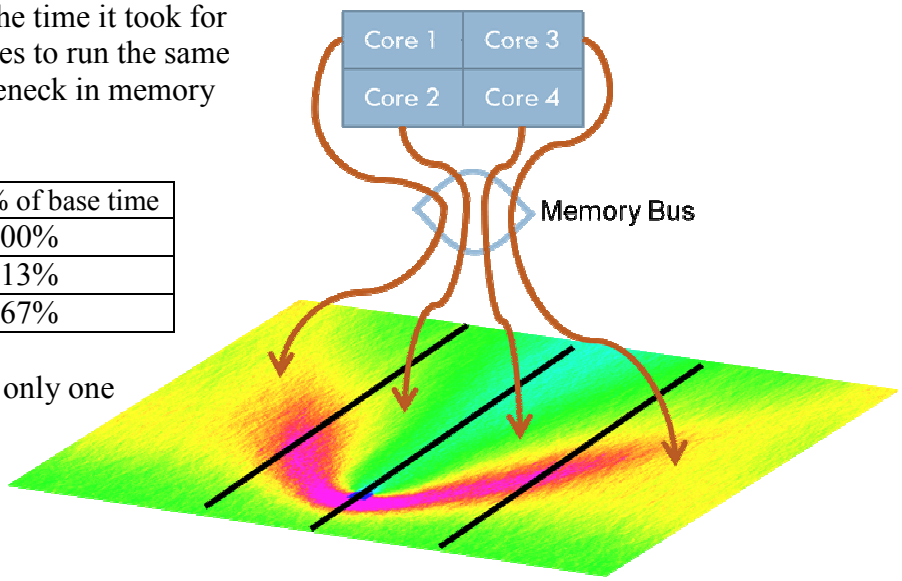
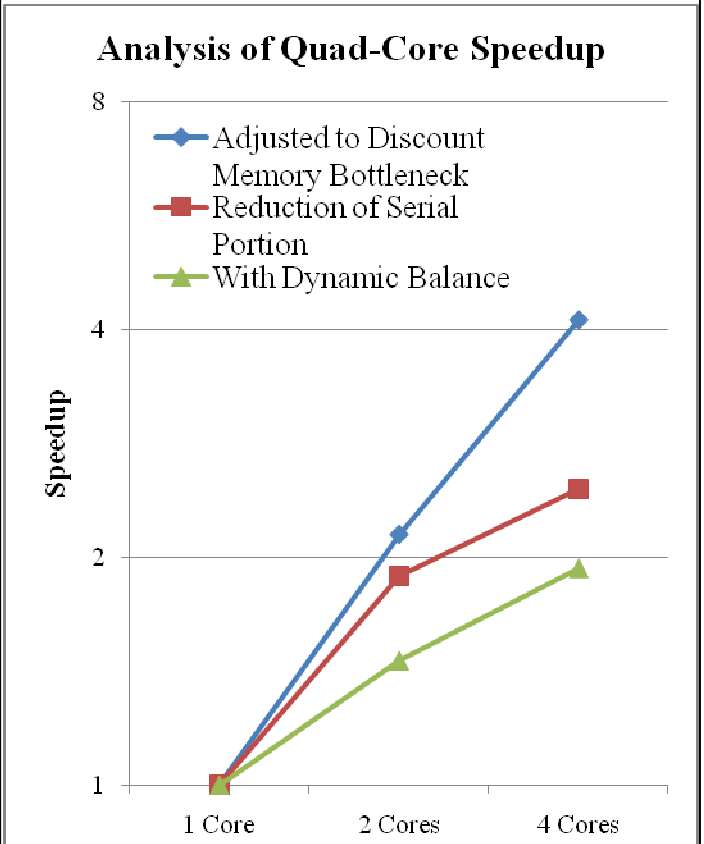


Figure 9 shows the Quad-core's speedup with different configurations of memory movement. We used the molecule-molecule timer test as a scale factor for how well the Desktop would perform if the memory bottleneck did not exist. Understanding the bottleneck problem on the Quad-core helps explain the shape of the "With Dynamic Balance" line. From one to two cores it has very good speedup, which tallies with the fact that the memory bus seems to be able to accommodate two cores. The fact that four cores did not seem to have a sufficiently significant effect on runtime is because four cores were greatly affected by the memory bottleneck, taking 167% of one core's time. The linear nature of the theoretical speedup suggests that additional modifications in the software would have a negligible effect on runtime because the slowdown originates in the hardware.

Even though the software is highly optimized for multiple cores, there are ways to increase performance. Adding particle chemistry, for example, would increase compute load but not memory bandwidth, relieving the bottleneck. Also, if the chip had more bandwidth per core, it would decrease the bottleneck. Either would increase the speedup multi-core provides without changing the software configuration significantly.

Figure 9



8. Conclusion

We have studied the physical basis of DSMC, developed and calibrated a robust program, simulated a wide variety of space objects, and observed the shockwave patterns they create. We have studied the algorithm's performance on multi-core, and optimized the efficiency of the software. From our work on this project, we have found that multi-core processors need high memory bandwidth to reach their potential performance.

9. Future Work

There are additional tests, simulations, and concepts we want to incorporate into this project. We would like to simulate other aerospace vehicles, the shockwaves they form, and the forces exerted on these objects. Executing this program on a supercomputer would help expand our understanding of the multi-core performance and help us further explore the computational design required for this application.

10. References

Bird, G. A. (2008, January). *Direct Simulation Monte Carlo Method Visual Programs at GAB Consulting*. Retrieved March 29, 2008, from http://www.gab.com.au/

Bird, G. A. (1994). *Molecular Gas Dynamics and the Direct Simulation of Gas Flows.* New York: Oxford University Press.

Bird, G. A. (2007). Sophisticated DSMC. *DSMC07.* Santa Fe.

Elert, G. (1998). *Shock Waves*. Retrieved April 1, 2008, from The Physics Hypertextbook: http://hypertextbook.com/physics/waves/shock/

Gallis, M., Personal Interview, (2008, March 25).

Haruhiko, O. (2002, January 26). *Random Number Generator*. Retrieved March 29, 2008, from http://oku.edu.mie-u.ac.jp/~okumura/cplusplus/mt19937.cc

Latta, L. *Building a Million Particle system.* My Game Developers Conference 2004.

Matthias, M. (1999, December 14). *Direct Simulation Monte Carlo*. Retrieved March 29, 2008, from http://www.hlrs.de/people/mueller/papers/honnef99/node4.html

Moss, J. N. (2007). Direct Simulation Monte Carlo Simulations of Ballute Aerothermodynamics Under Hypersonic Rarefied Conditions. *Journal of Spacecraft and Rockets*, 44.

Nave, R. (n.d.). *Kinetic Theory Concepts*. Retrieved March 29, 2009, from HyperPhysics: http://hyperphysics.phy-astr.gsu.edu/hbase/kinetic/ktcon.html#c1