JOURNAL OF COMPUTATIONAL PHYSICS **108**, 368–372 (1993)

NOTE

A Technique for Regularizing the Structure of a Monotonic Lagrangian Grid

1. INTRODUCTION

In many applications of molecular dynamics, Lagrangian fluid dynamics, and other types of object-tracking problems, it is necessary to keep track of particles, fluid elements, or more generally, nodes which are separated by less than a specified physical cutoff distance R_c. One approach is to calculate the distances between all pairs of nodes and compare these separations to R_c. This process ensures that all neighboring nodes are found, but because the operation count scales as N^2, where N is the total number of nodes, it becomes impractical for large systems. Much work has been done to develop algorithms which are more efficient in determining near neighbors, including the particle-particle-particle-mesh (PPPM) algorithm described by Hockney and Eastwood [1] and trees [2]. A drawback of the PPPM method is that it does not adapt to vector or parallel machines with full efficiency. Gunsteren *et al.* [3] have developed algorithms based on neighbor list techniques which vectorize well, but which are prohibitive for large systems because of the large storage requirements.

Algorithms based on the monotonic Lagrangian grid (MLG) [4–7] have been developed in which the operation count scales as $N \log N$. This method has the advantage that it adapts well to vector and parallel machines and requires minimal memory. The MLG has been shown to perform well on vector machines for the test problems of identifying near-neighbors, accessing near-neighbor data, and ordering of near-neighbors according to distance [8]. Each node is assigned a set of grid indices which are related to the relative spatial coordinates of the nodes. Nodes which are in MLG order satisfy the $N(3 - 1/N_x - 1/N_y - 1/N_z)$ monotonicity conditions,

$$\begin{aligned} x(i,j,k) &\leqslant x(i+1,j,k), \quad 1 \leqslant i \leqslant N_x - 1 \quad (\text{all } j, k)\\ y(i,j,k) &\leqslant y(i,j+1,k), \quad 1 \leqslant j \leqslant N_y - 1 \quad (\text{all } i, k) \qquad (1)\\ z(i,j,k) &\leqslant z(i,j,k+1), \quad 1 \leqslant k \leqslant N_z - 1 \quad (\text{all } i, j), \end{aligned}$$

where (N_x, N_y, N_z) define a three-dimensional data structure, (i, j, k) are the grid indices, and $N_x \times N_y \times N_z = N$ is the total number of nodes. The primary advantage of a well-structured MLG is that nodes which are nearby in physical space are also nearby in index space. Neighboring nodes can be found by searching index space with a maximum index offset N_c, rather than searching physical space with a cutoff distance R_c.

For most applications of the MLG, we are interested in the time evolution of a system in which the nodes can move significantly so that their neighbors change continually in time. An MLG can be restructured by exchanging, or "swapping," data stored in adjacent nodes until the monotonicity conditions of Eqs. (1) are satisfied. This method of grid restructuring is very efficient since local violations of monotonicity can be corrected without triggering global changes in the MLG. Because data for nodes which are adjacent in index space are stored in contiguous memory, codes based on the MLG vectorize and parallellize well. The MLG is particularly well suited for massively parallel machines such as the Thinking Machine CM-200 since data for adjacent nodes is stored in adjacent processors. The performance of MLG-based codes on the CM-200 is not hampered by the fact that in most applications a maximum index offset greater than one is required. A tour algorithm using repeated near-neighbor communications to follow a continuous path through the nearby processors very efficiently allows index offsets greater than unity. After communications are completed to copy necessary information into the relevant neighboring processors, all computations are then carried out within the processors.

Since the physical positions of the nodes define the grid, the MLG is well suited to problems in which there can be significant variation in the local density of the system. Molecular dynamics (MD) simulations have been performed on shock-induced detonations in solids [9] and $(N_2)_2$ [10] dimer formation on vector machines. More recent work on massively parallel machines includes shock-defect interaction in Lennard-Jones solids [11] and applications to related problems in battle management [12]. The MLG can be used both for systems with fixed and variable numbers of particles. The addition or deletion of particles is handled by the use of holes, empty nodes in the MLG structure.

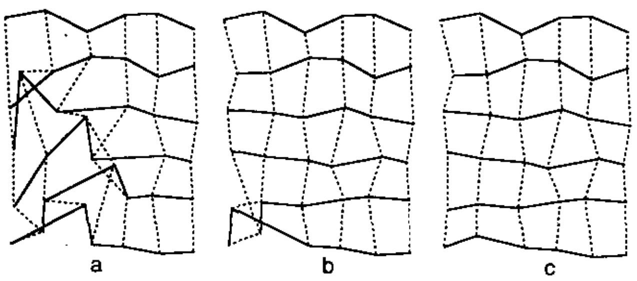

FIG. 1. Examples of three different monotonic Lagrangian grids based on the same set of node locations. Solid and dashed lines represent x and y links, respectively. Grid (a) was obtained by sorting nodes into MLG order from a random order. Grids (b) and (c) are derived from grid (a) by one and two iterations, respectively, of SGR.

An important property of the MLG is that the ordering of the nodes within the data structure, subject only to the conditions of Eqs. (1), is not unique so the overall quality of the MLGs can vary significantly for a single spatial distribution of nodes. This is best illustrated by an example for a 6×6 system of nodes as shown in Fig. 1. A serious drawback of an arbitrarily chosen MLG is the possibility of near misses. This occurs when nodes which are separated by less than the cutoff distance R_c in physical space have MLG indices which differ by more than the maximum index offset N_c. Previous work [7] studied the statistical properties of the MLG, including an analysis of the probability of a near miss as a function of the maximum index offset. This probability can be made vanishingly small by increasing the volume of index space to be searched. Unfortunately, the computational costs associated with the short range interactions increase proportionally with the volume of index space which must be treated.

This note presents an easily applied, inexpensive method of find a well-structured MLG, thereby reducing the maximum index offset required to give an acceptably small near-miss probability. This method takes advantage of the fact that the MLG ordering is not unique and that it is possible to restructure locally from a poorly to a well-structured MLG for a given set of spatially distributed nodes.

2. GRID RESTRUCTURING TECHNIQUE

The usual method for restructuring the grid involves swapping the node data until the monotonicity conditions in Eqs. (1) are satisfied. Even though the nodes may be in MLG order, a much more satisfactory MLG often exists. In the proposed method, several extra inexpensive restructuring steps are added. A single iteration of the technique, which is referred to as stochastic grid restructuring (SGR), consists of the following steps:

(1) Nodes are randomly displaced in space, retaining unperturbed positions.

(2) Data is swapped until monotonicity conditions are satisfied for the perturbed node positions.

(3) Starting from the MLG ordering established in the preceding step for the perturbed node positions, data is swapped until the monotonicity conditions are satisfied for the unperturbed node positions.

The perturbed node positions are obtained by independently choosing displacements along each coordinate axis from a uniform random distribution. The magnitude of the maximum node displacement along a single coordinate axis is the critical parameter in the SGR technique. Multiple iterations of the SGR on a single simulation timestep may be applied to further improve grid structure.

Well-structured MLGs appear to be statistically more stable than poorly structured MLGs. The usual grid restructuring method, however, has a tendency to allow fewer optimal MLG orderings to occasionally evolve and the system can become trapped in a relatively unfavorable, but locally stable, MLG configuration. By perturbing the system, much in the manner of simulated annealing, SGR makes it possible for the system to access and therefore settle into a more statistically likely, well-structured MLG.

3. RESULTS

Various statistical quantities can be used to give a good measure of the overall quality of the MLG. One useful set of parameters is the average link lengths between index neighbors in the MLG. The x, y, and z link lengths are defined as

$$\begin{aligned} x_{\text{link}}(i,j,k) &= |\mathbf{r}(i+1,j,k) - \mathbf{r}(i,j,k)| \\ y_{\text{link}}(i,j,k) &= |\mathbf{r}(i,j+1,k) - \mathbf{r}(i,j,k)| \\ z_{\text{link}}(i,j,k) &= |\mathbf{r}(i,j,k+1) - \mathbf{r}(i,j,k)|. \end{aligned} \tag{2}$$

For many cases a well-structured MLG is one where the average near-neighbor link length is a minimum. Another useful set of parameters is the average of the normalized dot products of vectors joining near neighbors with the unit normals. The dot products are defined as

$$\begin{aligned} x_{\text{dot}}(i,j,k) &= \frac{x(i+1,j,k) - x(i,j,k)}{x_{\text{link}}(i,j,k)} \\ y_{\text{dot}}(i,j,k) &= \frac{y(i,j+1,k) - y(i,j,k)}{y_{\text{link}}(i,j,k)} \\ z_{\text{dot}}(i,j,k) &= \frac{z(i,j,k+1) - z(i,j,k)}{z_{\text{link}}(i,j,k)}. \end{aligned} \tag{3}$$

The average dot products give a measure of the directionality of the links joining near neighbors in index space.

For a set of nodes arranged on a simple cubic lattice, i.e., $\mathbf{r} = a(l\hat{i} + m\hat{j} + n\hat{k})$, all dot products and therefore the average dot products equal one. For a random distribution of nodes, the average dot products are a measure of the MLG quality, with larger average dot products generally cooresponding to MLGs with better structure.

Figure 1 shows the restructuring of an MLG using SGR. The values of the average link lengths for grids (a), (b), and (c) are 1.189, 1.056, and 1.029, respectively. The corresponding average dot products are 0.818, 0.923, and 0.980, respectively. To show the advantages of SGR on a simple system of a size typical in a molecular dynamics simulation, we present the results for a $20 \times 20 \times 20$ system of noninteracting nodes. The nodes are initialized at time $t = 0$ on a simple cubic lattice and given random velocities. The nodes are enclosed within a cubic region of space and reflecting boundary conditions are imposed in all three directions. On each timestep of the simulation, the positions of the nodes are updated and the MLG is restructured either by the usual method of swapping nodes until MLG order is established or by the method described in the previous section. The effect of three different conditions on the quality of the resulting MLG were investigated. The first is the magnitude of maximal displacement of the nodes along each coordinate axis. The second is the frequency with which the SGR technique is used to carry out the restructuring. For example, an SGR frequency of 20% means that the optimized method is used on every fifth timestep and the usual method is employed on all other timesteps. The third is the number of iterations allowed on each simulation timestep.

We found that there is an optimal magnitude of node displacement. Figure 2 shows the average values of the link lengths and dot products as a function of the displacement magnitude. The average values are obtained by averaging both over all nearest-neighbor links of the $20 \times 20 \times 20$ MLG and over 250 simulation timesteps, neglecting early timesteps in which the system is still in a highly ordered state. The average dot products and link lengths computed at each timestep along the x, y, and z axes are nearly identical and show little variation over the duration of the calculation. According to these two criteria, the maximum displacement of nodes along each coordinate axis which gives the best MLG is approximately one-half of the initial internode separation. For very large maximal displacements the MLG quality can actually deteriorate. This is consistent with the observation that if a set of nodes is put into random order and sorted, the resulting MLG is often more poorly structured than one that is updated from the previously existing MLG order.

The effect of varying the frequency with which the SGR technique is used is shown in Fig. 3. The best quality grids are obtained when SGR is performed on each timestep. In all cases, using the SGR periodically still results in a better structured grid.

The quality of the MLG structure obtained can be improved by increasing the number of iterations on each timestep. The values of the average dot products increase and the average link lengths decrease as the number of iterations is increased, as shown in Fig. 4. There is an upper limit to the quality of the MLG that can be obtained, which is determined by the spatial positions of the nodes. Beyond 10 iterations of the SGR, the MLG diagnostics converge very slowly toward their maximum values. The addition of one iteration of the SGR per timestep increases the total number

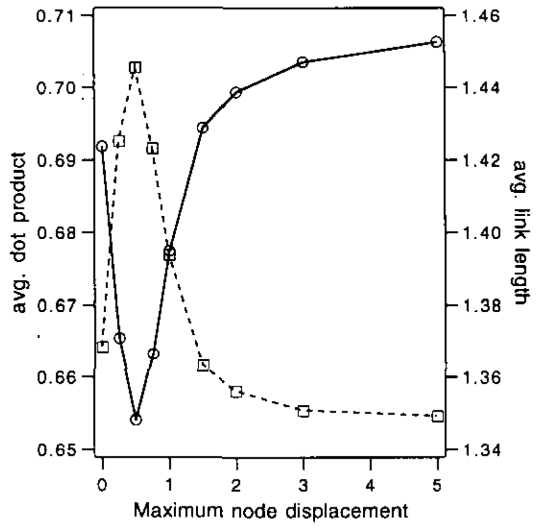

FIG. 2. Average link length (solid line) and dot product (broken line) as a function of maximum node displacement employed in SGR. Maximum displacements given in units of initial internode separation. The values shown are obtained by averaging both over all nearest-neighbor links of the $20 \times 20 \times 20$ MLG and over 250 simulation timesteps.

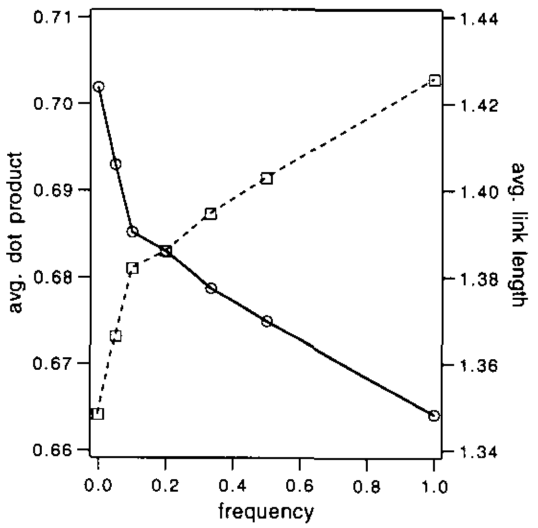

FIG. 3. Average link length (solid line) and dot product (broken line) as a function of frequency of SGR. The values shown are obtained by averaging both over all nearest-neighbor links of the $20 \times 20 \times 20$ MLG and over 250 simulation timesteps.

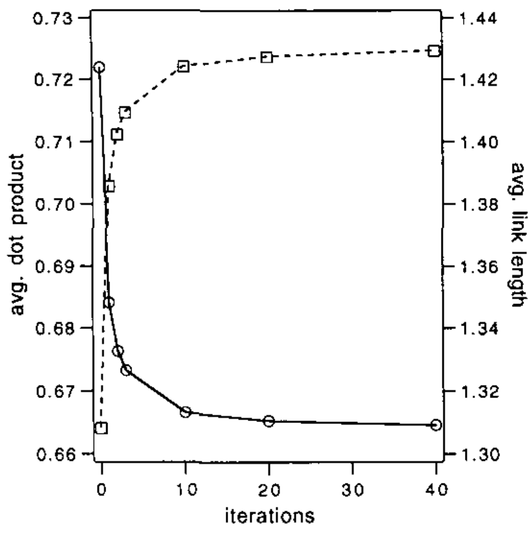

FIG. 4. Average link length (solid line) and dot product (broken line) as a function of number of iterations of SGR used on each timestep. The values shown are obtained by averaging both over all nearest-neighbor links of the 20 × 20 × 20 MLG and over 250 simulation timesteps.

of swaps which must be performed to restore MLG order. This increase is dependent on how badly the nodes are allowed to get out of MLG order between timesteps. In our simulations, using the optimal maximum node displacements, the number of swaps increase by approximately a factor of four. After the first iteration of SGR, the total number of swaps increases linearly with the number of iterations.

Although the average grid diagnostics are a good measure of the overall structure of the MLG and are relatively inexpensive to compute, an inherent weakness is that they may underestimate the influence of small locally distorted regions of the grid. The occurrence of rare events involving nodes with both large index offsets and small spatial separations was monitored in the simulations. The use of SGR has a profound effect on the distance of closest approach for nodes separated by four index offsets. Additional iterations of SGR on each timestep further increase the closest approach of nodes with large index offsets by up to 50%. In cases where the majority of the grid is already well structured, small improvements in the average grid diagnostics are indicative of large improvements in the structure of the most highly distorted regions of the grid.

4. CONCLUSIONS

Using the method described in this paper, it is possible to relax from a poorly structured to a well-structured MLG. The method is easy to apply and the quality of the MLG obtained does not depend significantly on the quality of the random number generator used to perturb the node positions. The advantage of working with a well-structured MLG is that the region of index space which must be included in the search for neighboring nodes can be kept smaller, resulting in decreased computational costs.

In many types of calculations, the grid restructuring takes up only a very small fraction of the computational cost. Although the overhead associated with grid restructuring scales linearly with the number of iterations, in these cases it may be worthwhile to perform several iterations on each timestep. In most applications, such as molecular dynamics, the starting point for a system is very ordered and a high quality MLG can be easily found. For cases where the starting point is a random ordering of nodes, it may be advantageous to continue the grid restructuring process until the MLG diagnostics converge to a predetermined limit and then use a smaller number of iterations on further restructurings.

The SGR has been applied to a two-dimensional molecular dynamics simulation of shock propagation in a lattice involving thousands of interacting particles. The MLG structure obtained in these simulations was often very poor due to the fact that a disordered fluid-like region was in contact with a periodic lattice with nonorthogonal lattice vectors. The SGR not only improved the overall quality of the MLG, but it significantly improved the grid structure in the most highly distorted regions of the MLG. Most impressive, however, was the result that the maximum index offset required for the calculation of the short-range interactions was reduced from $N_c = 8$ to $N_c = 4$. In this simulation the extra cost associated with the SGR was more than balanced by the factor of four reduction in costs associated with calculating the short-range interactions. More dramatic savings should be seen in three-dimensional calculations.

ACKNOWLEDGMENTS

This work was supported by the Office of Naval Research and the Applied and Computational Mathematics Program at DARPA. One of us (R.S.S.) thanks the Office of Naval Technology for support as on ONT postdoctoral fellow. The authors also thank Ronald Kolbe and David Fyfe for helpful insight and discussions concerning the MLG.

REFERENCES

1. R. W. Hockney and J. W. Eastwood, *Computer Simulation Using Particles* (McGraw–Hill, New York, 1981), Chap. 8, pp. 267.
2. D. E. Knuth, *The Art of Computer Programming* (Addison–Wesley, Reading, MA, 1973), Vol. 3, Chap. 6.
3. W. F. Van Gunsteren, H. J. C. Berendsen, F. Colonna, D. Perahia, J. P. Hollenberg, and D. Lellouch, *J. Comput. Chem.* **5**, 272 (1984).

4. J. Boris, *J. Comput. Phys.* **66**, 1 (1986).
5. J. P. Boris and S. G. Lambrakos, in *The Free-Lagrange Method*, edited by M. J. Fritts *et al.* (Springer-Verlag, New York, 1985), p. 158.
6. J. P. Boris and S. G. Lambrakos, in *Proceedings*, 1985 *Summer Computer Simulation Conference, Chicago, Illinois*, 1985 (North-Holland, Amsterdam, 1985), p. 206.
7. S. G. Lambrakos and J. P. Boris, *J. Comput. Phys.* **73**, 183 (1986).
8. J. M. Picone, S. G. Lambrakos, and J. P. Boris, *SIAM J. Sci. Stat. Comput.* **121**, 368 (1990).
9. S. G. Lambrakos, M. Peyrard, E. S. Oran, and J. P. Boris, *Phys. Rev. B* **39**, 993 (1989).
10. D. G. Lambrakos, J. P. Boris, R. H. Guirguis, M. Page, and E. S. Oran, *J. Chem. Phys.* **90**, 4473 (1989).
11. L. Phillips, R. S. Sinkovits, E. S. Oran, and J. P. Boris, NRL Memorandum Report 4440-92-6998, 1992 (unpublished).
12. R. L. Kolbe, J. P. Boris, and J. M. Picone, NRL Memorandum Report 6705, 1990 (unpublished).

Received October 29, 1992; accepted February 11, 1993

R. S. Sinkovits
J. P. Boris
E. S. Oran
Laboratory for Computational Physics and Fluid Dynamics
Naval Research Laboratory
Washington, D.C.