Journal of Computational Physics 229 (2010) 5857–5861

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

Short note

Searching for a near neighbor particle in DSMC cells using pseudo-subcells

M.N. Macrossan *

Centre for Hypersonics, School of Engineering, University of Queensland, Australia

ARTICLE INFO

Article history:
Received 12 June 2009
Received in revised form 22 January 2010
Accepted 26 April 2010
Available online 6 May 2010

Keywords:
Collision partner
Near neighbor
Collision selection
DSMC
Virtual subcells
Mean collision separation

ABSTRACT

LeBeau et al. (2003) [4] introduced the 'virtual-subcell' (VSC) method of finding a collision partner for a given DSMC particle in a cell; *all* potential collision partners in the cell are examined to find the nearest neighbor, which becomes the collision partner. Here I propose a modification of the VSC method, the 'pseudo-subcell' (PSC) method, whereby the search for a collision partner stops whenever a 'near-enough' particle is found, *i.e.* whenever another particle is found within the 'pseudo-subcell' of radius δ centered on the first particle. The radius of the pseudo-subcell is given by $\delta = Fd_n$, where d_n is the expected distance to the nearest neighbor and F is a constant which can be adjusted to give a desired trade-off between CPU time and accuracy as measured by a small mean collision separation (MCS). For 3D orthogonal cells, of various aspect ratios, $d_n/L \approx 0.746/N^{0.383}$ where N is the number of particles in the cell and L is the cube root of the cell volume. There is a good chance that a particle will be found in the pseudo-subcell and there is a good chance that such a particle is in fact the nearest neighbor. If no particle is found within the pseudo-subcell the closest particle becomes the collision partner.

To limit the CPU time required for large N the search is restricted to a subset of all particles in the cell; the nearest particle from that subset becomes the collision partner. Here the VSC search is restricted to 29 of the remaining particles and the CPU time never increases beyond what is required for $N = 30$. For $N > 30$ the restricted search is as accurate as a standard subcell method using 34 subcells in a 3D cell. The PSC search surveys up to 33 possible collision partners, to yield the same limiting value of MCS, and still save some CPU time. For $F = 1.1$, PSC uses between 12% and 20% less CPU than VSC while the accuracy is within 4% of that for VSC.

1. Introduction

The collision routine in DSMC requires that, for each potential collision, two particles in a cell be chosen at random. Meiburg [5] suggested that, because angular momentum is not conserved in a simulated collision, there can be significant errors if two particles undergoing a collision are located at opposite sides of a typical DSMC cell. To overcome this problem Bird [1] introduced subcells within each cell. In a subcell method (i) the first potential collision partner is chosen at random from the entire cell, (ii) its subcell is determined (or has been predetermined for all particles) and (iii) a potential collision partner is chosen from the same subcell, or from a nearby subcell if no collision partner can be found in the same subcell. Thus the use of subcells reduces the mean collision separation (MCS) to a fraction of the subcell size, rather than a fraction of the cell size.

* Tel.: +61 7 3365 4513; fax: +61 7 336 54799.
E-mail address: m.macrossan@uq.edu.au

0021-9991/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2010.04.042

LeBeau et al. [4] introduced what they called the virtual-subcell (VSC) method; for each attempted collision the first particle is selected and the distances to all remaining possible collision partners in the cell are calculated; the nearest neighbor particle becomes the second particle in the collision pair.[1] LeBeau et al. [4] report that this VSC procedure is the standard method used in NASA's 'state of the art implementation of the DSMC method' known as DAC, and that the procedure is computationally competitive with subcell methods. For Bird's code, Gallis et al. [3] report that, for fewer than 30–40 particles in a cell, the virtual-subcell method (nearest neighbor selection) is faster than the subcell method which is used in that code when there is a large number of particles in the cell. The subcell method compromise on accuracy, as measured by a small value of MCS, in order to reduce the computational effort when the number of particles in the cell is large.

2. The pseudo-subcell method

Here I describe a method of reducing the CPU requirement of the virtual-subcell method. The new method, which I will designate as the pseudo-subcell (PSC) method, is easily adjustable to give a range of accuracy (as measured by the MCS) and computational expense.

The new method starts by searching for the nearest neighbor in the entire cell, just as in the VSC method, but the search is terminated whenever a 'near-enough' particle is found, *i.e.* the search is finished if another particle is found within a distance δ of the first selected particle, where δ is the required 'near-enough distance'. We can think of the sphere of radius δ surrounding a particle as its 'pseudo-subcell', and the search stops when a particle is found within this pseudo-subcell. The volume of the pseudo-subcell is $v_{sc} = 4\pi\delta^3/3$ and δ might be chosen such that $v_{sc} = V/N$, where N is the number of particles and V is the total cell volume. Of course if δ is set too small no particle may be found in the pseudo-subcell, and in that case all other particles are eventually surveyed and the closest one becomes the collision partner. That is, the pseudo-subcell method becomes the same as the virtual-subcell method as $\delta(N/V)^{\frac{1}{3}} \to 0$.

For a given pseudo-subcell volume v_{sc} we can estimate how often the search for a near-enough neighbor will in fact find the nearest neighbor. There are $N-1$ possible collision particles in the cell. Assuming these particles are randomly distributed within the entire cell the chance that a given one of them lies within the pseudo-subcell is

$$p = \frac{v_{sc}}{V}. \tag{1}$$

The chance that a total of $k = 0, 1, 2, \ldots, N-1$ of these particles are within the pseudo-subcell is given by the binomial probability

$$P_k = \frac{(N-1)!}{k!(N-1-k)!} p^k q^{(N-1-k)}, \tag{2}$$

where

$$q = 1 - p. \tag{3}$$

When none of the $N-1$ particles is in the pseudo-subcell (probability of P_0) the PSC search is exhaustive (equivalent to VSC) and finds the nearest neighbor in the cell. When only one of the $N-1$ particles is in the pseudo-subcell (probability of P_1) that particle must be the nearest neighbor, and the PSC search will find it. When there are two possible collision particles in the pseudo-subcell (probability of P_2) one of them must be the nearest neighbor and there is a 1 in 2 chance that the PSC search will select that one; when there are three particles (probability of P_3) there is a 1 in 3 chance and so on. Hence the total probability that the particle found by the PSC search will be the nearest neighbor is

$$P(\text{nearest found}) = P_0 + P_1 + \frac{1}{2}P_2 + \frac{1}{3}P_3 + \cdots + \frac{P_{N-1}}{N-1}. \tag{4}$$

For the above choice of $v_{sc} = V/N$, giving the probability $p = 1/N$, Eq. (4) indicates that for $N = 5$ particles in the cell there is more than 90% chance that the nearest neighbor particle will be found. This reduces to a 86% chance for $N = 40$.

The CPU time required for PSC compared to VSC is related to the probability that the PSC search will be shorter than the VSC search. A full PSC search occurs when there are no particles within the pseudo-subcell (probability P_0) so

$$P(\text{short search}) = 1 - P_0 = 1 - q^{N-1} = 1 - \left(1 - \frac{1}{N}\right)^{N-1}.$$

Thus, if there are $N = 5$ particles in the cell, the chance that the PSC search will be shorter than the full VSC search is approximately 59%, and for $N = 40$ the chance of a short search is approximately 63%.

[1] By restricting collisions to the nearest neighbor, the number of possible discrete values of relative velocity in collisions is reduced from $N(N-1)/2$ to N, where N is the number of particles in the cell, which might seem a severe restriction on the method. However, the required distribution of relative velocities in collisions will be realized as long as the separation distance between a pair of particles is independent of the relative velocity between the particles. In addition the usual restriction on the DSMC time step (less than the local mean free time) ensures that fewer than half of the N possible near-neighbor collision pairs in the cell will undergo collisions at any time step.

3. Mean distance to nearest neighbor and pseudo-subcell radius δ

Particles near the edges and corners of the cell have fewer than average nearby particles within the same cell, so we cannot expect the distance to a near neighbor to be exactly proportional to $1/N^{\frac{1}{3}}$. The mean distance to the nearest neighbor d_n, over 2×10^7 random trials, was found for orthogonal cells of various aspect ratios, with varying numbers of particles in the cell. The mean (expected) distance to the nearest neighbor is very accurately given by

$$\frac{d_n}{L} = \frac{a}{N^b} \tag{5}$$

and a and b are constants and the characteristic length $L = V^{\frac{1}{3}}$ for 3D cells, or $L = A^{\frac{1}{2}}$ for 2D cells, where A is cross-sectional area of the 2D cell. Values of a and b for 3D cells ($V = \Delta x \Delta y \Delta z$) and 2D cells ($A = \Delta x \Delta y$) of various shapes are given in Table 1. The values of a and b for the cube and the square (the first row of each table) agree with what can be deduced from Fig. 1 of Ref. [4]. The last line of each table shows values of a and b which I suggest are accurate enough for cells of any near-orthogonal shape.

The pseudo-subcell radius is set to a value of

$$\delta = F d_n = F \frac{a}{N^b} L, \tag{6}$$

where F is a constant (a scale factor) which can be selected to give the required level of accuracy or saving in CPU time.

4. Performance compared to VSC method

By numerical experiment we can find how the mean collision separation, and the CPU time depend on the scale factor F. The results are shown in Fig. 1 for a 3D cell (with $\Delta x{:}\Delta y{:}\Delta z = 1{:}2{:}3$) and a 2D cell (with $\Delta x{:}\Delta y = 1{:}2$). There is a trade-off between a smaller value of MCS and larger CPU time for both 3D and 2D cells. For 3D cells, and a 'tight' value of $F = 0.9$ the pseudo-subcell method is in effect equivalent to VSC, but slightly faster, particulary for larger numbers of particles in the cell, *i.e.* the MCS is within 1% of d_n found by VSC, and there is a saving of about 5% in CPU time. If one is prepared to accept a 5% increase in MCS (compared to VSC) a factor $F = 1.1$ is a good choice; the CPU saving is about 8–17% for 3D cells and about 10–18% for 2D cells.

5. For N large

Bird et al. [2] say the VSC method is not prohibitively expensive in CPU time, compared to a subcell method, for up to 30 particles in a cell. Here the VSC search is restricted when the number of particles in the cell is greater than 30, *i.e.* for $N = 30$, all 29 possible collision partners are examined in the VSC method, and for $N > 30$ only 29 of the possible collision partners in the cell are examined. Thus the CPU time never exceeds that required for VSC with 30 particles in the cell.

For $N > 30$ the PSC search is truncated whenever a particle is found within a distance

$$\delta_{\text{lim}} = 1.1 \frac{a}{30^b} L, \tag{7}$$

corresponding to $N = 30$. Because the PSC method requires less CPU time than the VSC method, the search length for PSC can be longer. It was found that if the PSC search was limited to 33 remaining particles in the cell while the limiting pseudo-subcell radius of Eq. (7) was used to truncate the search, the limiting value of the mean collision separation was the same on average as for VSC with its search restricted to 29 particles.

Table 1
Fitted constants a and b in Eq. (5) for cells of various aspect ratios.

	a	b
$\Delta x{:}\Delta y{:}\Delta z$		
1:1:1	0.735	0.381
1:2:1	0.744	0.383
1:2:2	0.746	0.384
1:2:3	0.758	0.387
All	0.746	0.383
$\Delta x{:}\Delta y$		
1:1.0	0.660	0.557
1:1.5	0.663	0.558
1:2.0	0.669	0.560
1:3.0	0.698	0.570
All	0.672	0.561

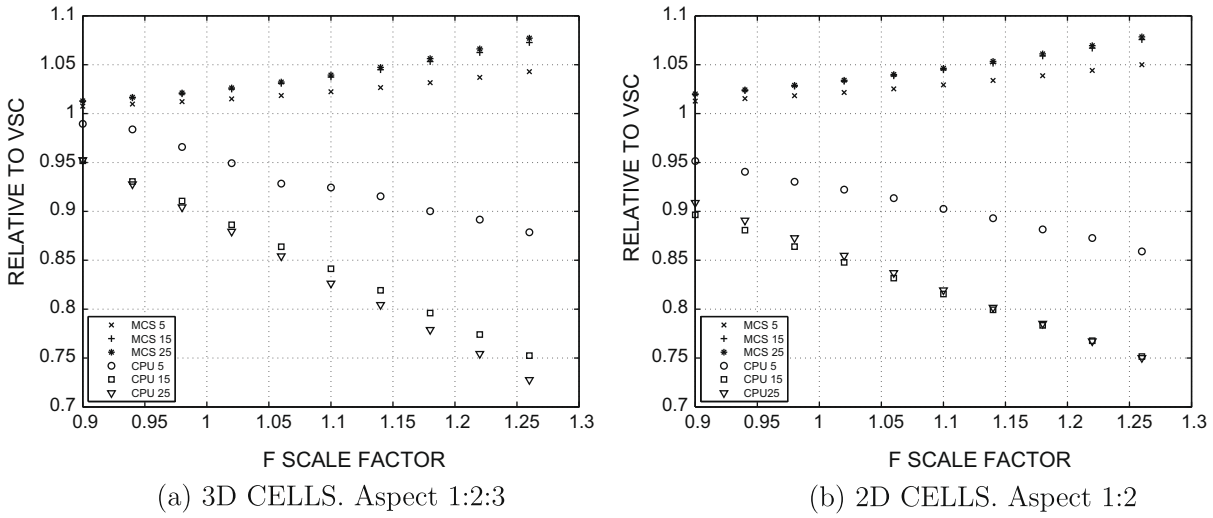

Fig. 1. Mean collision separation (MCS) and CPU time for PSC (relative to VSC values) as scale factor F is changed. $N = 5$, 15 or 25 particles/cell.

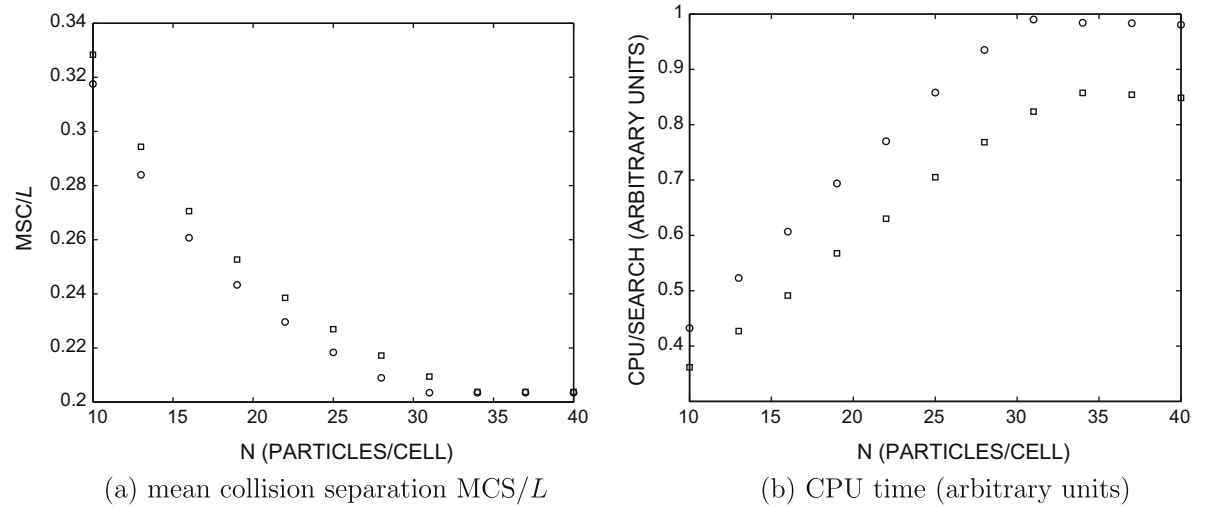

Fig. 2. Virtual-subcell (○, VSC) and pseudo-subcell method (□, PSC) compared. N is particles/cell. Search length limited to 29 (VSC) and 33 (PSC). PSC scale factor $F = 1.1$.

For both VSC and PSC, it was assumed that the list of all particles in the cell contained the particles in random order. For any particle chosen as the first member of a potential collision pair, the search for a collision partner started with the next particle in the list, and looped back to the start of the list if necessary.[2] Thus for each potential collision a different random selection of the remaining particles in the cell is searched. If there is any correlation between the position of the particles in physical space and the ordering of the particles in the list of particles within the cell, then extra calls to the random number generator will be required and the CPU time could become prohibitively large.

Fig. 2(a) shows the MCS value for both methods, coming to the same limiting value of $0.203L$ (for 3D cells with aspect ratio $\Delta x{:}\Delta y{:}\Delta z = 1{:}2{:}3$). This limiting value of MCS is similar to that for a standard subcell method using cubic subcells of side length $\ell = 0.307L$, for which the expected MCS would be $0.6615\ell = 0.203L$. The number of such subcells in the cell would be $L^3/\ell^3 \approx 34.6$. Thus, VSC and PSC, with the parameters described above, are slightly more accurate than a standard subcell method using 34 subcells/cell. The CPU times for PSC and VSC are compared in Fig. 2(b). The saving in CPU for PSC varies

[2] The same search procedure was used for all values of N; all that is required is that the search limit be set as the minimum of $N - 1$ or the limiting value of 29 (for VSC) or 33 (for PSC). For $N \leqslant 30$ this search strategy was found to be as fast as a search which began at the start of the list, and proceeded to the end of the list (while skipping the one particle in the list already selected for collision).

from 12% to 20% over the range of N shown, while the CPU time for PSC with $N > 30$ is no more than is required for VSC with 25 particles/cell.

6. Cells of other shapes

For each type of cell empirical tests are required to find the relationship between d_n and N. Numerical experiments show that, for randomly shaped triangular cells, with a ratio of longest to shortest side less than 6 and angles between adjacent sides greater than 6°, values of $a = 0.81$ and $b = 0.61$ in Eq. (5) predict the average distance to the nearest neighbor to within ±4.2%, for $10 < N < 30$. With these values for a and b a scale factor of $F = 0.7$, in Eq. (6) gives a value of MCS for the PSC method no more than 4% greater than for the VSC method, and a potential CPU saving of 15% (this CPU estimate is based on the average search length for PSC compared to the $N - 1$ steps for the full VSC search; the exact CPU saving will depend on coding details). There seems to be no reason why the PSC method cannot be applied to different cell shapes.

7. Conclusion

For orthogonal cells, the pseudo-subcell method gives a 5% increase in MCS compared to the virtual-subcell method, and is up to 20% faster than the virtual-subcell method. With the restricted search for large values of N the CPU time for PSC is never more than required for VSC with 25 particles in the cell, and is slightly more accurate as a standard subcells method which uses 34 subcells/cell. Users of DSMC can experiment to find the values of F and the search limit which suit their needs for accuracy and CPU time; they can also judge for themselves if the added complexity of retaining in their code a standard subcell method, using more than 34 subcells/cell, is justified.

References

[1] G.A. Bird, Molecular Gas Dynamics and the Direct Simulation of Gas Flows, Clarendon, Oxford, 1994.
[2] G.A. Bird, M.A. Gallis, J.R. Torczynski, D.J. Rader, Accuracy and efficiency of the sophisticated direct simulation Monte Carlo algorithm for simulating non-continuum gas flows, Physics of Fluids 21 (2009) 017103.
[3] M.A. Gallis, J.R. Torczynski, D.J. Rader, G.A. Bird, An improved-accuracy DSMC algorithm, in: T. Abe (Ed.), Rarefied Gas Dynamics, 26th International Symposium, AIP Conference Proceedings, vol. 1084, 2009, pp. 299–304.
[4] G.J. LeBeau, K.A. Boyles, F.E. Lumpkin III, Virtual sub-cells for the direct simulation Monte-Carlo method, AIAA paper 2003-1031, 2003.
[5] E. Meiburg, Comparison of the molecular dynamics method and the direct simulation technique for flows around simple geometries, Physics of Fluids 29 (1986) 3107–3113.