JOURNAL OF COMPUTATIONAL PHYSICS 105, 367–368 (1993)

NOTE

A Note on Algorithms for Billiard-Ball Dynamics

In a recent article, Lubachevsky [1] described a computational scheme for efficiently simulating hard-disk and hard-sphere fluids. The paper includes a statement to the effect that, apart from one of the original papers [2] that introduced the method and a later review article [3], there has been no other discussion in the literature of appropriate algorithms. The purpose of this note is threefold: to remind the readers of this journal that an article on the subject was published here in 1980 [4] (see also [5]), to briefly compare and contrast the principal features of the two algorithms, and to provide relative performance figures that will aid the interested reader in deciding which approach is preferable.

Both algorithms treat the problem as a discrete-event simulation, which is the only way the dynamics of systems based on discontinuous step potentials can be sensibly handled. The system evolves in time by means of a series of pairwise collisions. At each collision the velocities of the colliding particles change according to elementary dynamical rules, and then the future collisions in which the colliding particles may be involved are examined. To reduce the amount of computation required per collision to a manageable level, namely a constant $O(1)$ rather than $O(N)$ for an N-particle system, the spatial region occupied by the system is divided into small cells (or sectors) and future collision surveys are confined to particles in the same and adjacent cells. Cell crossings also represent events that must be monitored. The details of the scheme ensure no collisions are missed.

The other similarity between the methods is in the management of the set of possible future events. In order to limit the work needed to determine the next event, the calendar of future events is maintained as a binary tree (or heap). Since the total number of recorded future events is typically $O(N)$—exactly one per particle in [1], or some small number per particle in [4]—the binary tree [6] facilitates this determination in an average of $O(\log N)$ operations.

The principle difference between the algorithms is the role of the event as an element of the computation. In the new study [1] only the earliest possible collision event for each particle is recorded, with the implication that potentially useful information obtained about other possible collisions is discarded on the assumption that the earliest event is unlikely to be preempted. If it is, then the discarded computations are, in effect, repeated; the degree to which this occurs has not been studied. Furthermore, events that are preempted are still allowed to occur, but are no longer treated as collisions, but merely as position changes for the particle involved. By way of contrast, the old algorithm [4] does not discard information about subsequent possible collisions, and when a collision does occur all other scheduled events in which the particles involved were due to participate are explicitly erased from the event calendar. This reflects a design goal of minimizing the amount of work (the aptly-named "aggressive" approach rather than the "lazy" scheme adopted in [1]), and the performance figures (see later) apparently justify this goal. Other implementation issues are to be found in the original paper [4]. The only disadvantage is the memory needed to store the larger event set, but falling memory costs reduce the significance of this issue; both methods choose to sacrifice storage in pursuit of computational speed, although to differing degrees.

Unlike the new approach, which does not guarantee monotonically increasing time between events (with consequences that must be addressed) and which also requires a lengthy discussion to convince the reader of its consistency and correctness, the old approach is based on molding the data structures to fit the computational model in the most natural manner possible; the operations required to accommodate the slightly increased data complexity of the old method, though somewhat tedious, are entirely standard [7].

The bottom line is, of course, performance. Molecular dynamics is being used to study problems for which both space and time scales are ever increasing. Thus the processing time per collision is the key factor in choosing an algorithm. Performance figures for the new approach are quoted for a DEC VAX 8550 computer. Results for the old method obtained on a machine of similar power—the IBM 4381/13—will be given here. While the only truly reliable way to compare algorithm performance is by running the programs on each of the machines concerned, a reasonable alternative in the case of non-vector machines is to use the standard Linpack performance measurements [8]; in this instance the means of single and double precision Fortran tests conveniently amount to 1.2 MFlops for both machines.

The new method yields a collision rate of between 150 and 450 collisions/s (for $N = 2000$—the parameter that

0021-9991/93 $5.00

modifies the rate is not specified). The old method, coded in standard Fortran, delivers approximately 670 collisions/s (for $N = 2500$), a value that is almost constant over a range of densities in the dense fluid regime (e.g., 0.9 disks/unit area) and even faster at lower densities. Other performance figures (on an older computer and in three dimensions) were given in [4]; assembly language coding—especially of the data structure manipulations—can lead to even higher speeds, provided software protability is not an issue. Thus, despite the extra event information stored, the old method appears significantly faster.

As for applications, the approach is readily extended to handle such systems as polymers and a thermally convecting fluid in a gravitational field [9]. In ongoing studies of the Rayleigh–Bénard problem, systems ranging in size to over 10^5 particles are being simulated for several times 10^9 collisions on an IBM 6000/320 workstation. Such sizes—currently regarded as extreme, but destined to become the norm given the inexorable advance of the high-performance workstation—are necessary to allow the study of spatio-temporal phenomena (such as time-dependent convection or polymer transport) that embrace a broad range of length and time scales.

REFERENCES

1. B. D. Lubachevsky, *J. Comput. Phys.* **94**, 255 (1991).
2. B. J. Alder and T. E. Wainwright, *J. Chem. Phys.* **31**, 459 (1959).
3. J. J. Erpenbeck and W. W. Wood, in *Modern Theoretical Chemistry*, edited by B. J. Berne, Vol. 6B (Plenum, New York, 1977), p. 1.
4. D. C. Rapaport, *J. Comput. Phys.* **34**, 184 (1980).
5. D. C. Rapaport, *Comput. Phys. Rep.* **9**, 1 (1988).
6. D. E. Knuth, *Sorting and Searching, The Art of Computer Programming*, Vol. 3 (Addison–Wesley, Reading, MA, 1973).
7. D. E. Knuth, *Fundamental Algorithms, The Art of Computer Programming*, Vol. 1 (Addison–Wesley, Reading, MA, 1968).
8. J. J. Dongarra, Argonne National Lab. Math. and Comput. Sci. Tech. Memo No. 23, 1989 (unpublished).
9. D. C. Rapaport, *J. Chem. Phys.* **71**, 3299 (1979); *Phys. Rev. Lett.* **60**, 2480 (1988).

Received June 29, 1991; accepted August 23, 1992

D. C. RAPAPORT

Physics Department
Bar-Ilan University
Ramat-Gan 52900
Israel