Hashing-Cells Combination for Boundless Space Event Driven Molecular Dynamics

Mauricio Marín[1,2] and Patricio Cordero[3]

[1] Computing Department, University of Magallanes, Casilla 113-D, Punta Arenas, CHILE
[2] Present address: Computing Laboratory, University of Oxford,
Wolson Building, Parks Road, Oxford OX1 3QD, ENGLAND
[3] Physics Department, University of Chile, Casilla 487-3, Santiago, CHILE
E-mails: mmarin@ona.fi.umag.cl, pcordero@tamarugo.cec.uchile.cl

Abstract

This paper presents a cell method to perform event driven simulations of hard-particle systems on boundless space. An infinite cell division of the space is supposed where only the cells that contain particles are maintained in the computer memory. In order to detect efficiently these non-empty cells a standard hash method is used. This solution enables a more efficient $O(1)$ simulation of these N-particle systems since the alternative approach is use no cells at all, which leads to $O(N)$ simulations per each processed event.

1 Introduction

Event-driven molecular dynamics is a well established tool for the study of hard-particle fluids [1]. Particles move free of interparticle forces during finite intervals and at discrete instants (*events*) one or two particles suffer an impulsive force (the respective momenta change discontinuously). The evolution of the system during these intervals is trivial and the simulation proceeds jumping analytically from one event to the next. Applications of this type of simulations are illustrative and inspiring since the beginning of molecular dynamics till present days [1]-[13]. Efficient algorithms to simulate large systems of hard-particles can be found in [14]-[17].

A key feature in the efficiency of these simulations is the strategy used to find out the particles in the neighborhood of every particle under consideration. One first step is to divide the space in cells and look for the particles present in the home cell c where the particle under consideration is and also for the particles inside the cells adjacent to c. However, no solution for the case of boundless space has been proposed yet.

This paper presents an algorithm and data structure that enables the implementation of the cell method in boundless space systems. Our strategy assumes an infinite cell division of the space and uses a standard hashing method to maintain in the computer memory only the cells that contain

particles. Hashing enables the efficient search for those cells during the process of determining the particles located in the neighborhood of a given particle.

In [16] it is commented that in boundless space simulations it is not possible to apply the standard cell division method, and therefore it is necessary to study future collisions (events) for every particle with all the other $N-1$ particles of the system. This leads to a rather than large event-list if all the future events are allowed to reside in it. In such a case, however, it is more convenient to modify the standard cell method, by using hashing [19], in order to reduce the memory consumed by the event-list and the overall running time, since to calculate collisions with $N-1$ particles for each event that takes place is by far lesser efficient than calculating collisions with $O(1)$ neighboring particles (cell division enables us to calculate future collisions with only $O(1)$ neighboring particles.) Note that the memory used by the combination hashing-cells is similar to the standard cell method required in systems with hard or periodical boundaries.

2 Cell Division on Boundless Space

The existence of an infinite cell system is assumed when particles are free to move in boundless space. We describe the cell strategy for 2D systems while the extension to 3D should be direct from the 2D presentation.

The only cells kept in the computer memory are the non-empty ones regardless of the values of the cell coordinates (c,r). This is implemented by keeping *lists of cells* so that two cells (c_1,r_1) and (c_2,r_2) belong to the same list if and only if

$$(\,|c_1|,|r_1|\,) \bmod (p,p) = (\,|c_2|,|r_2|\,) \bmod (p,p)\,,$$

and using a $p \times p$ matrix H to reach these cell-lists.

Each element $H[a,b]$ of the matrix H is a *pointer* to the first member in the list of cells and each member of these lists is the 4-tuple $(c,r,next,first)$ where (c,r) are the coordinates of the cell, *next* is a pointer to the next member in the cell-list and *first* is the identifier of the first particle in the *list of particles* inside (c,r).

The lists of particles associated to the cells are kept in an array A of length N where each element $A[i]$ is a structured variable $A[i]=(next,c,r)$. The particles in every cell are ordered in an arbitrary way. The cell element equals $(c,r,next,i)$ while $A[i]$ is (j,c,r) if j is the label of the next particle in the cell (c,r). If k is the *last* particle in a cell (c,r) then $A[k]=(0,c,r)$. Operations of inserting (deleting) a particle label into (from) the data structure are necessary when a particle enters (leaves) a cell.

The function $h(i)=(|i| \bmod p)+1$ is used to map from (c,r) to (a,b). So the search for a cell with coordinates (c,r) can be performed in the cell-list pointed to by $H[a=h(c),b=h(r)]$. In this way, the matrix H enables us to use *direct chaining hashing* [19] to reach any cell with coordinates (c,r) regardless of the values of c and r. There is no *a priori* condition over the value of p but for better performance it should usually be close to $\lfloor \sqrt{N} \rfloor + 1$ to get shorter cell-lists.

To obtain the displacements of the neighboring cells in collision events and the displacements of the new neighboring cells in cell crossing events the sets

$$D^+ = \{\,(0,0),\,(1,0),\,(1,-1),\,(0,-1),\,(-1,-1),\,(-1,0),\,(-1,1),\,(0,1),\,(1,1)\,\}$$

and

$$\begin{aligned} D_0^{\#} &= \{\,(-1,-1),\,(0,-1),\,(1,-1)\,\} \\ D_1^{\#} &= \{\,(-1,1),\,(0,1),\,(1,1)\,\} \\ D_2^{\#} &= \{\,(1,1),\,(1,0),\,(1,-1)\,\} \\ D_3^{\#} &= \{\,(-1,1),\,(-1,0),\,(-1,-1)\,\} \end{aligned}$$

are used. The set D^+ contains the relative cell displacements for collision events and $D_p^{\#}$ the relative displacements for cell crossing events through a wall $p = Up = 0$, $Down = 1$, $Right = 2$ and $Left = 3$ respectively.

The list of neighboring particles for a particle i in the cell (c, r) after i has suffered a collision, is obtained by:

a) searching every cell (f, g) in the cell-lists pointed to by each $H[h(f), h(g)]$, where

$$(f, g) = (c + d, r + e) \;\; \forall \;\; (d, e) \;\in\; D^+ \,,$$

b) searching for all the particles in the particle-lists associated to each cell (f, g). The particles are collected from the array A and each entry to A is identified by the field *"first"* in each (f, g).

Note that some cells (f, g) could be absent from the lists of cells pointed to by each $H[h(f), h(g)]$ if they are empty cells. The list of new neighboring particles after a cell crossing event through a wall p is obtained by the same procedure but using the set $D_p^{\#}$.

3 Conclusions

We propose a hashing based implementation of the cell method for simulations on boundless space. This is a better alternative than using no division for these systems since both the memory required to maintain all the events and the total simulation running time are reduced. Using cell division implies to maintain $O(1)$ events per particle and $O(1)$ calculations of new events every time that an event takes place, which is better than $O(N)$ for both cases when no division is used [16].

Acknowledgements

Work partially funded by Chilean projects U.Mag. F1-O1IC-95 and Fondecyt 19311105.

References

[1] B.J. Alder and T.E. Wainright, *J. Chem. Phys.* **27**, 1208 (1957).

[2] B.J. Alder and T.E. Wainright, *J. Chem. Phys.* **31**, 459 (1959).

[3] M. Mareschal and E. Kestemont, *Phys. Rev. A* **30**, 1158 (1984).

[4] E. Melburg, *Phys. Fluids* **29**, 3107 (1986).

[5] M. Mareschal and E. Kestemont, *J. Stat. Phys.* **48**, 1187 (1987).

[6] A. Puhl, M. Malek-Mansour and M. Mareschal, *Phys. Rev. A* **40**, 1999 (1989).

[7] M. Mareschal, M. Malek-Mansour, A. Puhl and E. Kestemont, *Phys. Rev. Lett.* **41**, 2550 (1988)

[8] D.C. Rapaport, *Phys. Rev. Lett* **60**, 2480 (1988).

[9] D.C. Rapaport, *Phys. Rev. A* **43**, 7046 (1991); *Phys. Rev. A* **46**, 1971 (1992).

[10] D. Risso and P. Cordero in *Condensed Matter Theories* Vol. 7, Edited by A.N. Pronto and J. Aliaga (Plenum, New York, 1991).

[11] D. Risso and P. Cordero in *Instabilities and Nonequilibrium Structures* E. Tirapegui and W. Zeller (eds) (Kluwer Acad Press, 1992)

[12] J. Koplik, J.R. Banavar and J.F. Willemsem *Phys. Rev. Lett.* **60**, 1282 (1988).

[13] M.Alaoui and A. Santos, *Phys. Fluids A* **4**, 1273 (1992).

[14] D.C. Rapaport, Journal of Computational Physics, **34** (1980) 184.

[15] B.D. Lubachevsky, Journal Computational Physics, **94** (1991) 255.

[16] K. Shida and Y. Anzai, Computer Physics Communications, **69** (1992) 317.

[17] M. Marín, D. Risso and P.Cordero, Journal of Computational Physics, **109** (1993) 306.

[18] M. Marín and P.Cordero, Computer Physics Communications, **92** (1995) 214.

[19] G.H. Gonnet and R. Baeza-Yates, *Handbook of Algorithms and Data Structures*, (Addiddison-Wesley, 1991).