Equations derived from kinetic theory often express a desired quantity in terms of a probability density. For example, the Direct Simulation Monte Carlo (DSMC) method is a well-known powerful technique for computational rarefied gas dynamics. It uses an algorithm that begins with an initial distribution and, through random sampling, converges to a stationary distribution. Random sampling is achieved using random numbers obtained with pseudo-random number generators. Quasi-Monte Carlo methods (QMCs) replace calls to a pseudo-random number generator by calls to a quasi-random number generator. QMCs are known to have a better convergence rates than Monte Carlo methods for multidimensional integration, but it is not trivial to make QMC work well in contexts outside of Monte Carlo integration, such as DSMC.

In fact, naive replacement of calls to a pseudo-random number generator by calls to a quasi-random generator have been known to fail utterly. We illustrate these difficulties and discuss how to overcome them. In the context of DSMC, we conclude that little can be gained through the use of quasi-random sequences, however in the context of "direct methods" we find promising results.

Introduction

We report on a series of experiments with algorithms used for approximating solutions to the Boltzmann equation.\textsuperscript{1–4} We begin with a review of the the Discrete Simulation Monte Carlo (DSMC) method of Bird\textsuperscript{2} as given by Garcia.\textsuperscript{5} We then give a brief introduction to quasi-random number generators (QNRGs) as opposed to pseudo-random number generators (PRNGs). Next we describe some experiments in which the PRNGs in Bird’s programs\textsuperscript{6} were replaced by quasi-random number generators QNRGs and discuss the results. Finally, we consider the effect of QNRGs on the approximation of the Boltzmann collision integral needed for the “direct method”.\textsuperscript{4,7–10} A final Section summarizes our conclusions.

DSMC History

Bird\textsuperscript{2} applied Monte Carlo methods to gas dynamics in the mid 1960s in an algorithm he named Direct Simulation Monte Carlo (DSMC). This algorithm was based on physical reasoning analogous to the derivation of the Boltzmann equation as explained in Section 10.2 of the book just cited. The main idea of DSMC is to construct a stochastic process for a many-particle system that evolves in time. The process begins with an initial particle distribution \( t = 0 \) and for \( t > 0 \) models gas dynamics realistically in the way the particle distribution evolves. Wagner\textsuperscript{11} proved that Bird’s method converges to a solution of the Boltzmann equation in a suitable limit.

Alternatively, Nanbu\textsuperscript{12,13} derived a simulation procedure directly from the Boltzmann equation. A modification of this algorithm was given by Babovsky\textsuperscript{14} and Nanbu’s method was shown to converge\textsuperscript{15} to a solution of the Boltzmann equation as the number of particles is allowed to grow indefinitely (and the solution is sufficiently smooth).

The Kac \( N \)-particle equation\textsuperscript{16} was used as the basis for another class of algorithms by Ivanov et al.\textsuperscript{17,18} in the spatially uniform case. The authors modeled the Leontovich equation in the spatially non-uniform case. For more discussion of the relationship between the equations just mentioned and the Boltzmann equation, see the references just cited along with Nanbu’s 1983 paper.\textsuperscript{19}
DSMC

All the methods mentioned above have been implemented using PRNGs and several authors have studied the effects of replacing the PRNGs in various schemes for approximating solutions to Boltzmann’s equation in various special cases. We are interested in investigating the effect of adapting Bird’s algorithm to the quasi-random setting. Here we follow Garcia’s description of DSMC.

The region to be considered is partitioned into cells. Initially \( N \) particles are distributed in a uniformly random way throughout the system with a predetermined number of particles in each cell. Each particle in the simulation represents a number \( N_a \) of actual particles in the physical system. In addition to its initial position \( r_i \), each particle is also given an initial velocity \( v_i \) according to the time \( t = 0 \) (initial) distribution. Particle evolution is carried out in discrete time steps \( \Delta t \) in two phases, advection and collision.

In the advection phase, particles move without interaction so that their positions are updated using the formula \( r_i \rightarrow r_i + \Delta t v_i \). If a particle reaches a boundary it is processed according to appropriate boundary conditions. After all particles have moved, a given number are randomly selected for the collision stage. Only particles within the same cell are allowed to interact and a set of collisions is processed at each time step. All pairs of particles are candidates for collision regardless of their positions within the cell. A collision probability can be derived depending on the model. Instead, an accept/reject procedure is implemented. If the pair is selected, the new positions and velocities are calculated as explained below. If the pair is rejected, the algorithm repeats the second step above. The routine continues until the required number of candidate pairs has been processed. The post-collision velocities are computed using conservation of linear momentum and energy. This gives an incomplete set of equations which is closed by choosing two uniformly random angles distributed over a unit sphere. The required number of candidates is determined using the ratio of the total of accepted candidates to the total number of candidates.

Note that particle collisions are determined by a sample drawn from a discrete probability space (the space of all pairs of particles in a given cell) while the random angles over the sphere are drawn from a continuous uniform distribution.

Quasi–Monte Carlo Methods

We begin with a brief description of a classic quasi-random number generator and then give two examples of the effect of adapting standard Monte Carlo methods to use quasi-random number generators.

Quasi–Random Number Generators

In 1935 van der Corput constructed a sequence of very uniform numbers in \([0, 1]\). For a non-negative integer \( n \) and digits \( d_i \in \{0, 1\} \), write

\[
n = d_0 + d_1 2 + d_2 2^2 + \ldots + d_k 2^k. \tag{1}
\]

Let

\[
\Phi_2(n) = \frac{d_0}{2} + \frac{d_1}{2^2} + \ldots + \frac{d_k}{2^{k+1}}. \tag{2}
\]

This can be done for any “base” \( b \): For digits \( \delta_i(n) \in \{0, \ldots, b - 1\} \), write

\[
n = \sum_{i=0}^{k} \delta_i(n) b^i \tag{3}
\]

and let

\[
\Phi_b(n) = \sum_{i=0}^{k} \delta_i(n) b^{-i-1}. \tag{4}
\]

These sequences are called quasi-random or low discrepancy sequences. There are many types of quasi-random sequences. For now, the ones described above are the only kind we need consider. Quasi-random sequences are generally much more uniform than pseudo-random sequences. This property is made quite precise in Niederreiter’s book. For example, using the built in function \( \text{rand}() \) from the gcc compiler, Figure(1) compares a pseudo-random sequence with a van der Corput sequence base three using histograms. In two dimensions, a pseudo-random sample of 10,000 points in the unit square is compared with a quasi-random sample of 10,000 points in the same domain using the Halton sequence in Figure(2).
a) Histogram of a sample of 5,000 pseudo-random points.

b) Histogram of a sample of 5,000 base 3 van der Corput points.

Figure 1 Comparison of standard MC sampling and quasi-random MC when sampling from a uniform distribution.

a) Pseudo-random points tend to clump (not optimal).

b) Quasi-random LDS (Halton Sequence) produces a much more uniform distribution of points.

Figure 2 Comparison of pseudo-random and quasi-random sequences in the unit sphere. The Halton sequence (b) generates a much more uniform distribution.

(Quasi-)Monte Carlo Integration

Recall\(^{50}\) that the Monte Carlo integration of a function

\[
f : [0, 1]^s = [0, 1] \times \ldots \times [0, 1] \to \mathbb{R}
\]

is an approximation of the form

\[
\int_{[0,1]^s} \approx \frac{1}{N} \sum_{i=1}^{N} f(\bar{x}_i)
\]

(6)

where \(\{\bar{x}_1,\ldots,\bar{x}_N\}\) is a sequence of random elements of the hypercube \([0, 1]^s\). The van der Corput sequence can be generalized to any dimension \(s^{29}\) by simply taking

\[
\Phi(n) = (\Phi_{p_1}(n),\ldots,\Phi_{p_s}(n))
\]

(7)

where \(p_1,\ldots,p_s\) are the first \(s\) prime numbers. There are many other ways to obtain quasi-random sequences in dimension \(s^{28}\).
Consider the integral
\[ I = \int_0^1 \int_0^1 \sin(xy) \, dx \, dy. \] (8)

One may verify that
\[ I = \int_0^1 \int_0^1 -\cos(xy) / y \, dy = -Ci(1) + \gamma \approx 0.2398117420005647259. \] (9)

Let \( \{ \bar{r}_1, \ldots, \bar{r}_N \} \), be a pseudo random sample of the unit square \([0,1] \times [0,1]\) and \( \{ \bar{x}_1, \ldots, \bar{x}_N \} \) be a quasi-random sample. Let
\[ I_{pr}(N) = \frac{1}{N} \sum_{i=1}^{N} f(\bar{r}_i) \] (10)
and
\[ I_{qr}(N) = \frac{1}{N} \sum_{i=1}^{N} f(\bar{x}_i). \] (11)

We want to study the sequences of errors
\[ I_{pr}(10) - I, I_{pr}(20) - I, \ldots, I_{pr}(5000) - I, \] and \( I_{qr}(10) - I, I_{qr}(20) - I, \ldots, I_{qr}(5000) - I. \)

A good way to visualize the result is to graph the points
\[ \{(10, I_{pr}(10) - I), \ldots, (5000, I_{pr}(5000) - I)\} \] and \( \{(10, I_{qr}(10) - I), \ldots, (5000, I_{qr}(5000) - I)\}, \)
interpolating between the points in each case. Figure (3) shows the result.

The reader should note that since the dimension \( s = 2 \) is small in this case, a fine mesh might be used along with a good quadrature to get an approximation better than the one obtained using pseudo-random sampling; however, the same sort of comparison in the error between Monte Carlo integration and Quasi-Monte Carlo integration persists to higher and higher dimensions because of the following well known inequality for reasonably behaved functions:
\[ \left| \frac{1}{N} \sum_{i=1}^{N} f(\bar{x}_i) - \int f(\bar{x}) \, d\bar{x} \right| \leq V \cdot D_N \] (12)
where \( V \) is a constant depending only on \( f \) and \( D_N \) is a quantity (called the discrepancy) depending on the quasi-random sequence \( \{ \bar{x}_1, \ldots, \bar{x}_N \} \). This inequality is called the Koksma-Hlawka inequality and full details may be found in the book. We note here that because of this inequality, lower discrepancy results in better approximations to the integral. One might expect that quasi-random numbers may be useful in speeding up or otherwise increasing the quality of other Monte Carlo methods, however, we will see that there are some subtleties in achieving such adaptations.
a) Standard Monte Carlo Method. The results shown here for the dissociation rate of nitrogen in modeling \( N_2 - N_2 \) collisions are assumed to be converged, using 5,000 sample points.

b) Quasi-Random Monte Carlo Method. The results shown here for the dissociation rate of nitrogen in modeling \( N_2 - N_2 \) collisions are of equal quality to the converged solution, using only 750 sample points.

Figure 4  The Quasi-Random Monte Carlo method yields results that are comparable with the best (converged) results using the standard method. Here, only 750 quasi random are used to obtain accuracy comparable to that obtained with 5,000 points using pseudo-random Monte Carlo.

A More Complicated Example

To explore the potential benefits of using quasi-random numbers in realistic applications, we implemented a procedure for obtaining the dissociation rates for state-to-state chemical kinetics in joint work of Camberos, Josyula, and Lambe, a part of which we will summarize here.

State-specific dissociation rates coefficients of \( N_2 - N_2 \) collisions were calculated based on the semiclassical theory of Macheret and Adamovich who adapted a FORTRAN program by Billing. We adapted the program by replacing calls from a pseudo-random number generator to a specially adapted eight dimensional quasi-random number generator. Figure (4) shows that the Quasi-Monte Carlo adaptation using only 750 samples compares well with the Monte Carlo method using 5000 pseudo-random samples. Numerically, the norm of the difference of the output is 0.0000000000309.

Experiments with DSMC

The First Naive Try

In our first experiment, we simply replaced all of the calls to the PRNG by calls to a van der Corput sequence. Not only was there no improvement in this case, the resulting algorithm did not converge! We actually anticipated this result however. It is due to a well-known problem in trying to apply quasi-random sequences in Monte Carlo algorithms which we believe was first reported by Morokoff and Caflisch in 1993 in studying QMCMs for the heat equation.

Fixing the First Problem

The classic problem mentioned above is due to the following number theoretic fact concerning the van der Corput sequence base two. One has

\[ \Phi_2(2n) < \frac{1}{2}, \quad \Phi_2(2n + 1) \geq \frac{1}{2}. \]  \hspace{1cm} (13)

These inequalities doom failure any naive attempt to replace a PRNG by the van der Corput sequence or any other quasi-random sequence that has hidden correlations built in due to the mathematical nature of the algorithm used to implement them.

A general way to fix such problems is to scramble the quasi-random sequence – a procedure that goes at least back to Cranley and Patterson in one form in 1976 and Braaten and Weller in another form in 1979. The second form was considerably generalized by Owen. A detailed analysis of these processes is beyond the scope of the present note, but we will give an intuition behind the latter process. Recall that the building blocks of the van der Corput sequence base \( b \) are the digits \( \{0, \ldots, b - 1\} \) that make up the given integer \( n \) base \( b \).
Suppose that there are \( \beta \) such digits in the expansion of \( n \). Instead of constructing the number in \([0,1]\) as van der Corput does, scramble the digits first. That is, choose \( \beta \) permutations \( \sigma_1, \ldots, \sigma_{\beta} \) of the digits randomly and apply them to each digit in order before doing the construction. Note that there are \( \beta! \cdot 3^{\beta} \) different possibilities for each \( n \) using this scheme just in dimension one! Note also that this is the first time that probability theory has entered the subject of quasi-random numbers. As will be seen from our second experiment below, this addition of randomness to quasi-random sequences is enough to cause the algorithm to converge once more.

The Second Try

Having seen for ourselves the subtlety of applying quasi-random sequences in accept/reject algorithms, we went one step further. The first two places where the PRNG is called require three calls. The first two are for the two random particles and the third is for the accept/reject inequality. In the quasi-random setting, this should be seen as a three-dimensional process. The reason for this is what is known as the fundamental theorem of simulation as it is called in the book by Christian and George.\(^{26}\) viz. choosing a sample \( x \) according to a given distribution \( \rho \) is equivalent to conditionally sampling a pair \((x,u)\) uniformly such that \( u < \rho(x) \). In fact, this theorem can be seen to lead directly to the classic accept/reject algorithm. Next, we replace the two calls to the PRNG for the two angles by a two-dimensional quasi-random sequence. This resulted in a computation that converged, but there was only a very slight improvement in the rate of convergence as we report below.

The Third Try

It has been observed\(^{39–41}\) that the accept/reject algorithm for drawing a sample according to a given distribution corresponds to a discontinuous integrand and as such, the Koksma–Hlawka does not apply. In\(^{41}\) however, Moskowitz and Caflisch also showed that the situation could be smoothed so that the Koksma–Hlawka applies to the new situation. This is done at the expense of introducing weights for the samples drawn. McNenly\(^{24}\) also observed the inherent discontinuities of accept/reject and found a successful adaptation of a Monte Carlo method for the special case of gas flow he studied by eliminating the accept/reject stage using another algorithm; however that method does not adapt to the general case.

Weighted particle methods for approximating the Boltzmann have already been introduced by Rjasanow and Wagner.\(^{42–45}\) Since it was fairly trivial to adapt our previous code to the setting of smoothed accept/reject, we decided to run this sort of experiment as well. There was an issue about how the weights should evolve in Bird’s approach and we formulated an ad hoc algorithm for doing this. We make no claim that our ad hoc algorithm is related to the work of Rjasanow et al. just cited.

Previous Results on DSMC and Quasi-Random Sequences (DSQMC)

In the late 1980 and early 1990s a number of authors investigated the effect of adapting Nanbu’s algorithm to the quasi-random setting. Results by Babovsky et. al.\(^{21}\) and Lécot\(^{46}\) were fairly elaborate and the results were apparently not as striking as the difference that quasi-random sequences make with respect to Monte Carlo integration. In fact Bird comments on the work by Babovsky et. al. in chapter 10 of his book.\(^{2}\) A much more recent and successful attempt to adapt stochastic methods for gas dynamics to the quasi-random setting is the work of McNenly and Boyd.\(^{24, 25}\) McNenly studies the special case of collisionless flow and derives an algorithm that achieves near linear error convergence rate using quasi-random sequences. We decided to begin a sequence of experiments involving the full algorithm of Bird in this context. Our first results were a bit surprising and we will report them below in some detail followed by an analysis that will indicate why the results were as we found them and will also indicate our plans for current and future experiments in this area.

Test Case: Supersonic Flat Plate

We compared the results of calculations using the supersonic leading-edge problem as described by Bird\(^{2}\) pp. 340-348. The flow conditions simulate the supersonic flow of nitrogen gas at Mach 4, Knudsen number 0.0143, with free-stream number density at \(10^{20}\) particles per cubic meter, temperature 300 K, and initial velocity at 1412.5 m/s. The geometry is a flat plate of unit length (1 meter) corresponding to about 70 molecular mean-free-paths for the given Knudsen number.

Using Bird’s DSMC2.FOR code, we first verified that we could obtain the same results as Bird then modified calls to the pseudo-random number generator to use LDS numbers instead and implemented the smoothed accept/reject algorithm. Our investigation included: Does the LDS method yield significant improvement in convergence rate? Does the LDS method improve accuracy with less sampling? We describe the results herein and provide our answers below.

As described by Bird, supersonic flow over this simple geometry is characterized by a weak oblique shock wave induced by the viscous boundary layer at the surface. Using 37,000 simulated particles, we obtained the
a) Density contours for Mach 4 flow of $N_2$ over flat plate.

b) Mach contours for flow of $N_2$ over flat plate

Figure 5  The viscous boundary layer induces a weak oblique shock wave in supersonic flow.

The results for Mach number, mass density, temperature, and pressure fields over the flat plate surface. The flow field is essentially two-dimensional and colored contour plots are shown in Figure (5(a)) for the mass density and in Figure (5(b)) for the Mach number. The results are converged after about 10,000 iterations (time steps in the DSMC algorithm). We used a uniformly-spaced grid with 100 cells in the horizontal $x$-direction and 60 cells in the vertical $y$-direction. The figure is scaled such that the flat plate leading edge begins at $x = 0.10$ and the orientation of the flow is from left to right in the figures displayed. The weak oblique shock wave is clearly evident above the flat plate surface as a red stripe, originating near the leading edge and progressing diagonally, gradually thickening towards the exit edge. The viscous boundary layer is visible along the surface of the plate as curved contour lines in Figure (5(a)) and more so as a blue thin strip representing low subsonic flow just above the plate in Figure (5(b)).

To gauge convergence, we used two metrics using differences in local values of the dependent variables, including mass density, pressure, temperature, and velocity components. Results were similar for these except for the vertical component of velocity in the $y$-direction, which we discuss below. The maximum norm, defined as

$$\max_{\forall x, y} |\rho_{n+1} - \rho^n|$$

where $\rho$ is the mass density, $n$ is the iteration number, and the difference taken over the entire flow field. Similarly for the remaining flow variables. We include also results from a coarser calculation, using only 18,000 particles compared to the benchmark number of 37,000. We found that there was essentially no difference between the standard pseudo-random DSMC calculations and our implementation of the quasi-random DSQMC. There was a slight difference in convergence rates in both when using 18,000 particles compared to 37,000. The results are given in Figure (6(a)), which shows a drop of about 5 orders of magnitude $L_\infty$ norm after 10,000 iterations. The pseudo-random and quasi-random results are overlaid in the figure. Green and black lines correspond to the pseudo-random calculations for 18,000 and 37,000 particles respectively. The underlying blue and red lines correspond to the quasi-random calculations for 18,000 and 37,000 particles respectively. Another measure of convergence is denoted by the root-mean-square difference, or $L_2$ norm given as

$$\frac{1}{N} \sqrt{\sum (\rho_{n+1} - \rho^n)^2}$$

where $N$ is the total number of cells. This is a smoother measure of variation from one iteration to the next and is shown in Figure (6(b)) for the mass density. Again, the pseudo-random and quasi-random results are overlaid in the figure. Green and black lines correspond to the pseudo-random calculations for 18,000 and 37,000 particles respectively. The underlying blue and red lines correspond to the quasi-random calculations for 18,000 and 37,000 particles respectively. The $L_2$ norm drops about 4 orders of magnitude after 10,000 iterations. Results were similar for the other dependent variables except the vertical $y$-velocity component.

Interestingly, the vertical velocity component showed a clear difference in the maximum norm between the conventional pseudo-random DSMC and our quasi-random DSQMC calculations. The results were the same when using 18,000 and 37,000 particles. The convergence rate difference is more than one order of magnitude as can be discerned in Figure (7(a)). In this figure, the green and black lines correspond to the pseudo-random
a) Convergence rate using maximum difference in mass density ($L_\infty$ norm).

b) Convergence rate using root-mean-square (rms) difference in density ($L_2$ norm)

Figure 6  Simulation convergence rates show no significant difference between conventional DSMC and LDS-based DSQMC with smoothed A/R.

a) Convergence rate using maximum difference in y-velocity ($L_\infty$ norm).

b) Convergence rate using root-mean-square (rms) difference in y-velocity ($L_2$ norm)

Figure 7  Simulation convergence rates show no significant difference in the $L_2$ norm between conventional DSMC and LDS-based DSQMC with smoothed A/R. Some difference is evident in the $L_\infty$ norm although it remains inconclusive as to whether this is significant or not.

DSMC with 18,000 and 37,000 particles respectively. These drop about 4 or 5 orders of magnitude after 10,000 iterations. The blue and red lines corresponding to the quasi-random DSQMC with 18,000 and 37,000 particles respectively. These clearly show a faster convergence with the same number of particles for our quasi-random DSQMC calculations. We may explain this difference by noting that a portion of the core algorithm requires uniform sampling from a sphere to compute the post-collision particle directions. The conventional DSMC algorithm of course uses pseudo-random sampling which we replaced with quasi-random sequences. We found no more perceptible differences between our experiments in replacing this part of the algorithm with quasi-random sampling and incorporating ad hoc smoothed accept/reject for particle collisions. We must conclude that the improvement here is due mostly to the decrease in discrepancy for a more uniformly random sample of the sphere to obtain post-collision particle directions. This advantage, however, is not carried over to the root-mean-square measure, as shown in Figure (7(b)) which again shows that the pseudo-random and quasi-random calculations converge at about the same rate. The blue lines correspond to the pseudo-random DSMC and the red lines...
The pressure distributions along the plate show some difference when comparing conventional DSMC with the quasi-random DSQMC results. The differences correspond to quasi-random DSQMC in the figure. The difference is the same as before when using 18,000 compared to 37,000 particles.

Besides convergence rates, we also wanted to see if we could get better accuracy in terms of the aerodynamic quantities of interest. We obtained pressure, shearing, and heating results over the flat plate surface and compared the pseudo-random DSMC against the quasi-random DSQMC. Here the differences were more pronounced although comparison with experiment or other validated data is desirable. For these figures, the horizontal axis is scaled to match the unit-meter length of the plate. Results for the pressure coefficient are shown in Figure (8(a)) for the pseudo-random DSMC. The blue line and triangles overlap the red line and circles, showing that there’s no difference in the results when using 18,000 or 37,000 particles. This result agrees with that presented by Bird. We use the standard definition for the pressure coefficient as

\[ C_p = \frac{p - p_\infty}{\frac{1}{2} \rho_\infty U_\infty^2} \]  \hspace{1cm} (16)

where the free-stream values are \( \rho_\infty, p_\infty \), and \( U_\infty \) for density, pressure, and velocity respectively. With the quasi-random DSQMC, the pressure coefficient in Figure (8(b)) shows a stronger peak at the leading edge and there is a slight difference when using 18,000 particles (blue triangles) compared to 37,000 particles (red circles).

Results for the friction and heating coefficients are directly compared for the two methods in Figure (9). We use the standard definitions for friction and heating coefficients, given by

\[ C_f = \frac{\tau_w}{\frac{1}{2} \rho_\infty U_\infty^2} \]  \hspace{1cm} (17)

\[ C_h = \frac{q_w}{\frac{1}{2} \rho_\infty U_\infty^3} \]  \hspace{1cm} (18)

where the subscript \( w \) indicates the quantity at the wall (surface of the plate). The shear stress and heating, \( \tau \) and \( q \) respectively, are obtained by adding the resultant incident and reflected values, which is consistent with Bird’s calculations. For the friction coefficient in Figure (9(a)), results are shown in blue triangles for pseudo-random DSMC, which match Bird’s results. The red circles represented the calculated results for the friction coefficient using the quasi-random DSQMC. Differences were minor for the two methods when using 18,000 compared to 37,000 particles. The heat transfer coefficient along the flat plate is shown in Figure (9(b)), again with blue triangles representing the pseudo-random DSMC and the red circles representing the quasi-random DSQMC. To determine which method is “more accurate”, comparison with experiments is desirable.
Comparison of friction coefficient with conventional DSMC and quasi-random DSQMC.

Comparison of heat transfer coefficient with conventional DSMC and quasi-random DSQMC.

Figure 9 The friction and heat transfer coefficients along the plate show some difference when comparing conventional DSMC with the quasi-random DSQMC results.

Direct Methods

The direct method\textsuperscript{4, 7–10} for approximating solutions to Boltzmann’s equation involves approximating the Boltzmann collision integral directly and goes back to the 1960s.\textsuperscript{7, 8} Deshpande and Narasimha\textsuperscript{8} actually derive an analytic expression for the collision integral in the special case when the initial distribution is a sum of Maxwellians. In that case, the collision integral is explicitly

\[
\frac{1}{\pi} \int_{\mathbb{R}^3} \int_0^{2\pi} \int_0^\pi ||\tilde{w}|| e^{h(u, \tilde{w}, \phi, \psi)} e^{-||\tilde{w}||^2} \sin \phi d\phi d\psi d\tilde{w}
\]  

where \( u \) is a parameter, \( \phi \in [0, \pi], \psi \in [0, 2\pi], \tilde{w} \in \mathbb{R}^3 \), and the function \( h \) is given by

\[
h(u, \tilde{w}, \phi, \psi) = -2u^2 + u (w_1 + w_2 + (w_2 - w_1) \cos \phi) + u \sin \phi \sin \psi \left( \sqrt{w_2^2 + w_3^2} + \frac{||\tilde{w}|| w_3 \cot \psi + w_1 w_2}{\sqrt{w_2^2 + w_3^2}} \right).
\]

The value of this integral at \( u = 0.20 \) is 2.4793846754936771044. We use importance sampling with respect to the distribution \( e^{-||\tilde{w}||^2} \) in the integrand and compare Monte Carlo integration to Quasi-Monte Carlo integration for the evaluation of this integral. Using the known value from the analytic solution, we can compare the errors using

\[ A \]

\[ B \]

\[ C \]

\[ D \]

\[ E \]

\[ F \]

\[ G \]

\[ H \]

\[ I \]

\[ J \]

\[ K \]

\[ L \]

\[ M \]

\[ N \]

\[ O \]

\[ P \]

\[ Q \]

\[ R \]

\[ S \]

\[ T \]

\[ U \]

\[ V \]

\[ W \]

\[ X \]

\[ Y \]

\[ Z \]
both methods as the number of samples increases. The graph of the analytic solution as a function of $u$ as well as the error graph comparing MCI to QMCI are given in Figure(10). Notice that the quasi-random integration has essentially converged at about 2500 samples whereas the pseudo-random integration is still oscillating at 5000 samples. This is a considerable reduction of work.

A general theory for importance sampling in the direct evaluation of the Boltzmann collision integral has been developed in a series of papers$^{9,10}$ in which it is shown that if the importance sampler is chosen carefully, a considerable reduction in the number of samples for Monte Carlo integration can be made. We believe that this reduction can be lowered still if these methods are combined with smoothed accept/reject in conjunction with quasi-random methods for drawing the samples. Results of experiments along these lines will be reported elsewhere.

Conclusions

We have noted that some caution must be used in adapting Monte Carlo methods to the quasi-random case. In cases where the Koksma–Hlawka inequality applied, we saw a great improvement in the adapted algorithm. It has been observed$^{39–41}$ that the accept/reject algorithm for drawing a sample according to a given distribution corresponds to a discontinuous integrand and as such, the Koksma–Hlawka inequality does not apply. They further go on to derive smoothed accept/reject algorithms that show considerable improvement using quasi-random samples. As we pointed out, McNenly$^{24}$ also observed the inherent discontinuities of accept/reject and found a successful adaptation of a Monte Carlo method for the special case of gas flow he studied by eliminating the accept/reject stage using another algorithm. We began a sequence of experiments to see what could be done to possibly improve the full Bird algorithm and have seen, consistent with other researchers, that simply replacing calls to a PRNG can give incorrect results. Correcting that by scrambling can lead to correct results, but not necessarily a vast improvement in DSMC. Adapting the direct method to the quasi-random setting in connection with smoothed accept/reject for importance sampling however has a very promising outlook.

Acknowledgements

The authors would like to thank the many fruitful discussions with numerous researchers about the subject matter. Also, we gratefully acknowledge the support of the USAF SBIR program which partially supported the development of the LDS mathematical library by MSSRC.

References


