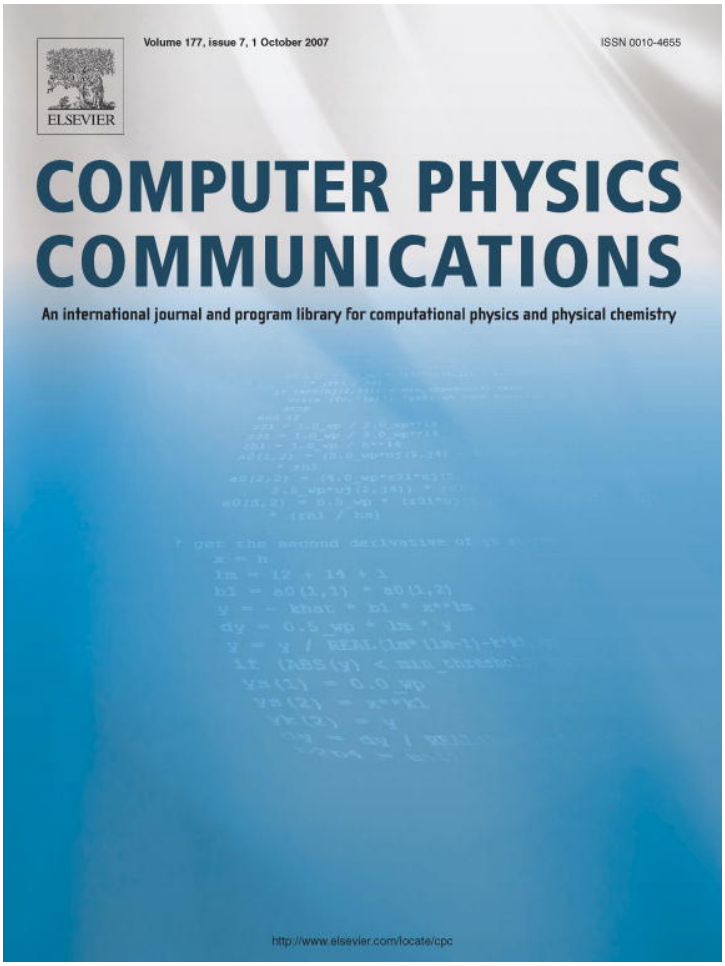


Available online at www.sciencedirect.com
ScienceDirect
Computer Physics Communications 177 (2007) 551–559
Computer Physics Communications
www.elsevier.com/locate/cpc

Algorithm optimization in molecular dynamics simulation

Di-Bao Wang [a,*], Fei-Bin Hsiao [a], Cheng-Hsin Chuang [b], Yung-Chun Lee [c]

[a] *Institute of Aeronautics and Astronautics, National Cheng Kung University, 70101 Tainan, Taiwan*
[b] *Department of Mechanical Engineering, Southern Taiwan University of Technology, 71005 Tainan, Taiwan*
[c] *Department of Mechanical Engineering, National Cheng Kung University, 70101 Tainan, Taiwan*

Received 12 February 2007; received in revised form 16 May 2007; accepted 21 May 2007

Available online 5 June 2007

Abstract

Establishing the neighbor list to efficiently calculate the inter-atomic forces consumes the majority of computation time in molecular dynamics (MD) simulation. Several algorithms have been proposed to improve the computation efficiency for short-range interaction in recent years, although an optimized numerical algorithm has not been provided. Based on a rigorous definition of Verlet radius with respect to temperature and list-updating interval in MD simulation, this paper has successfully developed an estimation formula of the computation time for each MD algorithm calculation so as to find an optimized performance for each algorithm. With the formula proposed here, the best algorithm can be chosen based on different total number of atoms, system average density and system average temperature for the MD simulation. It has been shown that the Verlet Cell-linked List (VCL) algorithm is better than other algorithms for a system with a large number of atoms. Furthermore, a generalized VCL algorithm optimized with a list-updating interval and cell-dividing number is analyzed and has been verified to reduce the computation time by 30 $\sim$ 60% in a MD simulation for a two-dimensional lattice system. Due to similarity, the analysis in this study can be extended to other many-particle systems.

Keywords: Molecular dynamics; Neighbor list; Algorithm; Optimization

1. Introduction

Molecular dynamics (MD) simulation has been developed for more than half a century. MD is a very useful tool in studying the momentum exchange and energy transfer in submicron scales or even in atomic levels. Specifically, MD is usually used to investigate the thermodynamics during an equilibrium process, to estimate the transport coefficients during a non-equilibrium process and even to provide information on atomic scales in a multi-scale simulation. MD simulation thus plays an important role in nano science and nano technology.

No matter what algorithm is utilized or what physical quantity is going to be derived, the typical flow chart of a MD simulation can be illustrated in Fig. 1. The post processing here means the calculation of system energy (potential and kinetic energy) and transport coefficients through displacement and velocity. With regard to the computational cost at each time-step, the identification of neighboring atoms and the calculation of inter-atomic distance/force require the majority of CPU time. Usually there are two ways to accelerate the computation of MD. One is to use and optimize parallel computation [1,2] and the other is to improve the MD algorithm implemented on a single machine [4–10], and it is the latter method that this study will concentrate on.

MD computation acceleration can be achieved either through choosing the most suitable algorithm or optimizing the numerical parameters for each algorithm. Frenkel [3] provided some empirical criteria for choosing a certain kind of algorithm for a certain class of MD problems. However, they are only qualitative instead of quantitative and thus hard to implement in real MD simulations.

With considering the long-range forces, some early effort was made to obtain a computational cost of O(N) [4]. Since the

* Corresponding author at: Institute of Aeronautics and Astronautics, National Cheng Kung University, No. 1, Dasyue Rd., Tainan 70101, Taiwan. Tel.: +886 6 2757575 63642; fax: +886 6 2088214.
E-mail addresses: dibao.wnag@gmail.com (D.-B. Wang), fbhsiao@mail.ncku.edu.tw (F.-B. Hsiao), chchuang@mail.stut.edu.tw (C.-H. Chuang), yunglee@mail.ncku.edu.tw (Y.-C. Lee).

0010-4655/$ – see front matter
doi:10.1016/j.cpc.2007.05.009

Nomenclature

C_d	cell-dividing number
E_0	equilibrium potential energy of system atoms
k	list-updating interval
k_B	Boltzmann constant
m_0	the mass of atom
N	total number of atoms
N'	average number of atoms inside a cube with edge-length of R_V
N''	average number of atoms inside a cube with edge-length of R_C
N'_c	defined as $(27N' - 1)$
N'_s	defined as $(\frac{4\pi}{3}N' - 1)$
R_C	cut-off radius
R_V	Verlet radius
T	the system mean temperature
t_0	the characteristic time of system atoms
Δt	increment of time integration
$V_{\max}$	the maximum speed of atoms
$\langle V \rangle$	mean speed of system atoms

Greek symbols

α	reduction factor of computation time
τ_c	time to calculate the indices of cell
τ_f	time for force calculation
τ_h	time to check whether if a cell is empty or not
τ_j	time to judge if an atom stays inside the Verlet radius
τ_i	time for integration
τ_L	the time to establish the list array
τ_{MD}	the total computation time at each time-step
τ_n	time to identify neighboring atoms
τ_r	the time for distance calculation
τ_s	time for summation
$\bar{\rho}$	average number density of atoms

Abbreviations

CL	cell-linked list
FC	full computation
GVCL	generalized Verlet cell-linked list
MD	molecular dynamics
VL	Verlet list
VCL	Verlet cell-linked list

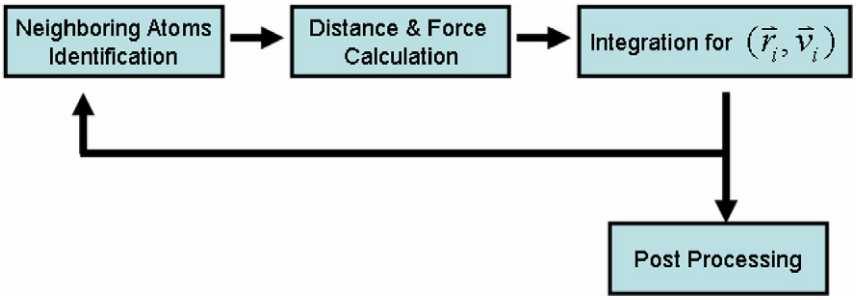


Fig. 1. Typical computation flow chart of MD simulation.

absolute value of inter-atomic potential or inter-atomic force decays very fast to zero when the inter-atomic distance gets larger than several times the equilibrium distance, a cut-off radius, R_c (typically about three or four times of the equilibrium distance) is usually defined so that the neighboring atoms outside this radius will be ignored. Using this cut-off radius, the interaction among atoms is actually reduced to a short-range type. Algorithms about the efficient identification of the neighboring atoms inside the cut-off radius have been proposed for a long time, with examples, such as Verlet list, cell-linked list and their combination [3]. In recent years, some improved MD algorithms also appear to provide more efficient ways to identify the neighboring atoms [6–10], but studies of these are either restricted to a specific system (either with periodic boundaries or for slowly-moving particles) or lack both analytic evaluation and prediction of the efficiency. Sutmann and Stegailov [9] analytically discussed the optimization of neighbor list technique for Verlet list and cell-linked list separately instead of in combination. However, it is known that the combination of these two algorithms performs much better than each one alone under many physical conditions. Yao et al. [7] introduced some data processing techniques to optimize the hybrid algorithm of the Verlet list and cell-linked list, but no analytical estimation of MD computation time has been proposed, and hence the optimization is somewhat restricted.

This paper first defines the Verlet radius and develops an approximate formula to estimate the computation time for each MD algorithm evaluated. The effects of total number of atoms, system density, system temperature and list-updating interval will be addressed for comparison. Based on the current Verlet cell-linked list (VCL) algorithm, a generalized version is proposed so that it can further reduce the computation time by 40% at best for very large atomic systems. The effect of physical conditions on the proposed algorithm will also be investigated in detail.

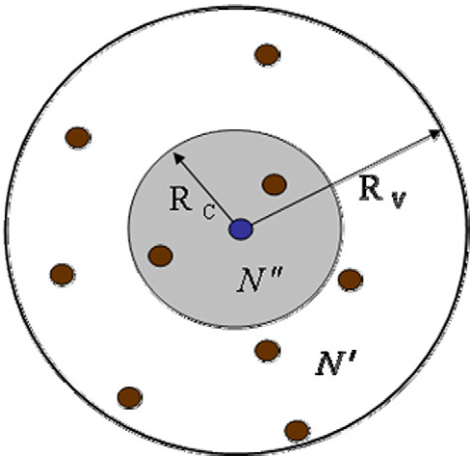


Fig. 2. Illustration of the cut-off radius R_C and Verlet radius R_V as well as the corresponding number densities of atoms N'' and N'.

2. Verlet radius

Let's first define R_C and $\bar{\rho}$ to be the cut-off radius (usually three or four times the equilibrium distance of inter-atomic potential) and the average number density of atoms in a system, respectively, which is defined as

$$N'' \equiv R_C^3 \bar{\rho}, \tag{1}$$

where N'' is the average number of atoms inside a cube with edge-length of R_C. In order to establish the list in recording the neighboring atoms around each atom more efficiently during the MD simulation process, a commonly applied skill is to use a large radius called, the Verlet radius, R_V, in which the atoms outside the sphere with radius R_V are prevented from penetrating the sphere with radius R_C in the following k time-steps (k is defined as the list-updating interval). The concept is illustrated in Fig. 2, where the sphere with radius R_V is often called the interaction sphere [11]. This list will be updated every k time-steps. Obviously, the value of R_V depends on the selection of k, but their relation has not been well discussed in th published literature to date. The relation between the two will be proposed in the present study.

As shown in Fig. 2, if the list recording the neighboring atoms is established by R_V and will be repeatedly used in the following k time-steps, then R_V would be defined so that after $(k-1)\Delta t$ steps some neighboring atoms, initially located on the sphere with radius R_V, which may penetrate the sphere with radius R_C, can be contained by such a list in the beginning. That is,

$$R_V \equiv R_C + (k-1)V_{\max}\Delta t, \quad k \in \mathbb{N}, \tag{2}$$

where the maximum speed, $V_{\max}$, is considered to cover the extreme situations. For practical applications, it is suggested that $V_{\max}$ be estimated by one hundred times the mean speed derived in statistical mechanics for gas, and Δt is estimated by one thousandth of the characteristic time, t_0, of the system atoms. Since the characteristic time of atoms is about the order of magnitude of $R_C\sqrt{m_0/E_0}$, thus $V_{\max}$ and t_0 can be written as

$$V_{\max} \approx \langle V \rangle \times 10^2 = \sqrt{\frac{8k_BT}{\pi m_0}} \times 10^2, \tag{3}$$

$$\Delta t = t_0 \times 10^{-3} \approx R_C\sqrt{\frac{m_0}{E_0}} \times 10^{-3}, \tag{4}$$

where $\langle V \rangle$ is the mean speed of system atoms, k_B ($= 1.38 \times 10^{-23}$ J/K) the Boltzmann constant, T (K) the system mean temperature, m_0 (kg) the mass of atom, and E_0 the equilibrium potential energy of atoms in the system. Finally the average number of atoms inside a cube with edge-length of the Verlet radius, N', can be estimated as

$$N' \equiv R_V^3\bar{\rho} \approx \left[1 + (k-1) \times 10^{-1} \times \sqrt{\frac{8k_BT}{\pi E_0}}\right]^3 N''. \tag{5}$$

In most real cases, E_0 is around 10^{-20} (J), thus N' could be estimated as $[1 + (k-1) \times 10^{-2}\sqrt{T}\,]^3 N''$ and R_V as $R_C[1 + (k-1) \times 10^{-2}\sqrt{T}\,]$.

3. Computation time of current MD algorithms

In general, the total computation time required at each time-step in MD simulation is composed of the time to identify the neighboring atoms, the time to calculate inter-atomic distance as well as inter-atomic force, the time to sum up the force, and the time to perform time-integration. Thus for an entire MD simulation, the total computation time τ_{MD} at each time-step shall include

$$\tau_{\mathrm{MD}} = \tau_n + \alpha \times (\tau_r + \tau_f + \tau_s) + N \times \tau_i, \tag{6}$$

where N is the total number of atoms in a system, α the reduction factor, τ_{MD} the time consumption in MD simulation, τ_n the time to identify neighboring atoms, τ_r the time for distance calculation, τ_f the time for force calculation, τ_s the time for summation, and τ_i the time for integration. In Eq. (6), τ_r, τ_f, τ_s and τ_i are identical for each MD algorithm if the integration scheme is the same; however, τ_n and α will be changed in different MD algorithms, an issue which will be introduced in the following sections. Before mentioning each MD algorithm, other terms of time consumption are required in τ_n, such as the time to judge if an atom stays inside the Verlet radius (τ_j), the time to calculate the indices of cell (τ_c), the time to establish the list array (τ_L), and time to check if a cell is empty or not (τ_h). The overall corresponding numerical operation for individual time consumption is defined in Table 1, by which the estimation formula of computational time for the relevant MD algorithm can be formulated, and this will be discussed below. Here, note that the assumptions of each algorithm are made as follows:

(i) pair-wise interaction,
(ii) uniform distribution of system atoms,
(iii) the time consumption of each numerical operation independent of others.

Table 1
The time consumption and corresponding numerical operation defined in this study

Time consumption	Numerical operation
τ_r	$r_{ij} = [(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2]^{1/2}$
τ_f	$\begin{cases} f_{ij} = (A/r_{ij})^6 - (B/r_{ij})^{12} & \text{or} \\ f_{ij} = e^{-Ar_{ij}} - e^{-Br_{ij}} \end{cases}$
τ_s	$\sum f_{ij}$
τ_j	if $r_{ij} \leqslant R_V$ then O.K., else Skip
τ_c	$I_j = \lvert(x_j - x_{j,0})/\delta_{j,0}\rvert$
τ_L	$\begin{cases} List(n, m, l, 1) = List(n, m, l, 1) + 1, \\ List(n, m, l, i) = j \end{cases}$
τ_h	Check if $N_{ijk} \neq 0$

3.1. Full computation (FC)

If no reduction skill is utilized, τ_n is zero and α is C_2^N $(= N(N-1)/2)$, therefore, the total computation time for the Full Computation algorithm is

$$\tau_{\mathrm{MD,FC}} = C_2^N \times (\tau_r + \tau_f + \tau_s) + N \times \tau_i. \tag{7}$$

3.2. Verlet list (VL)

In the beginning of each list-updating interval, C_2^N operations are required to calculate the inter-atomic distance so as to judge if this distance is less than the Verlet radius. Finally, it requires NN_s' operations to establish the Verlet list, where N_s' is defined as $(\frac{4\pi}{3}N' - 1)$, since every atom only has to interact exclusively with the neighboring atoms inside the sphere with radius of R_V. Thus the time consumption at each time-step for the Verlet List algorithm will be

$$\tau_{\mathrm{MD,VL}} = \frac{1}{k}\left[C_2^N(\tau_r + \tau_j) + NN_s'\tau_L\right] + NN_s'(\tau_r + \tau_f + \tau_s) + N\tau_i. \tag{8}$$

3.3. Cell-linked list (CL)

In the beginning of each list-updating interval, it requires N operations to assign each atom to a certain cell and establish a list for each cell to record the index for each atom. Since some cells may contain no atoms, in each step for each atom, it has to check whether or not the neighboring cells around it are empty, which requires twenty seven operations in three-dimensional case. In addition, the computation is also spent on the interaction with the atoms in the neighboring cells. In average, the number of atoms to interact with a certain atom is N_c', which is calculated as $(27N' - 1)$. Therefore, the total computation time for the Cell-linked List algorithm is

$$\tau_{\mathrm{MD,CL}} = \frac{1}{k}\left[N(\tau_c + \tau_L) + N \times 27\tau_h + NN_c'\tau_L\right] + NN_c'(\tau_r + \tau_f + \tau_s) + N\tau_i. \tag{9}$$

Table 2
Physical CPU time for each numerical operation

	τ_f Morse	τ_f L-J	τ_c	τ_r	τ_L	τ_j
CPU time	1.0	0.25	0.36	0.29	0.08	0.01

Note that each time consumption is normalized by the Morse-typed force computation.

3.4. Verlet cell-linked list (VCL)

The Verlet-cell list (VCL) algorithm combines the advantages of Verlet list and cell-linked list. In the beginning of each list-updating interval, it requires N operations to assign each atom to a certain cell. It also requires NN_c' operations to calculate and judge the inter-atomic distance between the central atoms and neighboring atoms in surrounding cells and NN_s' operations to establish the neighbor list with neighboring atoms inside the interaction sphere. Using this algorithm, the reduction factor α is the same as in Verlet list, while τ_n is much smaller. The total computational time at each time-step for the Verlet-Cell List algorithm is

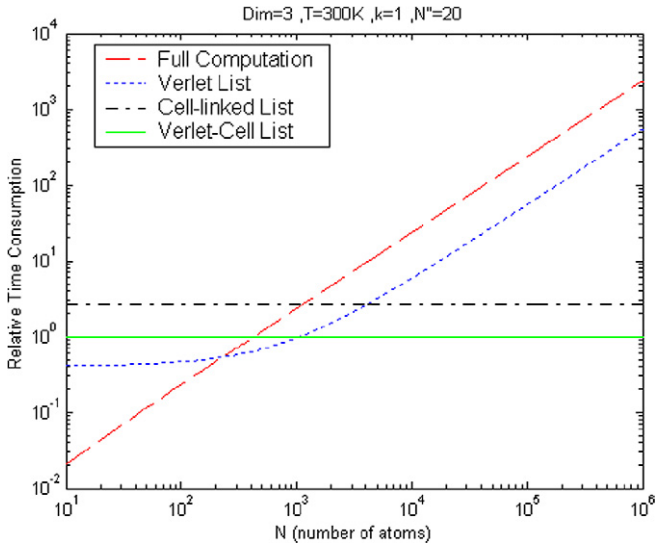
$$\tau_{\mathrm{MD,VCL}} = \frac{1}{k}\left[N\tau_c + NN_c'(\tau_r + \tau_j) + N \times 27\tau_h + NN_s'\tau_L\right] + NN_s'(\tau_r + \tau_f + \tau_s) + N\tau_i. \tag{10}$$

3.5. General comparison of current algorithms

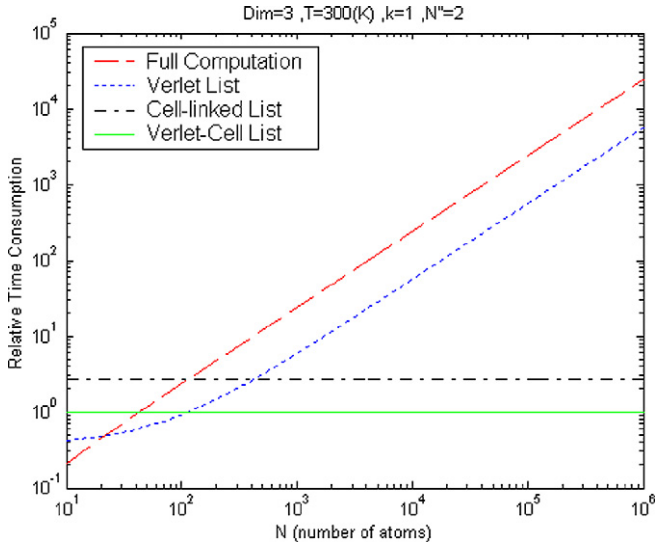
Before evaluating the computational time for all algorithms derived in the formulas given in Eqs. (7)–(10), the estimation of each numerical operation in shown in Table 1 has to be performed in advance. The estimation is realized by repeating the numerical operations for a large number of atoms and then the mean value is obtained. The normalized results are listed in Table 2. It has been found that the computed results here are independent of programming languages, program compilers and computing machines. Thus the data and its extended predictions can be universal.

If the list-updating interval, k, and system temperature, T, are fixed, the variation of the MD time consumption with respect to the total number of atoms is shown in Fig. 3. In order to make the comparison more clearly, among all algorithms the time consumption has been normalized with the one spent by the VCL algorithm for each value of N. Fig. 3 indicates that for dense systems with a small number of atoms, the full computation with no reduction process will be the fastest; while for systems with a large number of atoms, the VCL algorithm clearly advances over all other algorithms. As for the systems with an intermediate number of atoms, the VL algorithm will consume the least time. Equaling Eqs. (7) and (8) and Eqs. (8) and (10), the optimized algorithm on the N–N'' plane for the MD simulation can be categorized in the relevant domain shown in Fig. 4. According to this figure, the fastest MD algorithm can be chosen depending on N, the given number of atoms, and the average system density N''.

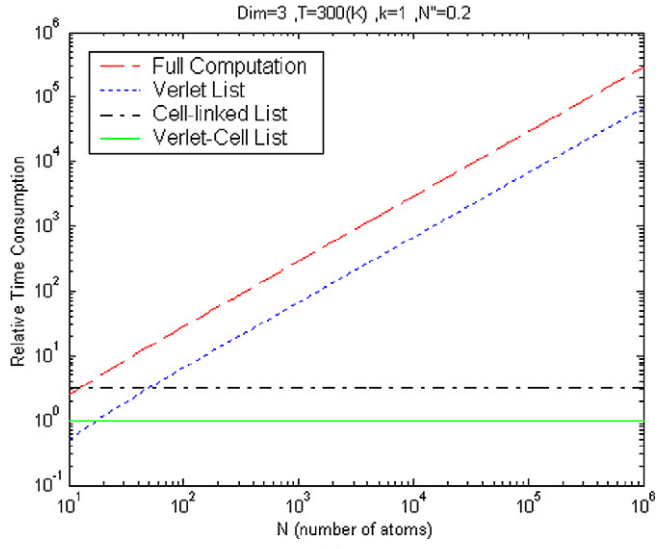
Next, the effect of the list-updating interval, k, is going to be discussed. Without deep investigation in finding the relationship between the list-updating interval and the Verlet radius, it



(a)



(b)



(c)

Fig. 3. Variation of MD time consumption with respect to total number of atoms for each algorithm. (a) $N'' = 20$, (b) $N'' = 2.0$, (c) $N'' = 0.2$. The time consumption for each algorithm is normalized by that of VCL at each value of N.

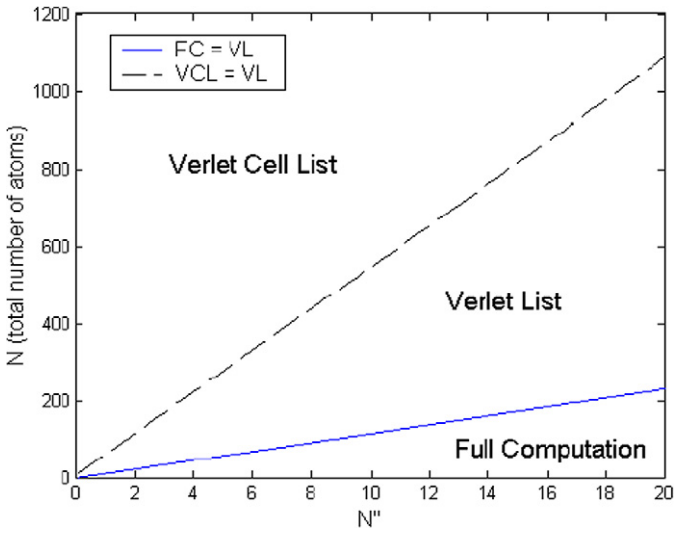


Fig. 4. Optimization territory for each current MD reduction algorithm in N–N'' plane ($k = 1$, $T = 300$ K).

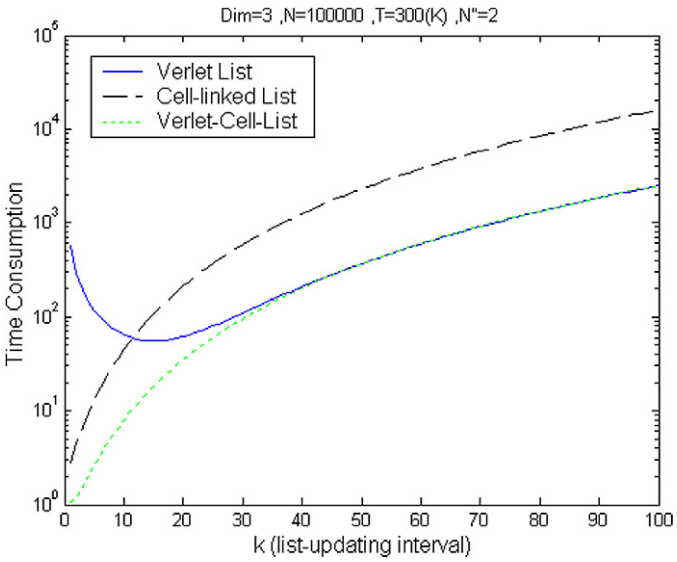


Fig. 5. Variation of MD time consumption with respect to list-updating interval for each algorithm. Here $N'' = 2.0$.

is often misunderstood that a large value of k always results in a less computation time to identify the neighboring atoms for each atom. Such a misunderstanding is mainly due to the impression of a fixed value of the Verlet radius. However, if Eq. (5) and Eqs. (8)–(10) are observed in detail, it will be found that the fact is not that simple and intuitive. Although the factor of $1/k$ in Eqs. (8)–(10) represents the decreasing effect for the time consumption in MD computation, the terms such as N_c' and N_s', represent the increasing effect. Therefore, it is not clear whether the increase of k will reduce τ_{MD} until the variation of τ_{MD} is examined. Since an analytical solution of k of equation $\partial\tau_{\mathrm{MD}}/\partial k = 0$ does not exit, it can only be examined in a graphic manner.

Fig. 5 illustrates the typical variation of MD computation time with respect to the list-updating interval for algorithms VL, CL and VCL. For the first glance at this figure, only the

VL algorithm has an optimized choice of k, where the time consumption reaches a minimum; whereas an increase in k causes a monotonic increase for both CL and VCL algorithms. Note that the variation of τ_{MD} or the existence of the minimum of τ_{MD} is influenced by the system average temperature and average density, and thus the effects of T and N'' should be investigated more carefully.

Since the system with a large number ($>10^3$) of atoms is of more practical interest, especially in nano-scale simulation, and the VCL algorithm advances the rest of the above mentioned algorithms in large systems, the following discussion focuses on the MD simulation results based on the VCL algorithm. Consider a system with a million atoms and the temperature T varying from 10 to 500 K under each value of N'', as shown in Fig. 6. Observing Fig. 6 (a) to (c), it is found that the existence of the minimum of τ_{MD} for the VCL algorithm occurs only at low temperatures. Furthermore, the decrease of τ_{MD} at an optimized value of k becomes more obvious for lower densities in the lower temperature range. What's more interesting is that the optimized values of k in each low temperature have nothing to do with the value of system densities, which can be seen in Fig. 6 (a) to (c). In comparison with the case of $k = 1$, the MD computation time can be reduced drastically from 10% up to 60% (depending upon the system temperature and density) with the optimized list-updating interval.

Thus it is concluded that the list-updating interval in the VCL algorithm should not be arbitrarily chosen and does not necessarily have an optimized value to minimize the computation time. Instead, the optimized list-updating interval only exists in the lower system temperature range and should be derived through Eq. (10) in a graphic manner.

4. Generalized Verlet-cell list

It has been shown that for a system with a large number of atoms, the VCL is currently the fastest algorithm and its efficiency can be optimized by selecting a suitable number of list-updating intervals, as mentioned in the previous section. Based on the VCL algorithm, its generalized version, called generalized Verlet cell list (GVCL), is introduced in the present study and is carefully discussed to demonstrate its improvement with higher computation efficiency in this section.

The fundamental concept of GVCL is to utilize a smaller edge-width of the cell, which will result in a smaller covering area to identify the neighboring atoms inside the interaction sphere. This kind of concept was mentioned before by other researchers [7,11], but the analytical estimation and corresponding optimization of the computational time have still not appeared in the published literature. Originally, the VL algorithm tries to identify the neighboring atoms which are possibly inside the interaction sphere, but now the VCL algorithm requires searching for all atoms within all neighboring cells (that is, 9 cells for two-dimensional case and 27 cells for three-dimensional case). As illustrated in Fig. 7, in order to identify the neighboring atoms inside the interaction sphere using algorithm VCL, all the distances between the central atom and the atoms located at the whole nine solid-lined cells have to be cal-

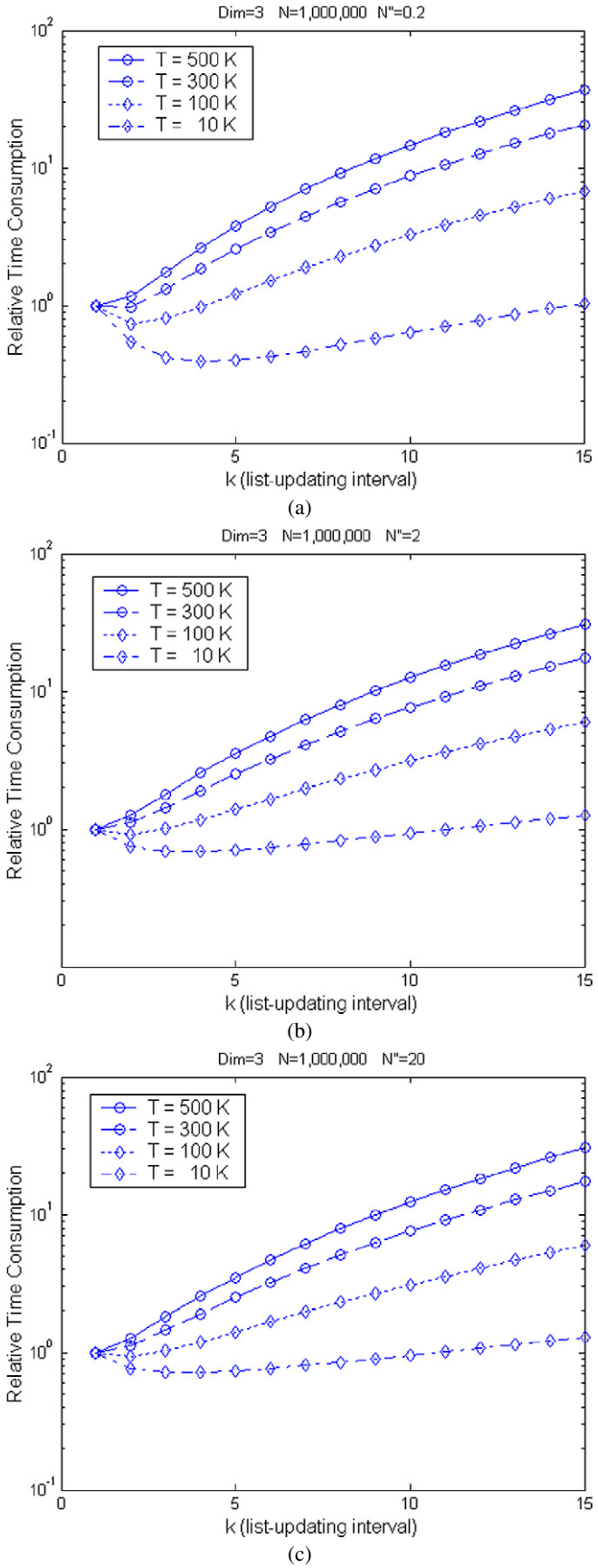


Fig. 6. Variation of MD time consumption with respect to list-updating interval for algorithm VCL. (a) $N'' = 0.2$, (b) $N'' = 2.0$, (c) $N'' = 20$. Here the time consumption has been normalized by that for case of $k = 1$.

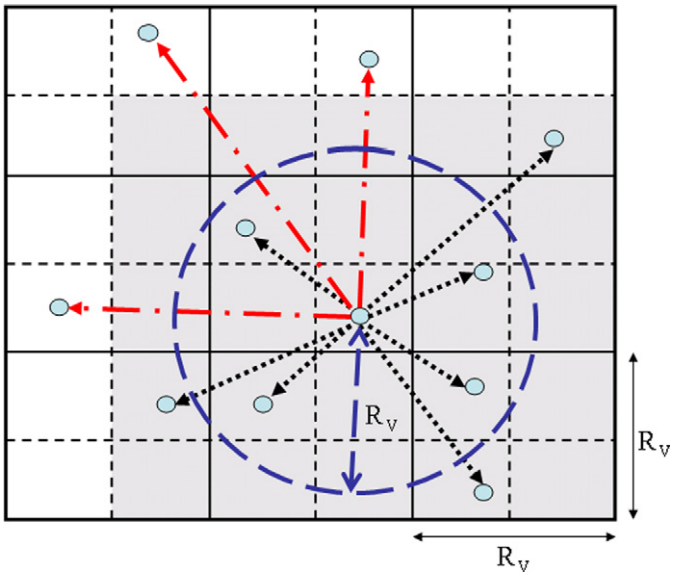


Fig. 7. Illustration for algorithm VCL and algorithm GVCL in two-dimensional cross-section. The shaded area represents the case of cell-dividing number equaling two.

culated, as indicated by the dashed-dotted-line arrows and the dotted-line arrows. However, if the edge-width of the cell is divided by two, then only the distances indicated by the dotted line have to be considered. On the other hand, the searching area is immediately shrunk to $(5/6)^2 = 25/36$ times compared to the original one. Here the cell-dividing number is defined as C_d. For $C_d = 1$ it returns back to the original VCL algorithm. As for using the GVCL algorithm, the value of $C_d \geqslant 2$. Ideally, when C_d increases to infinity, the searching area shrink to $4/9$ for two-dimensional cells and $8/27$ for three-dimensional cells.

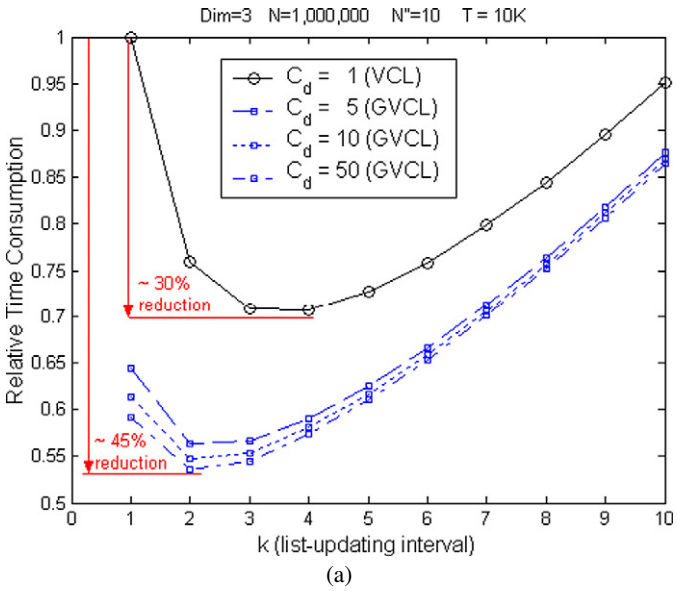
Although the number of atoms to be identified decreases as C_d increases, the time consumed to check whether a neighboring cell is free of atoms increases in a fashion of $(2C_d + 1)^3$. This implies that an optimized value of C_d exists such that the MD computational time can be minimized accordingly. Finally, the analytical estimation formula of the MD computational time using the GVCL algorithm be derived as,

$$\tau_{\mathrm{MD,GVCL}} = \frac{1}{k}\left[N\tau_c + N(2C_d+1)^3\tau_h + NN'_d(\tau_r+\tau_j) + NN'_s\tau_L\right] + NN'_s(\tau_r+\tau_f+\tau_s) + N\tau_i, \quad (11)$$

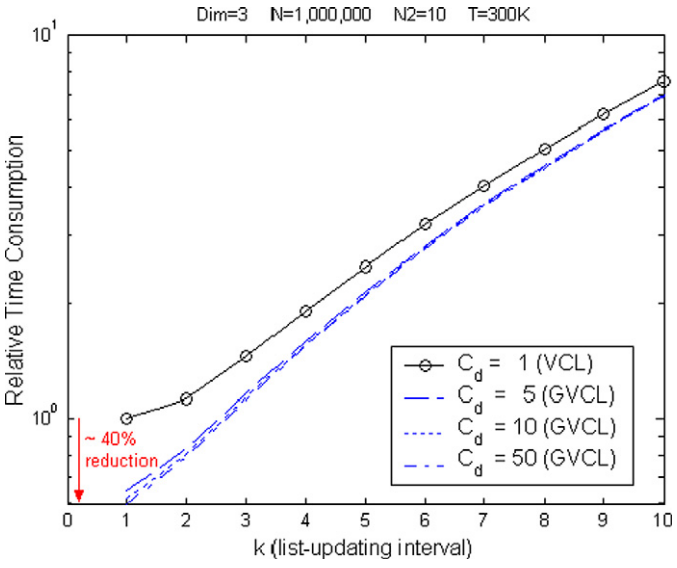
$$N'_d \equiv \left(\frac{2C_d+1}{C_d}\right)^3 N' - 1, \quad C_d \in \mathbb{N}, \quad (12)$$

where N'_d gives an average number of neighboring atoms to be considered in each cell of GVCL with a certain value of C_d.

Since the GVCL algorithm is extended from the VCL algorithm in the present study, its characteristics in the variation of MD computational time are similar to those of the VCL algorithm. From the MD simulation study here. When the average temperature of the system is low (<100 K), the reduction of computation time can be maximized (for $k = 2$) up to 15%



(a)



(b)

Fig. 8. Variation of MD time consumption with respect to neighbor-holding times using each value of cell-dividing times, C_d. (a) $T = 10$ K, (b) $T = 300$ K.

in comparison with the VCL algorithm (for $k = 4$) if the list-updating interval k and cell-dividing times C_d are carefully chosen, as illustrated in Fig. 8(a). Similar to the VCL algorithm, when the list-updating interval k increases, no reduction of MD computation time is obtained when the average temperature of the system is higher; thus keeping k equal to one is the best choice. With this selection of k, using the GVCL algorithm for the MD simulation can still lower the computation time down to 50% compared to what the VCL algorithm consumes. This is clearly illustrated in Fig. 8(b).

Although it seems that a larger value of C_d will result in a greater reduction of the MD computation time, the selection of the cell-dividing number by no means follows the concept of "the larger, the better". In order to obtain the minimum of the MD computational cost, in principle the optimized combination of (C_d, k) does exist and can be derived by solving

$\partial\tau_{MD,GVCL}/\partial k = 0$ and $\partial\tau_{MD,GVCL}/\partial C_d = 0$ simultaneously through Eq. (11), but the optimized values of (C_d, k) are usually solved by numerical schemes, such as the Newton–Raphson method, and may not be convenient to apply. Therefore, the practical way to determine the optimized (C_d, k) is carried out by conducting several combinations of testing through real MD simulation in a systematic manner. Fortunately, according to the testing experiences in the present study, the optimized (C_d, k) usually ranges from $(1, 1)$ to $(5, 5)$ in most cases tested with different temperatures and densities. Hence, before running for a large number of time-steps, a suggested procedure would be to run the real MD simulation for decades or thousands of time-steps with these twenty five combinations of (C_d, k) to find out the most favorable numerical parameters.

5. Numerical validation of GVCL algorithm

Before an algorithm comes to realization, its accuracy and efficiency first have to be verified. Since the number of neighboring atoms to interact with for the GVCL algorithm is the same as that for the VCL algorithm, the accuracy of MD simulation results such as displacement and velocity are identical and thus do not need to be inspected. However, only the theoretical prediction for the efficiency optimization has been provided so far, and the numerical validation through a real case is still required.

In the following, a case of two-dimensional wave propagation inside a hexagonal lattice system is going to be used as a testing problem. This lattice is composed of copper atoms using Morse-typed inter-atomic potential. Free boundary condition is applied to all boundary atoms. For atoms located around the central area, the initial condition is the thermal random motion combined with the assigned distribution of displacement. For other atoms, the initial condition is thermal random motion only. The total number of atoms is set to be ten thousand and the system temperature ranges from 10 to 300 K. This MD simulation was run on a personal computer with a 3 GHz Pentium dual CPU and 2.0 GB DDR RAM for one thousand time-steps.

The tested results are selectively shown in Fig. 9. It can be seen that there exists an optimized combination of list-updating interval and cell-dividing times for the system with lower temperature and higher temperature. In order to compare the computation reduction more conveniently, all results here are normalized by that of the VCL algorithm with the list-updating interval equal to unity; that is, normalized by the case with $(k, C_d) = (1, 1)$. If the numerical setting of (k, C_d) has been selected with optimization, the MD computation time here could be reduced by 25 to 40%, depending on the system temperature provided. This numerical testing therefore recognizes the excellent efficiency of the proposed GVCL algorithm and verifies the theoretical prediction of its performance when compared with Fig. 8.

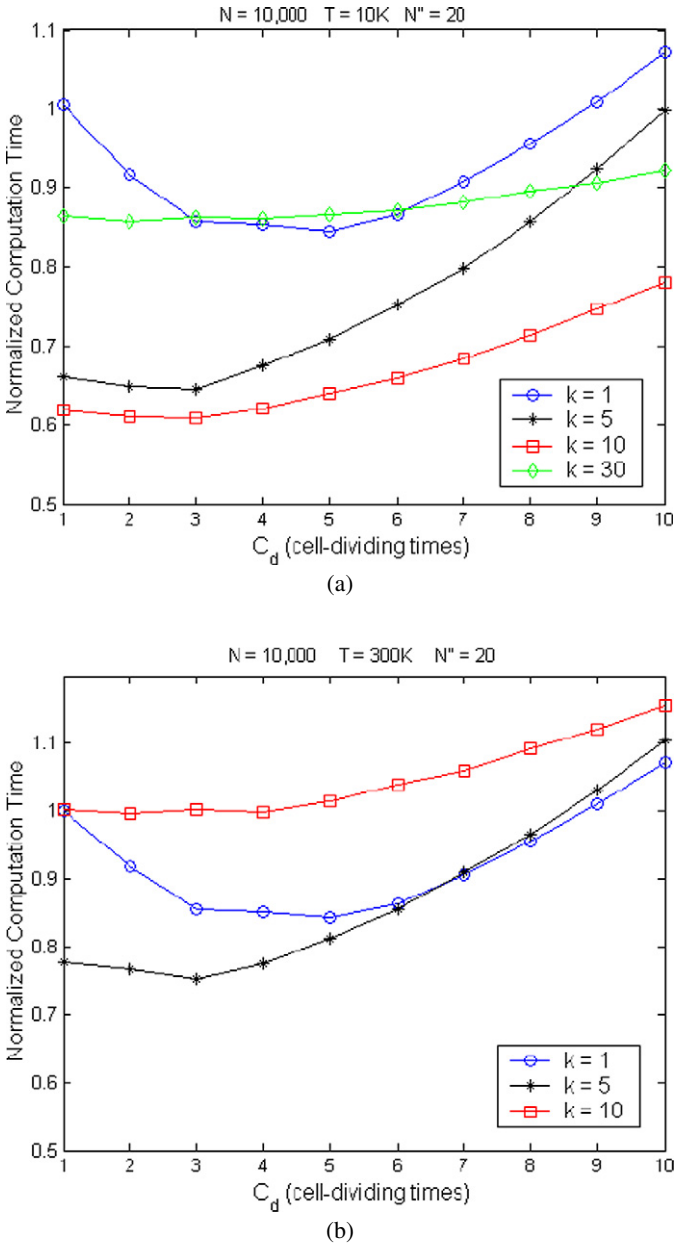


Fig. 9. Numerical validation for effects of neighbor-holding number and cell-dividing number. All results are normalized by the case with $(k, C_d) = (1, 1)$. (a) $T = 10$ K, (b) $T = 300$ K.

6. Conclusions

Starting from statistical mechanics, a rigorous definition of Verlet radius is formulated and estimated as a function of system average temperature and list-updating intervals. Based on this definition, the estimation formulas of Molecular Dynamics (MD) computational time for existing MD algorithms such as Full Computation (FC), Verlet list (VL), Verlet-cell link (VCL) and the proposed generalized Verlet cell-list (GVCL) algorithm are derived and discussed in the present study.

For algorithm selection, the FC algorithm is suitable for small and dense systems, while the VCL algorithm is better for large and rarefied systems, and the VL algorithm for intermediate systems.

For algorithm optimization for systems with a large number of atoms, it is concluded that the list-updating intervals

of VCL and GVCL algorithms should not be arbitrarily chosen and not necessarily have an optimized value to minimize the computation time. The optimized list-updating interval only exits in the lower temperature system and should be derived from Eqs. (10)–(11) in a graphic manner. Therefore, for systems around or above room temperature, establishing the list at each time-step would be the best choice.

As for the GVCL algorithm herein proposed, it is found that with the cell-dividing times optimized the MD computation cost could be further reduced by 15% in low-temperature systems and reduced by $30 \sim 60\%$ in high-temperature systems, compared with the VCL algorithm. With the successful verification by the real MD simulation, the value of the GVCL algorithm discussed in this study has been demonstrated.

Due to the similarities in computational procedures, the analysis in this study can be extended to other many-particle systems such as point-vortex method in computational fluid dynamics and celestial dynamics in astronomy to identify the neighboring particles or objects in a more efficient way.

Acknowledgements

This research was partially supported by the National Science Council of Taiwan through Grant NSC 95-2120-M-006-002.

References

[1] M. Mu, J. Comput. Phys. 179 (2002) 539.
[2] R. Murty, D. Okunbor, Parallel Comput. 25 (1999) 217.
[3] D. Frenkel, B. Smit, Understanding Molecular Simulation: From Algorithms to Applications, second ed., Academic Press, San Diego, 2002.
[4] R.W. Hockney, S.P. Goel, J.W. Eastwood, Chem. Phys. Lett. 21 (1973) 589.
[5] B. Quentrec, C. Brot, J. Comput. Phys. 13 (1973) 430.
[6] D.R. Mason, Comput. Phys. Comm. 170 (2005) 31.
[7] Z. Yao, J.-S. Wang, G.-R. Liu, M. Cheng, Comput. Phys. Comm. 161 (2004) 27.
[8] T.N. Heinz, P.H. Hunenberger, J. Comput. Chem. 25 (2004) 1474.
[9] G. Sutmann, V. Stegailov, J. Mol. Liq. 125 (2006) 197.
[10] T. Maximova, C. Keasar, J. Comput. Biol. 13 (2006) 1041.
[11] M.P. Allen, D.J. Tildesley, Computer Simulation of Liquids, Oxford Univ. Press, New York, 1990.