JOURNAL OF COMPUTATIONAL PHYSICS 111, 15–23 (1994)

Calculating Complex Interactions in Molecular Dynamics Simulations Employing Lagrangian Particle Tracking Schemes

J. H. DUNN AND S. G. LAMBRAKOS

Naval Research Laboratory, Washington, DC 20375-5000

Received May 21, 1991; revised March 8, 1993

In this report, we describe some general features of Lagrangian particle tracking and the method of table look-up that are important for calculating many different types of interactions of arbitrary complexity in large molecular dynamics simulations. Based on our experience using these approaches and state-of-the-art of computer technology, we state formally those properties of Lagrangian tracking and table look-up that are important to the design of optimal programming structures. We examine the relationship between Lagrangian indexing and list indexing of data and how this influences the complexity of programming structures for effecting vector operations. Included in this report are criteria for the efficient use of Lagrangian particle tracking schemes in molecular dynamics simulations. © 1994 Academic Press, Inc.

I. INTRODUCTION

In molecular dynamics simulations of systems consisting of many different types of complex molecules, many different types of potential energy functions must be evaluated at each time step. In such simulations, the evaluation of the total force on an individual particle may require the evaluation of a wide range of expressions which consist of terms containing fractional powers or transcendental functions. In addition, for a given pair of interacting particles, the appropriate potential energy function and associated parameters must be identified according to particle type. In general, molecular dynamics simulations of complex systems present three design problems: an efficient algorithm for evaluating a range of complicated expressions requiring many operations, an efficient procedure for identifying the type of interaction for any given pair of particles in the model system, and architecture independence of programming structures, e.g., design of programming structures which are well suited to vector or scalar architectures. In principle, the method of table look-up represents a general approach to addressing the first problem; however, in practice, large numbers of particle types can result in programming structures consisting of conditional operations that limit the efficient utilization of sequential or parallel architectures.

For a given method of function evaluation, e.g., table look-up, the efficiency of any procedure for computing potential energy or force magnitude will depend on the efficiency of accessing or "gathering" information concerning particles which are "near" neighbors of a given particle, e.g., a vector of positions and possibly orientations. List indexing schemes require that a vector of pointers to this information be explicitly computed, stored, and retrieved from each of a particle's near neighbors. Lagrangian schemes employ an implicit pointer scheme which obviates the need for computation, storage, and retrieval of pointers to near neighbors, at the cost, however, of computing the interactions with a greater number of neighbors. Although the type of "gather" operations employed in any given procedure will depend on the nature of the computer architecture, it usually involves accessing memory which is external to the processor computing the value of the potential energy or force magnitude. Further, the term vector of quantities, e.g., indices, in no way implies that the discussion is limited to vector architectures. Rather, it indicates that the operation in question is a mapping from $\mathfrak{R}^1$, the set of particles, to $\mathfrak{R}^n$, a set of particles associated with any given particle.

In this paper, we analyze the computational complexity of procedures that employ Eulerian and Lagrangian particle tracking schemes for accessing information about particles in molecular dynamics simulations along with table look-up. We present a general procedure for optimizing Lagrangian indexing of data in conjunction with table look-up. We outline the properties of Lagrangian tracking schemes in the context of contemporary computational environments. Previous reports [1–3] give a detailed analysis of many of the properties of Lagrangian particle tracking; however, based on our experience using these schemes as well as the current state of computer technology, we show that some of the constraints on Lagrangian tracking outlined in those reports are no longer essential for effective implementation of Lagrangian tracking schemes. Through formal definitions and conditions, we present the

0021-9991/94 $6.00

necessary conditions on Lagrangian tracking for the design of efficient programming structures in molecular dynamics simulations. In particular, we present a criterion for minimizing the number of superfluous operations performed due to Lagrangian tracking.

II. BACKGROUND

A. *Eulerian Particle Tracking*

An Eulerian particle tracking (EPT) scheme is any method which uses a condition based explicitly on particle separation for determining the "near" neighbors of any given particle, p, where near neighbors are defined to be those particles whose interactions with the given particle are non-negligible. Notable among such methods are procedures for constructing neighbors or linked lists [4–6]. An example of particle tracking based on neighbors list method is the construction of atom-site maps. Beeler [7, 8] describes the use of atom site maps for molecular dynamics modelling of bcc, fcc, and hcp crystal structures. In this section, we present the aspects of EPT which contribute to its computational complexity, examine programming structures for its implementation and derive expressions for its computational cost with respect to computing particle–particle interactions.

Before proceeding to the analysis of the computational complexity of Eulerian particle tracking, we define computational cost, C_A, of an algorithm, A, to be the sum of the computational costs, c_i, of each of the N_t tasks associated with it. Further, the cost of each task is simply the product of the order of its computational complexity, i.e., the order of the leading term in the polynomial expression for operation count, of the task, o_i, and the amount of computer time required for each operation, s_i. Therefore,

$$C_A = \sum_{i=1}^{N_t} s_i o_i. \tag{1}$$

In the case of molecular dynamics simulations employing EPT, the computation of particle–particle interactions requires four distinct computational tasks: construction of the neighbor's list, accessing positions of neighbors, evaluation of the force magnitude or potential energy and updating the partial forces or potential energies of the neighbors as well as the particle of interest, p. In general, the first task will be performed at most every few time steps. The remaining three tasks must be performed every time step for each particle.

Before examining each computational task explicitly, we outline the general structure of the molecular dynamics simulation, specifically the evaluation of particle–particle interactions. Although this treatment does not necessarily represent an analysis of optimal task partitioning for Eulerian tracking, these aspects will contribute to the complexity, and hence the cost, of any specific programming structures corresponding to optimal task partitioning for schemes employing EPT.

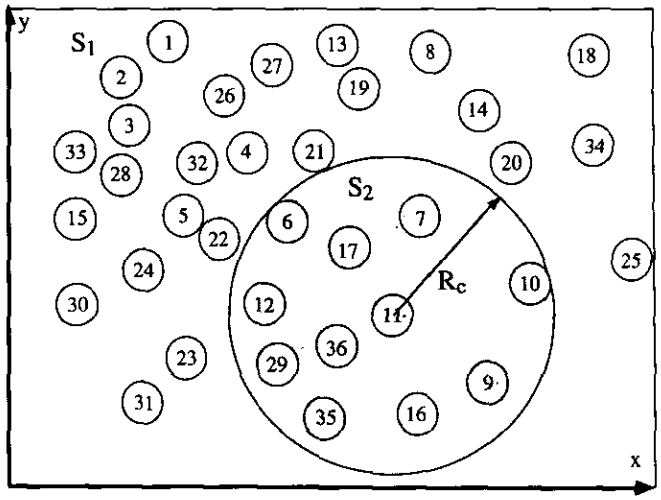


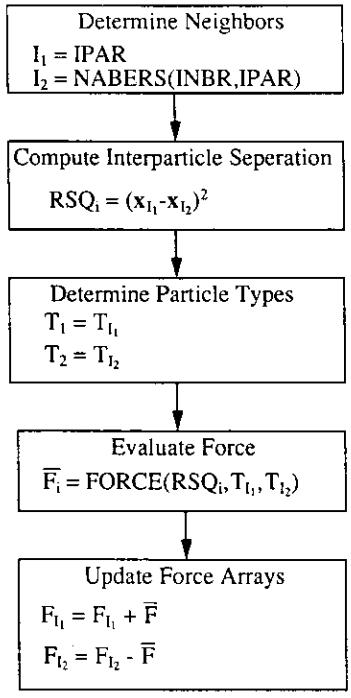
FIG. 1. Close neighbors to particle IPAR for IPAR = 11.

For simplicity, we will consider only a bounded region of two-dimensional space. The discussion which follows is identical for three dimensions. Consider a region, S_1, which contains 36 particles and represents part of the xy-plane (see Fig. 1). There exists a subregion S_2 that contains all particles considered close neighbors of the particle labeled 11 in Fig. 1. In Eulerian tracking schemes, the neighbors of a given particle are defined as those particles having a spatial separation less than some specified distance, R_c in Fig. 1. Given this condition, one constructs a list of neighbors for each particle in the system. One may then access information concerning any of the close neighbors of the particle of interest.

Figure 2 depicts the EPT mapping of S_1 onto computer memory along with the mapping of S_2 represented by the

1	2	3	4	5	6
7	8	9	10	(11)	12
13	14	15	16	17	18
19	20	21	22	23	24
25	26	27	28	29	30
31	32	33	34	35	36

FIG. 2. Close neighbors of IPAR in computer memory when Eulerian tracking is employed.



FLOWCHART 1. General programming structure associated with Eulerian tracking.

shaded boxes. A programming structure for the evaluation of particle–particle interactions is outlined in flowchart 1. The variable labeled IPAR is the particle identifier (PID) of particle p and corresponds to particle 11 in Fig. 1 and Fig. 2. The array labeled NABERS holds the list of the PIDs of the close neighbors of all system particles indexed by the PID of each particle. Although our example depicts the neighbor's list method, the complexity of the method is the same for either linked list or neighbor's list methods. The variable INBR varies from one to the number of particles less one within the grey shaded region in Fig. 1.

We may now analyze the computational complexity of each of the four tasks required for the implementation of the programming structure outlined above and how they relate to the computational cost of computing particle–particle interactions in the context of EPT.

The first task is the construction of a list of PIDs corresponding to the near neighbors of the particle of interest. With respect to computational complexity, the construction of either neighbor's lists or linked lists is equivalent to performing a sort. The complexity of these algorithms, and therefore task 1 is

$$o_1 = N_p \log N_p, \tag{2}$$

where log refers to the base 2 logarithm and N_p is the number of particles in the system. Second, the positions of the $N_c(p)$ close neighbors of each particle p must be accessed. Clearly, the complexity of this task is

$$o_2 = \sum_{p=1}^{N_p} N_c(p). \tag{3}$$

The complexity of the third and fourth computational tasks is the same as that of the second. Thus the computational cost of particle–particle interaction evaluation is

$$C_{\mathrm{EPT}} = \alpha_1 s_1 N_p \log N_p + \alpha_2 \sum_{j=2}^{3} s_j \sum_{p=1}^{N_p} N_c(p), \tag{4}$$

where s_1 is the time required to generate one neighbor's list element, s_2 is the time to gather the particle positions and update the partial forces, s_3 is the time to evaluate force magnitude or potential energy. The factors α_1 and α_2 are the number of times a task must be performed during a simulation divided by the number of time steps in the simulation resulting in the expected number of tasks performed per step. In general, α_2 will be one; however, α_1 depends on the type of physical system being modelled. For solids, α_1 will tend to be much smaller than one, but in the case of liquids it may be quite close to one.

B. *Lagrangian Particle Tracking*

Based on the following definitions, we present the aspects of Lagrangian particle tracking (LPT) which contribute to its computational complexity, examine programming structures for its implementation and derive expressions for its computational cost with respect to computing particle–particle interactions.

DEFINITION 1. Given a system of N particles in a bounded region of three-dimensional space, a Lagrangian tracking scheme is one which establishes a one-to-one mapping between N spatial coordinates (X_i, Y_j, Z_k) and N Lagrangian coordinates (i, j, k) (where $i = 1, ..., N_x$, $j = 1, ..., N_y$, $k = 1, ..., N_z$, and $N = N_x N_y N_z$), subject to the ordering conditions

$$\begin{aligned} X_i &\leqslant X_{i+\Delta i} \\ Y_j &\leqslant Y_{j+\Delta j} \\ Z_k &\leqslant Z_{k+\Delta k}, \end{aligned} \tag{5}$$

where Δi, Δj, and Δk are fixed integral differences that are greater than or equal to one.

DEFINITION 2. A Lagrangian attribute is a particle attribute which is identified only by its Lagrangian coordinates (i, j, k).

The conditions given by Eq. (1) permit one to adopt an implicit criterion based on the relative separation of par-

ticles for determining the neighbors of any given particle in a bounded region of space. In Lagrangian tracking schemes, the neighbors of a given particle are defined as those particles for which the difference of their Lagrangian coordinates is less than some specified difference. This criterion is not a sufficient condition for tracking near neighbors, but is a sufficient condition for tracking "closest" neighbors. In particular, this criterion is sufficient for tracking the set of "first closest" neighbors, the set of "second closest" neighbors, etc. LPT tracks a set of neighbors of a given particle of which a proper subset contains all near neighbors. Any operations used to determine this subset would be associated with Eulerian tracking. Thus, in Lagrangian schemes, one always tracks a set of particles that is larger than the set of neighbors that are close.

In the case of molecular dynamics simulations employing LPT, the process of computing particle–particle interactions is composed of three distinct computational tasks: sorting the Lagrangian coordinate system subject to the conditions in (1), evaluating the force magnitude or potential energy, and updating the partial forces or potential energies of the neighbors as well as the particle of interest, p. In general, the first task will be performed at most every few time steps. The remaining two tasks must be performed every time step for each particle.

Before examining each task in detail, we outline the general structure of the molecular dynamics simulation. Again, we consider the two-dimensional bounded region in Fig. 1. In LPT schemes, the neighbors of a particle are defined to be those particles whose separation, in each Lagrangian coordinate, is less than some cutoff, $N_L(p)$. Note that the Lagrangian cutoff is defined in terms of a vector of Lagrangian coordinates rather than a scalar magnitude. This simplifies programming structures used in this technique. Thus, the number of particles interacting with particle p is $(N_L(p)+1)^3-1$.

Figure 3 represents the mapping of the regions S_1 and S_2, in Fig. 1, onto computer memory. In a Lagrangian scheme, the coordinates of a particle's closest neighbors are stored in contiguous memory locations. The programming structures necessary to implement this scheme are outlined in Flowchart 2. Note that Flowchart 2 represents a three-dimensional implementation of Lagrangian tracking in a molecular dynamics simulation. In Fig. 3, the dark border around the closest neighbors of particle 11, whose Lagrangian coordinates are *ijk* in Flowchart 2, indicates the extent of the Lagrangian "neighbors template." The integer pairs in each location represent the Lagrangian coordinates of that particle. The template in Fig. 3, corresponds to the range of variables $i2$, $j2$, and $k2$ in Flowchart 2. For example, $i2$ would vary from i to $i+2$. Note also that this template can be characterized by a single number which is the analogue of a "cutoff distance" in Lagrangian coordinates and is two in our example. Using a template to index

2 (6,1)	1 (6,2)	27 (6,3)	13 (6,4)	8 (6,5)	18 (6,6)
33 (5,1)	3 (5,2)	26 (5,3)	19 (5,4)	14 (5,5)	34 (5,6)
28 (4,1)	32 (4,2)	4 (4,3)	21 (4,4)	7 (4,5)	20 (4,6)
15 (3,1)	5 (3,2)	22 (3,3)	6 (3,4)	17 (3,5)	25 (3,6)
30 (2,1)	24 (2,2)	12 (2,3)	36 (2,4)	(11) (2,5)	10 (2,6)
31 (1,1)	23 (1,2)	29 (1,3)	35 (1,4)	16 (1,5)	9 (1,6)

FIG. 3. Close neighbors of IPAR in computer memory when Lagrangian tracking is employed.

eliminates the need for a linked or neighbors list and, accordingly, reduces the computational complexity of the algorithm.

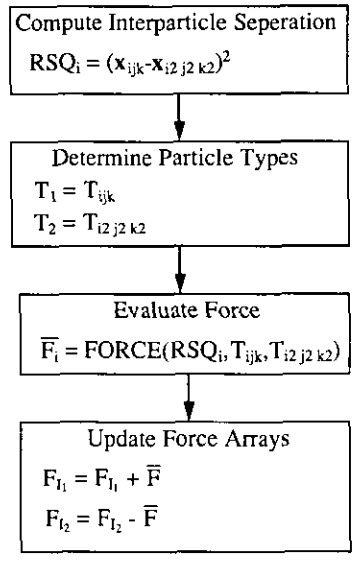
We now analyze the computational complexity of each of the three tasks required for the implementation of the programming structure outlined above and their effect on the computational cost of computing particle–particle interactions in the context of LPT.

The first task is the sorting of the Lagrangian coordinates subject to (5). The complexity of this algorithm is

$$o_1 = N_p \log N_p. \tag{6}$$

The complexity of the second and third computational tasks is

$$o_2 = \sum_{p=1}^{N_p} [(N_L(p)+1)^3 - 1]. \tag{7}$$



FLOWCHART 2. General programming structure associated with Lagrangian tracking.

For the case where $N_L(p)$ is a constant,

$$o_2 = N_p[(N_L(p)+1)^3 - 1]. \tag{8}$$

Thus the computational cost of particle–particle interaction evaluation is

$$C_{\mathrm{LPT}} = \alpha_1 s_5 N_p \log N_p + \alpha_2 (s_3 + s_4)\, N_p[(N_L(p)+1)^3 - 1], \tag{9}$$

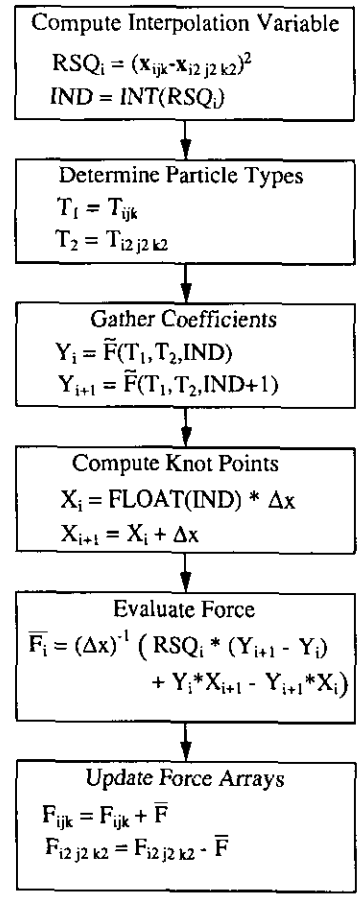
where s_5 is the time required to sort one element and s_4 is the time required to gather particle positions and update the partial forces. Again, factors α_1 and α_2 are the number of times a task must be performed during a simulation divided by the number of time steps in the simulation resulting in the expected number of tasks performed per step. Unlike the Eulerian case, it can be shown [9] that inflow/outflow boundary conditions, even in fluid volume elements subjected to a global shearing force, have little effect on the value of α_2. This is due to the inherent structure of the mapping of Lagrangian coordinates onto computer memory and is discussed below. Note that the cost of accessing the positions of close neighbors is not included since the information in the EPT neighbors list (NABERS in Flowchart 1) is inherent in the LPT indices ($i2$, $j2$, and $k2$ in Flowchart 2).

C. *Table Look-up for Function Evaluation*

Function evaluation via table look-up is a well established method in many different areas of computational physics where the weighting of a grid point or the evaluation of a matrix element requires many operations. Notable among these methods is the use of table look-up in conjunction with fast Fourier transforms [10], for the calculation of equation-of-state parameters in computational fluid dynamics models [11] and for the calculation of complex interactions in molecular dynamics simulations [12, 13]. Table look-up procedures usually consist of two stages. In the initialization stage, the function is evaluated at a set of values of the independent variable called tie points and the tables necessary for interpolation are generated. This stage does not necessarily entail the evaluation of complex analytic functions. For some applications, e.g., equation-of-state calculations, the original representation may be empirical data in a table. The next stage, which is the one used repeatedly, consists of a procedure for efficiently accessing specific values in the tables and computing the interpolant. For a given interpolant, the computational cost of table look-up methods is independent of the complexity of the original function.

As the potential energy functions used become more complicated or the number of particle types increases, the method of table look-up combined with interpolation procedures can be used to avoid the increased computational cost of the direct evaluation of potential energies and forces. Several table look-up procedures for using table look-up in molecular dynamics have been developed [12, 13]. The problem of the accuracy of interpolants has also been addressed and has resulted in a variety of interpolation schemes [14]. For molecular dynamics simulations in which particle–particle interactions are evaluated using table look-up, the analysis of the computational cost, Eq. (1), is independent of the interaction types and the number of particle types being modelled. Rather, it depends on the type of interpolation scheme employed. One common scheme is piece-wise polynomial interpolation. In this case, the computational cost of function evaluation, s_3, increases linearly with the order of the polynomial; however, the maximum error in the interpolant, $\varepsilon_{\max}$, is proportional to the magnitude of the spacing between the tie points, Δx raised to the power m [14]. Thus, the cost of function evaluation by an interpolant of order m is

$$s_3(m) \approx m s_3(1) \tag{10}$$



FLOWCHART 3. General programming structure associated with table look-up and Lagrangian tracking.

while

$$\varepsilon_{\max} \approx |\Delta x|^m. \tag{11}$$

The number of coefficients required for a given $\varepsilon_{\max}$ and set of polynomials, e.g., the Hermite polynomials, scales linearly with m while the number of tie points, $N_\tau(m)$, is

$$N_\tau(m) \propto 1/\sqrt[m]{\varepsilon_{\max}}. \tag{12}$$

For example, given $\varepsilon_{\max} = 10^{-6}$, the computational cost of linear interpolation using Hermite polynomials is approximately one-third that of using cubic Hermite polynomials while over 3000 times the number of coefficients are required. The general form of function evaluation by linear interpolation is depicted in Flowchart 3.

III. CRITERIA FOR THE EFFICIENT USE OF LAGRANGIAN PARTICLE TRACKING

For a given method of function evaluation, the computational cost of molecular dynamics simulations employing LPT depends on the method of implementation. To be specific, the two factors which negatively impact performance are the number of extraneous function evaluations and sorts of the Lagrangian coordinate system performed. In this section, we present criteria for minimizing both.

A. *Minimizing Unnecessary Function Evaluations in LPT*

It follows from Definition 1 that any algorithm used in conjunction with Lagrangian tracking must satisfy the following conditions:

Condition 1. It is never necessary to access memory in order to determine a particles neighbors.

In other words, Lagrangian tracking uses direct addressing of particle attributes, rather than indirect addressing as do neighbors-list algorithms. This ensures that the number of memory accesses required to, and hence computational cost of, assembling the information necessary for a given calculation is a minimum. With respect to accessing information, Lagrangian tracking is optimal for both vector and scalar architectures; however, in order to use Lagrangian tracking efficiently it is necessary to minimize the number of particles whose interactions with the particle of interest are negligible. The following three conditions are necessary for developing general criteria for this minimization:

Condition 2. The interactions between particles are local in nature and are negligible at interparticle separations exceeding some finite distance.

In the case of EPT, this separation corresponds to R_c. In order to define the corresponding quantity for LPT, the following conditions are necessary.

Condition 3. Each particle in the system has a characteristic size.

That is, there exists some minimum interparticle separation which corresponds to the distance at which two particles will invariably deflect.

Condition 4. Particles may enter or leave the model system only through the boundaries, i.e., particles are neither created or destroyed.

Note that Condition 4 in no way limits the use of LPT to simulations in which molecules undergo chemical reactions. Rather, it requires that, if such dissociation is to be modelled, the constituents in the reaction, atoms or groups of atoms, be treated as separate entities.

It follows from Condition 2 that there exists some maximum separation, R_c, at which the influence of neighboring particles may be ignored. It follows from Condition 3 that there exists an average number density, ρ_n, of system particles which represents the average number of particles in a unit volume. Given ρ_n, we may define a characteristic volume, $V(\rho_n) = \rho_n^{-1}$, which may be identified with average size of a unit Lagrangian cell. It follows from Condition 2 that

$$N_L(p) \geq \frac{R_c}{V(\rho_n)^{1/3}} - 1. \tag{13}$$

That is, the Lagrangian cutoff must be at least as large as the number of unit cell lengths corresponding to R_c.

It follows from these conditions that Lagrangian tracking is well posed for many different types of molecular dynamics calculations [1, 9]; i.e., many molecular dynamics simulations satisfy the above conditions. The task, then, is to appropriately choose $N_L(p)$ such that (13) is satisfied while $N_{\text{null}}(p)$, the number of superfluous operations, is minimized. This corresponds to the optima of

$$\min \tau^{-1} \int_0^\tau N_{\text{null}}(p)\, d\tau$$

s.t.

$$\tau^{-1} \int_0^\tau \left\| \sum_{i \in \{N_L(p)\}} f(p,i) - \sum_{j \in \{r_j \leq R_c\}} f(p,j) \right\| d\tau \leq \varepsilon_{\max}, \tag{14}$$

where $\varepsilon_{\max}$ is the largest acceptable error in the force or potential energy evaluation due to the Eulerian cutoff distance and $f(p, i)$ is the force on particle p due to particle i. It follows from the Conditions 3 and 4 that the spatial separation of particles is a continuous function of time; therefore, the time average rather than the instanteous values may be used.

Clearly, the optimization problem (14) is intractable, given the standard methods of optimization. It serves, rather, as a formal problem statement from which criteria for minimizing extraneous function evaluations based on system type may be established. We will consider the optimization of (14) in the case of a specific type of molecular dynamics simulation and then we comment on the case of general molecular dynamics simulations.

In the case of a molecular dynamics simulation of a fluid volume element, equating $V(\rho_n)^{1/3}$ with the average radial interparticle separation results in an overestimate of the actual separation. If we consider the neighbors of the particle of interest to be arranged in "shells" around it, e.g., the shell of first closest neighbors contains 26 particles, then in the case of the nth, as n approaches infinity, the overestimate of the average separation approaches $2^{1/2}$. Thus, using

$$N_L(p) = \frac{R_c}{V(\rho_n)^{1/3}} \tag{15}$$

is generally appropriate in this case. In the general case, the choice of $N_L(p)$ is more difficult. For the general equilibrium molecular dynamic simulation, the kinetic energy may be monitored and, if large fluctuations occur, $N_L(p)$ may be increased by one. In the general non-equilibrium case, this is a poor criterion. If the model system is a crystal, where phase transitions occur at specific temperatures, $N_L(p)$ may be estimated from the unit cell for a given configuration, e.g., fcc, and adjusted as a function of system temperature. In the absence of any specific information concerning the model system, it is best to start with a conservative estimate of $N_L(p)$ and perform identical simulations using various values. If the results of the simulations, i.e., the final particle positions and velocities, are identical to within the tolerance of the machine, then the lower value of $N_L(p)$ is acceptable.

Finally, we may reduce the number of extraneous function evaluations by adjusting the Lagrangian cutoff distance to take into account the spherical nature of the neighbors shells. In effect, rather than circumscribing the sphere with a cube, i.e., allowing the Lagrangian coordinates i, j, and k to vary from one to $N_L(p)$, we restrict their range of the coordinates with the following constraint:

$$(i+j+k)^2 \leqslant N_L(p)^2. \tag{16}$$

This constraint reduces the number of function evaluations from $N_L(p)^3$ to $\pi/6 N_L(p)^3$. For properly chosen $N_L(p)$, this number will be very close to the number of particles tracked in an EPT scheme.

B. *Minimizing the Number of Sorts Required*

In previous reports, many of the properties of what is defined in this paper as Lagrangian tracking are referred to as properties of particle tracking using a "monotonic Lagrangian grid" [1, 3] or a "monotonic logical grid" [2], i.e., MLG. We have chosen the more general designation, Lagrangian tracking, since the designation MLG implies certain limitations on the flexibility of these types of implicit pointer schemes. A strictly monotonic Lagrangian grid would correspond to the special case where the fixed integral difference defined in Eq. (5) have a value of one, i.e., $\Delta i = \Delta j = \Delta k = 1$. This condition is not a general requirement for maintaining the implicit pointers that are the basis of Lagrangian particle tracking. Lagrangian tracking establishes a correspondence between the "proximity" of particle locations in space and locations in computer memory, i.e., their Lagrangian coordinates. Establishing an exact correspondence between the relative values of spatial coordinates and locations in computer memory as is implied by a strict monotonicity condition is not a necessary condition for effecting Lagrangian tracking. The MLG requires monotonicity conditions solely for effecting this proximity correspondence; however, Lagrangian tracking schemes can function with a significant degree of local non-monotonicity in spatial coordinates and maintain substantial correspondence between the proximity of spatial and memory locations. For example, if the Lagrangian coordinates of particles 7 and 20 were interchanged (see Fig. 3), the Lagrangian grid would no longer be monotonic with respect to the x coordinate (see Fig. 1); however, both particles are still within the Lagrangian cutoff distance and the calculation of interparticle interactions remains unchanged. Therefore, satisfying monotonicity requirements, or any ordering requirement on spatial coordinates at every timestep, is not a necessary condition for Lagrangian tracking. It is, however, necessary to maintain a high degree of correspondence between the proximity of spatial and Lagrangian coordinates. Finally, since it is not necessary to apply ordering operations at every timestep for the purpose of constructing the implicit pointers used with Lagrangian tracking, Lagrangian tracking algorithms afford more flexibility with respect to increasing the computational efficiency of tracking particles in an evolving system.

To order the Lagrangian coordinate system, we introduce the concept of swapping neighboring particles in a particular direction. For example, an x swap would be one in which the information concerning particles at Lagrangian coordinates (i, j, k) and $(i+1, j, k)$ is exchanged. Since this is a lock-step procedure, i.e., each pair of particles is examined individually and exchanged if necessary, the actual number of swaps depends on the ordering of the system. In general, the order of the coordinate system is maintained at a fairly high degree; therefore, the number of swaps will be small compared to the number of particles in the system, N_p. In specific, if we define ν_s to be the frequency with which the Lagrangian coordinate system is sorted and $\mu_s(\nu_s)$ to be the average number of swaps per timestep for a

given v_s, it is generally true the μ_s is independent of v_s. This is due to the inherent ordering of Lagrangian coordinates. If a Lagrangian coordinate system becomes disordered in one dimension, the ordering in the other two dimensions may or may not be effected; therefore, the computational complexity of the swapping of Lagrangian coordinates is

$$o_1(LPT) \approx \max(N_x, N_y, N_z) \tag{17}$$

which is significantly less than the $N_p \log N_p$ necessary with EPT. Further, since each comparison is done in a lock-step pair-wise fashion, each sort requires only $3N_p$ comparisons and no computation as opposed to $N_p \log N_p$ comparisons and computations of the interparticle separation in the case of EPT.

IV. THE SIGNIFICANCE OF POINTERS AS LAGRANGIAN ATTRIBUTES

In the previous section, we presented the general structure of Lagrangian tracking algorithms and showed that they were highly efficient for particle tracking in simulations where "non-bonded" interactions are being computed. Non-bonded interactions are interactions where interacting particles are identified by the proximity of the particles. Thus, non-bonded interactions are independent of a particles identity, if not particle type; however, in simulations of some physical systems, there exist interactions where the knowledge of a particle's identity is necessary. For example, time dependent interactions, such as covalent interactions, or systems of linked particles, such as long chain molecules, require that the interacting particles be specified uniquely. These interactions represent "bonded" interactions since one particle is associated uniquely with another. In the case of bonded interactions, there is no increase in the efficiency of accessing this type of information when Lagrangian tracking is employed; however, the efficiency of the programming structures necessary for these interactions is not degraded when they are implemented in conjunction with Lagrangian tracking. This is because there is no need to impose an additional global data structure on the model system. In other words, the programming structures for bonded interactions may be embedded in a Lagrangian tracking scheme.

The programming structures for bonded interactions are inherently based on neighbors or linked list indexing. Since the identities of the particles interacting with a given particle through bonded interactions are unique, each particle must have a list of particles which are bound or linked to it. This corresponds to a list of pointers to these particles; however, since there is a one-to-one correspondence between the set of particles in the system and the set of lists, each list may be uniquely accessed via a particle's Lagrangian coordinates. It follows that a list of pointers is simply another Lagrangian attribute and, as it is accessed with a unique PID, does not require swapping. In the case of a static list, it is uniquely identified by a pointer, i.e., the PID, which is swapped. Further, the generation of dynamic lists, e.g., lists of hydrogen bonded neighbors, may also be treated in this manner; therefore, their generation will not degrade the performance of LPT.

In the preceeding sections, we have discussed both list and Lagrangian tracking algorithms. We have indicated that Lagrangian tracking constitutes a highly efficient programming structure for non-bonded interactions. This type of interaction accounts for the major portion of the computational cost of force evaluation in molecular dynamics simulations; however, there are interactions where list indexing algorithms are necessary in order to uniquely specify the identity of the interacting particles. The programming structures necessary for this type of indexing can be embedded into Lagrangian tracking schemes. Therefore, it is the task of the researcher designing the code to integrate the proper structures into an overall scheme which is both efficient and functional.

V. COMPUTATIONAL RESULTS

In this section, we outline the computational environment of our test simulations and present their results. We first present general results of test codes based on the programming structures outlined in the flowcharts. Next, we present the results of a specific molecular dynamics simulation for the purpose of illustrating the efficiency and flexibility of LPT.

Table I lists the computational cost in microseconds of each of the tasks required for LPT and EPT. Both the LPT and EPT codes were written in ANSI standard FORTRAN. For results produced on Cray machines, the codes were compiled and linked using the cf77 version 5.0 compiling system. The default optimization (vectorization) was enabled. In the case of the SGI machine, the codes were compiled and linked using the f77 version 3.10 compiling system. Safe microcode (level 2) optimization was enabled. Comparing the computational costs of tasks which occur with the same frequency, the time to perform an EPT sort, s_1, is approximately half that to perform an LPT swap, s_5;

TABLE I

Computational Costs, s_i, in Microseconds

Machine type	s_1	s_2	s_3	s_4	s_5
Cray YMP	5.81	2.40	0.50	0.77	11.0
Cray YMP-EL	27.7	11.0	1.74	4.04	47.3
Cray XMP	7.51	2.58	0.80	0.95	16.4
SGI R4000	17.1	1.60	3.42	1.05	9.04

TABLE II

$\mu_s(v_s)$ vs v_s

v_s	$\mu(v_s)$
0.05	7.47
0.1	7.75
0.2	7.62
1.0	7.49

however, the time to access particle positions and update partial forces in the EPT case, s_2, is twice that of the LPT case, s_4. Since the number of sorts per timestep is much less than the number of particle–particle interaction evaluations per timestep, LPT is almost twice as fast as EPT in the general case.

Table II shows the average number of swaps per time step as a function of the frequency of the re-ordering of the Lagrangian coordinate system for a specific molecular dynamics simulation, 1000 Ar atoms in a box with inflow/outflow boundary conditions at 120K. Clearly, μ_s is independent of v_s. It should also be noted that the final system configuration was unaffected by the choice of partitioning and may be optimized for a particular computer architecture, e.g., a very small value could be selected in the case of message passing distributed memory architectures where communication between processors is very expensive. It should also be noted that the number of swaps per step is of order $N_p^{1/3}$. Thus the number of swaps per step scales as the cube root of the system size. In the same set of simulations, the number of particles tracked by EPT was computed and compared to the number tracked by LPT. For a given R_c, using (15) resulted in the same number of particles being tracked by both methods. Thus, for this class of simulation, LPT is as efficient as EPT with respect to the number of particles tracked.

VI. CONCLUSIONS

In this report, we have described some general features of Lagrangian tracking and table look-up and stated formally those properties which are important to the design of efficient programming structures. Further, we have provided a formal definition of Lagrangian tracking and the necessary conditions for efficient programming structure design. We have examined list indexing and shown that, in simulations where list indexing is required due to the inclusion of bonded interactions, it can be incorporated into a Lagrangian tracking scheme without either imposing a separate global indexing scheme or effecting the efficiency of Lagrangian tracking. Finally, we have outlined programming structures for Lagrangian tracking, Eulerian tracking and table look-up in the form of flowcharts. These flowcharts represent efficient programming structures for Lagrangian tracking and table look-up in the context of vector computer architectures.

In the general case, we have shown that EPT requires almost twice the execution time of LPT. We have also shown that LPT is highly flexible with respect to task partitioning and that, in a specific case, it tracked the same number of particles as EPT.

ACKNOWLEDGMENTS

The authors extend special thanks to Gary Jones (DARPA Advanced Submarine Technology (AST) COTR for Hydrodynamics and Ship Control) and K. J. Moore and J. V. Dugan of Cortana for their interest and support. This work was performed under Contract MDA 972-88-C-0064.

REFERENCES

1. S. G. Lambrakos, J. P. Boris, R. H. Guirguis, M. Page, and E. S. Oran, *J. Chem. Phys.* **90**, 4473 (1989).
2. J. M. Picone, S. G. Lambrakos, and J. P. Boris, *SIAM J. Sci. Stat. Comput.* **11** (2), 368 (1990).
3. S. G. Lambrakos and J. P. Boris, *J. Comput. Phys.* **73**, 183 (1987).
4. R. W. Hockney and J. W. Eastwood, *Computer Simulation Using Particles* (McGraw–Hill, New York, 1981).
5. J. W. Eastwood, *Computational Methods in Classical and Quantum Physics* (Advance, London, 1976).
6. J. W. Eastwood, R. W. Hockney, and D. N. Lawrence, *Comput. Phys. Commun.* **19**, 215 (1980).
7. J. R. Beeler, Jr., *Computer Simulation in Material Science* (Carnes, 1988).
8. J. R. Beeler, Jr., *Radiation Effects Computer Experiments* (North-Holland, Amsterdam, 1983).
9. S. G. Lambrakos, J. H. Dunn, P. G. Moore, and W. C. Sandberg, *J. Chem. Phys.*, in press.
10. O. Buneman, *SIAM J. Sci. Stat. Comput.* 7 (2), 624 (1986).
11. P. J. Roache, *Computational Fluid Dynamics* (Hermosa, Albuguergue, NM, 1982).
12. J. A. Barker, R. A. Fisher, and R. O. Watts, *Mol. Phys.* **21**, 657 (1971).
13. T. A. Andrea, W. C. Swope, and H. C. Anderson, *J. Chem. Phys.* **79**, 4576 (1983).
14. C. de Boor, *A Practical Guide to Splines* (Springer-Verlag, New York/Berlin, 1978).