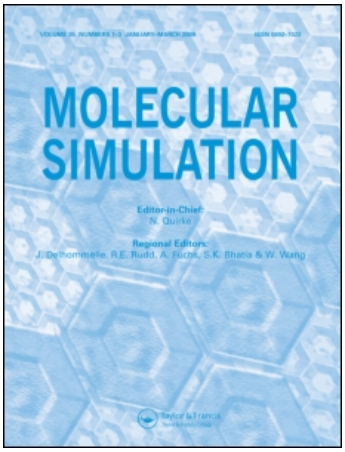


**Molecular Simulation**
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t713644482

# Computer Dependence of An Improvement of The Cell Neighbour-Table Method For Short-Range Potentials

Juan J. Morales[a]
[a] Departamento de Física Faculatad de Ciencias, Universidad de Extremadura, Badajoz, Spain

**To cite this Article** Morales, Juan J.(1996) 'Computer Dependence of An Improvement of The Cell Neighbour-Table Method For Short-Range Potentials', Molecular Simulation, 18: 5, 325 — 337
**To link to this Article: DOI:** 10.1080/08927029608024127
**URL:** http://dx.doi.org/10.1080/08927029608024127

*Molecular Simulation*, 1996, Vol. 18, pp. 325–337
Reprints available directly from the publisher
Photocopying permitted by license only



# COMPUTER DEPENDENCE OF AN IMPROVEMENT OF THE CELL NEIGHBOUR-TABLE METHOD FOR SHORT-RANGE POTENTIALS

JUAN J. MORALES

*Departamento de Física, Facultad de Ciencias, Universidad de Extremadura, 06071 Badajoz, Spain*

*(Received May 1996, accepted June 1996)*



A comparative test is presented for molecular dynamics, MD, computer simulation between the original Cell Neighbour-Table method, CNT, and a later development, the Link-Cell Neighbour-Table, LCNT. The test was simultaneously carried out on a vector Convex 210 computer and on a scalar Pentium 120. The comparison of the two methods for very large systems (up to 100000 particles) and for two short-range pair potentials (Weeks-Chandler-Andersen, WCA, and Lennard-Jones, LJ) showed that on the Convex, in vectorial mode, the LCNT method is about 25% more efficient in time than the CNT method for the shorter-range WCA potential, the difference being unnoticeable (less than 5%) for the LJ potential. However, when the scalar mode is on in the Convex, the difference disappears. In the Pentium, which was found to be systematically faster than the Convex even in vectorial mode, no significant difference of any kind between the both methods was found, with the either of the two potentials.

*Keywords:* Computer dependence for short-range potentials

## 1. INTRODUCTION

One major goal of Computational Physics is the introduction of new computational techniques, or the improvement of existing techniques, to make computer simulation of physical systems as accurate and fast as possible. In Molecular Dynamics, MD, one integrates numerically the classical equation of motion for a dense many-particle system in an iterative fashion for a number of discrete time-steps. If the particles interact via continuous, pair-

wise additive potentials, calculation of the pair force matrix at each time-step is the most time consuming computational task [1]. Efficiency of the force calculation consequently determines the performance of a MD program. Thus, programming strategies aimed at reducing the computational cost of a force evaluation play an important role in MD [2]. The basic philosophy behind these strategies is to eliminate less important interactions beforehand by considering only particles in a large enough neighbourhood of a given reference particle. Only interactions between the reference particle and those neighbouring particles are evaluated explicitly: all other interactions are neglected. Neighbouring particles can be identified using lists (i.e. one-or more-dimensional arrays) which need to be updated periodically because the neighbourhood of each reference particle will generally change on account of diffusion during the course of the simulation. The present article suggests a modified algorithm based on this general strategy which is aimed at pre-eliminating interactions prior to the actual force computation. The method described here, the Link-Cell Neighbour-Table, LCNT, is the result of developments of a previous method, the Cell Neighbour-Table, CNT, method [3]. In that study, the CNT method was compared in scalar and vectorial code with the standard Neighbour-Table, NT, and Link-Cell, LC, methods [4,5], and was found to be the most effective for all the systems studied (up to $N = 10^5$ particles), the effectiveness being greater with increasing system size. The Achilles' heel of the CNT method was that the memory size needed for the simulated systems was very large, although the increase in memory requirements was linear with respect to the system size. While, in principle, this is not a problem for a vectorial computer, for a scalar computer, where the memory is both limited and segmented, the speed of computation depends on the memory being used.

The present work deals with MD computer simulations of a classical system devoted to improving the CNT method, checked on a vectorial Convex 210 computer and on the currently fashionable new scalar Pentium 120, for two short-range pairwise potentials. Section 2 briefly describes the computational aspects of the LCNT method compared with the CNT, while Sect. 3 describes the details of the simulations and presents the results. A discussion and the conclusions are given in Sect. 4.

## 2. IMPROVING THE CNT METHOD: THE LCNT METHOD

It is well known that the performance of current techniques depends not only on the characteristics of the physical system and the simulation set-up

(state point, potential cut-off, number of particles, etc.) but also on the computer architecture (scalar, vectorial, parallel, etc.). However, for short-to-middle range interactions, whatever the simulation parameters or class of computer, currently-used techniques are mainly based on the earliest Verlet's NT method [4], or on the more recent Link-Cell method [5], or on an appropriate combination of the two [6,7]. As these two methods have been very well studied in the MD simulation literature [1], we consider it unnecessary to develope any further into their basic philosophy. Here, we shall rather describe in some detail the results obtained with the LCNT method and the comparison with its precursor, the CNT method [3].

A fundamental parameter in computer simulation is the range of the potential within which the particles interact, usually denoted by $r_{co}$, the cut-off distance. The force on particle $i$ due to all its neighbour particles $j$ within a distance $r_{co}$, has to be computed every time-step, and this is the most time-consuming process of the simulation. Computation time will be saved if one can avoid calculating the $j$ neighbour particles within $r_{co}$, every time-step. To this end, another spherical distance ($r_{sd} > r_{co}$) is defined allowing the updating of the neighbour table of radius $r_{sd}$ to be done every $n$ time-steps. The distance ($r_{sd}$-$r_{co}$) has to be chosen in such a way that, during the following $n$ steps, particles located within this skin can either remain in this same skin, cross to inside $r_{co}$, or cross to outside $r_{sd}$ but a particle can never cross the skin entirely, i.e., go from within $r_{co}$ to outside $r_{sd}$ or vice versa.

Unlike the NT method, in which the neighbours of each particle $i$ are stored in a one-dimensional array B(KK) [8], the original CNT method divided the simulation box up into rectangular cells (not necessarily of equal size), with edges much larger than $r_{sd}$ so as to permit the inclusion of enough particles inside for each particle to have its neighbour table coming mainly from its own cell. At a first stage, and after applying periodic boundary conditions, all the particles in the box are stored in a matrix BB ($NP, NC$), where $NC$ is the total number of cells into which the simulation box has been divided, and $NP$ the number of particles per cell, with $N = NC*NP$. The minimum image convention for cells is taken into account in the construction of the matrix BB. At a second stage, a matrix B(IK,2) is created storing the number of pairs of particles within the same and adjacent cells whose separations $r_{ij}$ are less than $r_{sd}$. The MD cycle then starts calculating the forces and the thermodynamic variables of interest for $n > 1$ steps, after which the matrices BB and B are updated, and so on.

The LCNT method is an amalgam of the LC and NT methods. Firstly, the simulation box is divided into equal-size cells with edges equal to, or

slightly greater than, $r_{sd}$. If $L_x$ and $L_y$ are the dimensions of the box in a two-dimensional system, the total number of cells, $NC$, which are automatically calculated in the program will be given by $NC = \text{NCX*NCY}$, where $\text{NCX} = \text{INTEGER}(L_x/r_{sd})$ and $\text{NCY} = \text{INTEGER}\ (L_y/r_{sd})$. The systems will thus have many cells with few particles per box (just the contrary of the case for the CNT method), so that a cell-linkage step must also be performed. In the traditional LC method [5], as the cell edges are $r_{co}$, there are few particles per cell (the cell could even be empty), and cell-linkage has to be performed every time-step $h$ to avoid a particle within a given cell escaping, or even diffusing through other cells. In the LCNT method of the present work, as the dimension of the cells is $r_{sd}$, and the particles are located in a two-dimensional surface, hence depending on $r_{sd}^2$, one can expect a greater number of particles than in the traditional LC method, permitting the creation of a neighbour table for the particles and the linkage of cells to be carried out only every $n$ time-steps, as in the NT and CNT methods. In this way, vector LCT (N) is created which links the particles, and, by performing a primary loop over all cells and a secondary loop over the nearest cells, it is possible to construct B(IK,2) as in the CNT method, but now avoiding the need to construct BB($NP, NC$).

A practical example comparing the LC and LCNT methods is illustrated in Figure 1 for a system of $N = 256$ particles. In the LC method the number

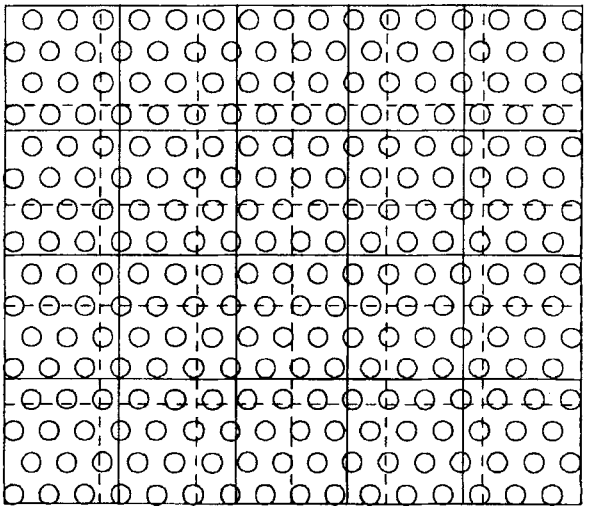

FIGURE 1 An illustrative example: The edges of a system of $N = 256$ particles is divided into an $r_{co} \cong 2.8\sigma$ grid (dashed lines) in the LC method, or an $r_{sd} \cong 3.37\sigma$ grid (full lines) in the LCNT method. For the discussion, see Sect. 2.

of cells is $NC=30$ and the average number of particles per cell (with $r_{co}\cong 2.8\sigma$) is $N'=8.5$, and cell-linkage has to be done every time step. For the LCNT method with $NC=20$, we have $N'=12.89$ (within $r_{sd}\cong 3.37\sigma$), i.e., 50% more particles than in the LC method, permitting linkage only every certain number of steps.

## 3. SIMULATION DETAILS AND RESULTS

As we wanted to test the efficiency of the LCNT method relative to the CNT under the same conditions, a wide range of systems were simulated, each with two potentials, one for very-short and one for short range interactions: the Weeks-Chandler-Andersen, WCA, and the Lennard-Jones, LJ, potentials [9]. The WCA is a very stiff potential that can be considered to be a particular case of the LJ with the attractive part left out. The cut-off for the WCA was $r_m=2^{1/2}\sigma=1.12246\sigma$, where the LJ potential has its minimum energy $\varepsilon$, and for the LJ potential was $r_{co}=2.5r_m\cong 2.8\sigma$. The equations of motion of the particles were integrated using a very accurate algorithm for the canonical MD (T,V,N) ensemble [10], with a unit time-step value of $h=0.005(mr_m^2/\varepsilon)^{1/2}$. The simulations were performed for a temperature $kT/\varepsilon=1.0$ (k is Boltzmann's constant) and density $\rho r_m^2=1.18$, which corresponds to a two-dimensional solid [11]. All the systems were started from their perfect traiangular lattice arrangement, updating the neighbour tables every 20 h for both methods.

As was described in Sect. 2, the LCNT method uses many cells with few particles inside, while the CNT method is just the reverse. In principle, in the LCNT method, the edges of the systems must be divided by $r_{sd}$, while in the CNT method the mean number of particles per cell is chosen to be about $N'=25-50$ (see Ref. [3]). In the first case, one chooses the number of cells, not the number of particles per cell, while in the second case, one chooses the particles per cell and not the dimension of the cells. In a first attempt to compare the two methods under the same conditions, some systems were checked by changing the number of cells into which they will be divided, and thus changing the number of particles per cell. Table I and II list the results of this test for WCA and LJ potentials, respectively, from the two computers. In the number of cells column, the last row for each system corresponds to the maximum number of cells, when the edges of the sytstems were divided by $r_{sd}=1.4r_m$ and by $r_{sd}=3.0r_m$ for the WCA and LJ potentials, respectively. Respecting the dependence of the CPU time on the number of cells, one can easily see that both methods tend to be faster when

TABLE I Results from the Convex 210 and Pentium 120 of the Preliminary test comparing the CNT and LCNT methods for different numbers of cells and for different systems, with the WCA potential. The entry $N'$ is the average number of particles per cell. The CPU time corresponds to 100 $h$

| $N$ | *number of cells* | $N'$ | *CNT method CPU (s) Convex, Pentium* | | *LCNT method CPU (s) Convex, Pentium* | |
|---|---|---|---|---|---|---|
| 2500 | 11 × 11 | 20.7 | 7.7 | 3.7 | 6.4 | 3.3 |
| | 20 × 20 | 6.3 | 6.4 | 3.1 | 5.1 | 2.9 |
| | 25 × 25 | 4.0 | 6.3 | 3.0 | 4.9 | 2.8 |
| | 35 × 30 | 2.4 | 6.2 | 3.0 | 4.8 | 2.8 |
| 4096 | 30 × 30 | 4.6 | 11.2 | 6.8 | 9.3 | 6.4 |
| | 45 × 39 | 2.3 | 11.0 | 6.7 | 9.1 | 6.2 |
| 10000 | 40 × 40 | 6.3 | 28.5 | 19.8 | 23.5 | 19.0 |
| | 70 × 61 | 2.3 | 27.6 | 18.9 | 22.2 | 18.3 |

TABLE II As Table I, but for the LJ potential

| $N$ | *number of cells* | $N'$ | *CNT method CPU (s) Convex, Pentium* | | *LCNT method CPU (s) Convex, Pentium* | |
|---|---|---|---|---|---|---|
| 2500 | 10 × 10 | 25.0 | 27.2 | 15.0 | 25.9 | 14.2 |
| | 16 × 14 | 11.2 | 26.2 | 14.5 | 24.5 | 13.7 |
| 4096 | 15 × 15 | 18.2 | 47.1 | 27.2 | 45.7 | 25.8 |
| | 21 × 18 | 10.8 | 46.2 | 26.7 | 44.4 | 25.4 |
| 10000 | 20 × 20 | 25.0 | 111.1 | 67.6 | 107.6 | 64.0 |
| | 32 × 28 | 11.2 | 105.9 | 64.9 | 102.2 | 62.3 |

TABLE III For the WCA potential, comparison of the CPU speeds(averaged over 100$h$) of Pentium and Convex for the CNT and LCNT methods, and of the CNT and LCNT methods for Convex and Pentium, where, for example, 108% means that the Pentium is 108% faster than the Convex (i.e., more than twice as fast)

| $N$ | *CNT method Conv./Pent. CPU (%)* | *LCNT method Conv./Pent. CPU (%)* | *Convex CNT/LCNT CPU (%)* | *Pentium CNT/LCNT CPU (%)* |
|---|---|---|---|---|
| 2500 | 108 | 79 | 26 | 8 |
| 4096 | 65 | 46 | 21 | 7 |
| 10000 | 45 | 23 | 23 | 4 |

TABLE IV As Table 3 but for the LJ potential

| $N$ | *CNT method Conv./Pent. CPU (%)* | *LCNT method Conv./Pent. CPU (%)* | *Convex CNT/LCNT CPU (%)* | *Pentium CNT/LCNT CPU (%)* |
|---|---|---|---|---|
| 2500 | 81 | 81 | 6 | 6 |
| 4096 | 73 | 76 | 4 | 5 |
| 10000 | 64 | 66 | 4 | 5 |

the number of cells is increased, or, in other words, when the number of particles per cell is smaller. Respecting the performance of the computers, the scalar Pentium 120 is always noticeably faster than the vectorial Convex 210. For a better comprehension of this result, Tables III and IV list for the WCA and LJ potential, respectively, the averaged CPU ratios between Convex and Pentium for a fixed method (either CNT or LCNT) and between the CNT and LCNT method for a fixed computer (either Convex or Pentium). The values in these tables show two interesting aspects. First, for both potentials and both methods, the effectiveness of Pentium over Convex diminishes with system size, the reduction in the difference being sharper for the WCA than for the LJ potential. It is also noticeable that the Convex/Pentium CPU ratio differences which appear between the CNT and LCNT with the WCA potential (Tab. III), not only disappear with the LJ potential, but tend to be inverted (Tab. IV). Second, comparing now the two methods on each computer (fourth and fifth columns in Tabs. III and IV), one can see that the potential and the computer truly determine the efficiency of the methods. While for the WCA potential the LCNT method is, on average, nearly 25% faster than the CNT on the Convex, on the Pentium it is only about 6% (Tab. III). For the LJ potential, however, this difference is about 5% on both computers (Tab. IV).

In the light of these preliminary results, and in order to see whether the same trend could be found for larger systems, a wider range of systems was

TABLE V Number of particles and cells of all the systems simulated, with the WCA ($r_{sd} = 1.4r_m$) and LJ ($r_{sd} = 3.0r_m$) potentials

| $N = NX*NY$ | *WCA potential* $NC = NCX*NCY$ | *LJ potential* $NC = NCX*NCY$ |
|---|---|---|
| 256 = 16*16 | 99 = 11*9 | 20 = 5*4 |
| 1024 = 32*32 | 418 = 22*19 | 90 = 10*9 |
| 2500 = 50*50 | 1050 = 35*30 | 224 = 16*14 |
| 4096 = 64*64 | 1755 = 45*39 | 378 = 21*18 |
| 5776 = 76*76 | 2438 = 53*46 | 525 = 25*21 |
| 7744 = 88*88 | 3286 = 62*53 | 725 = 29*25 |
| 10000 = 100*100 | 4270 = 70*61 | 896 = 32*28 |
| 14884 = 122*122 | 6364 = 86*74 | 1360 = 40*34 |
| 20164 = 142*142 | 8600 = 100*86 | 1840 = 46*40 |
| 24964 = 158*158 | 10656 = 111*96 | 2340 = 52*45 |
| 30276 = 174*174 | 12932 = 122*106 | 2793 = 57*49 |
| 40000 = 200*200 | 17202 = 141*122 | 3705 = 65*57 |
| 50176 = 224*224 | 21646 = 158*137 | 4599 = 73*63 |
| 60516 = 246*246 | 25950 = 173*150 | 5670 = 81*70 |
| 80656 = 284*284 | 34600 = 200*173 | 7533 = 93*81 |
| 99856 = 316*316 | 43039 = 223*193 | 9360 = 104*90 |

simulated each with the maximum number of cells. Table V lists the systems studied, where $N$ is the total number of particles, $NX$ and $NY$ are the particles along the $X$ and $Y$ axes, respectively, and $NC$ is the total number of cells obtained with the procedure described in Sect. 2.

Figure 2 is a plot of the CPU time used by the two methods with the repulsive potential WCA. On both computers the behaviour of the two methods is almost linear with respect to the system size. However, in the Pentium the two methods are practically indistinguishable, and the behaviour is closer to a straight line than the Convex. As found for the smaller systems studied previously, the LCNT method on the Convex is about 25% faster than the CNT. This quite noticeable difference in the Convex disappears when the attractive part of the potential is taken into account, i.e., when the full LJ potential is considered. Figure 3 shows in the two computers how close the experimental points are for the two methods with the LJ potential. Even though the LCNT is still always faster than the CNT, the time differences are quite small, about 5% on average on both computers.



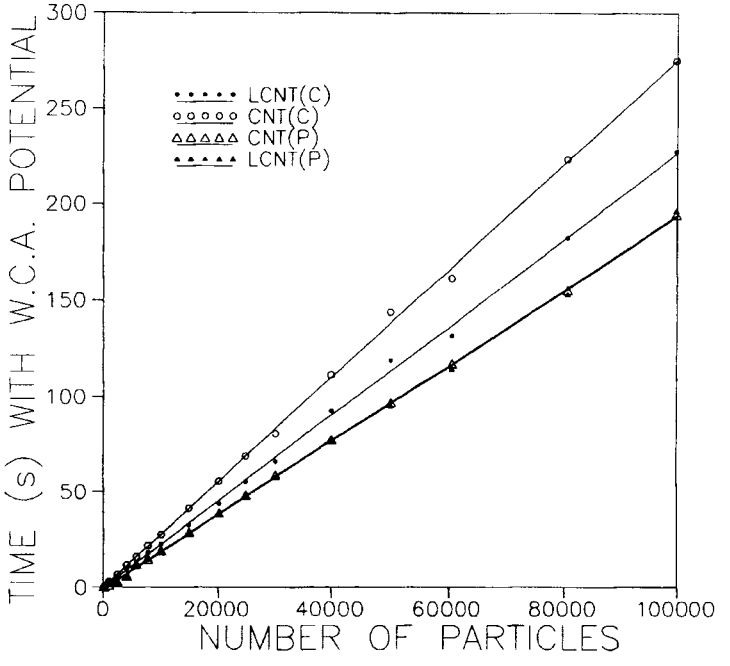


FIGURE 2 CPU-time comparison between the CNT and LCNT methods in Convex (C) and Pentium (P) with the WCA potential. Each point corresponds to the CPU time of 100 $h$.

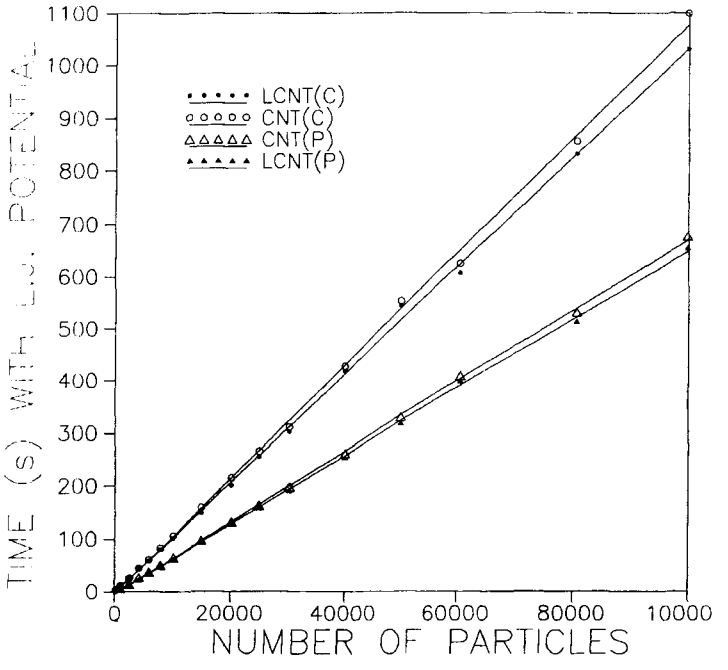


FIGURE 3 As Figure 2 but with the LJ potential.

The memory required is intrinsic to the Fortran code and is not machine-dependent, (although the treatment of the memory, of course, does depend on the computer's particular architecture [12]). The memory comparisons between the CNT and LCNT methods are shown in Figure 4. Both WCA (lower lines) and LJ (upper lines) potentials show a clearly linear behaviour. One sees that there are no significant differences of memory between the two methods, despite the difference in their philosophy.

All the results on the Convex were obtained using the vectorized versions of the Fortran programs. As netihter the NT nor the LC techniques can be fully vectorized and results can be affected by the computer architecture [12], a final check was made for all systems using the scalar mode of the Convex. The results of this test showed that there are no differences between the CNT and the LCNT methods, confirming what was found from the results in the Pentium. The CPU differences between the two methods with the WCA potential in the Convex are no longer seen when the computer mode is changed from vectorial to scalar. The behaviour is still linear, however, but with a greater slope than in Figures 2 and 3.

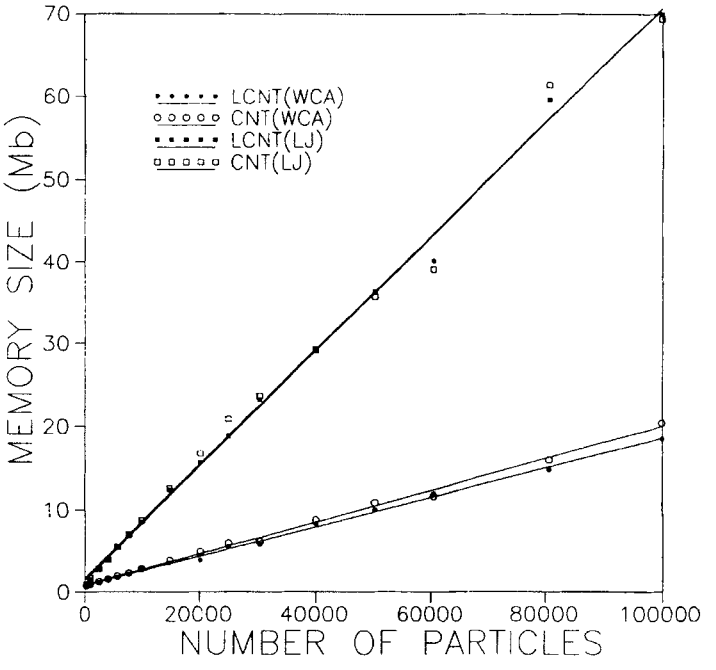


FIGURE 4 Memory-size comparison between the CNT and LCNT methods, for the WCA and LJ potentials.

Finally, a check was made of one of the problems in computer simulation that can lead to misleading results: the influence of the periodic boundary conditions, p.b.c., on the thermodynamic variables which one wants to study. While any simulated system is always finite, with p.b.c. it is possible to extrapolate the results to the thermodynamic limit ($N \to \infty$). However, depending on the size of the system being simulated the p.b.c. can affect the results to a greater or lesser degree. The statistical error of a thermodynamic quantity will vary as $N^{-1/2}$ [1] and therefore one should expect fluctuations at low $N$ to be greater than at high $N$. Figure 5 shows a clear example of this effect on the pressure, the basic quantity for most of the properties that might be calculated in a system. The large fluctuations in value for the smaller systems (up to $N = 5776$) are smoother for the intermediate systems (up to $N = 24964$) and disappear for the larger systems (from $N = 40000$ on). The plots seem to show that there is not systematic trend in the pressure and what one can see is a perfectly normal decrease in statistical error consistent with a constant value of $p$. This effect must be taken into account in any simulation independently of the properties to be calculated.

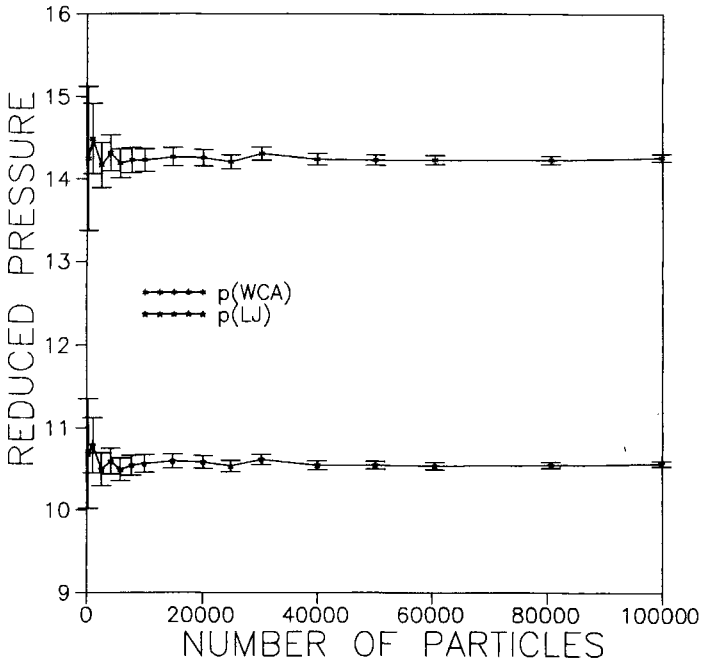


FIGURE 5 Behaviour of the pressure fluctuation with system size.



## 4. DISCUSSION AND CONCLUSIONS

The CNT method has shown itself to be very useful for molecular dynamics computer simulations for very large numbers of particles [3]. An improvement of this method, based on cell linkage and the creation of a neighbour table for particles, has been developed and compared here with the CNT method. In the comparison, two potentials were used, one for very-short range (WCA) and one for short range (LJ) interactions, and two computers, the scalar Pentium 120 and the vectorial convex 210.

In the preliminary test of the methods (Tab. I–IV), it was found in both computers that the LCNT and the CNT are both more effective when the simulation box is divided into cells with edges of length $r_{sd}$. If the cells are larger, then the methods are slower, due to CPU time being wasted. With smaller cells, the methods are faster but are liable to give wrong values for thermodynamic variables, since there are particles excluded from within $r_{sd}$ which contribute to the forces on a given particle $i$.

As outlined in Sect. 2, there is a certain difference in philosophy between the two methods, this is not reflected in the memory use which is nearly the same, as was seen in Figure 4. The LCNT method has a different distribution of memory from the CNT. While the latter uses a matrix BB $(NP, NC)$ of dimension $NP*NC = N$, the former uses a vector LCT(N). This means that the particular architecture of each computer treats the memory in different ways. Thus, while in the scalar Pentium computer there are no significant differences between the two methods for either potential, the different distribution of the memory is noticed when the Convex computer is used in vectorial mode, with the LCNT being faster than the CNT by nearly 25% for the very-short range repulsive interaction (Fig. 2). This difference disappears in the Convex when the attractive part of the potential is turned on (Fig. 3). These results, of course, do not depend on the quality of the potential that was used, whether WCA or LJ, but on the number of particles per subcell, i.e., on the dimension of the cells. While the spherical distance for the WCA potential was $r_{sd} = 1.4r_m$, for the LJ potential it was $r_{sd} = 3.0r_m$. Hence, for the few particles inside the small cells of edges $1.4r_m$, the vector mode of the Convex uses the LCT(N) vector more effectively than the BB $(NP, NC)$ matrix. This advantage is lost either when the edges of the cells are made larger ($r_{sd} = 3.0r_m$), i.e., with a greater number of particles per cell, or when the scalar mode is on.

Finally, the code used for the Fortran programs here, that can be supplied on request, has been kept completely standard, i.e., using no special in-line sentences as is the usual practice on the Cray or Cyber computers. This means that the CNT and LCNT methods can be performed effectively on any kind of computer, whether scalar or vectorial, and, more importantly, it may not be worthwhile spending time in improving Fortran code for a particular computer, as has been here and in Ref. [12]. With the advent of the new generation of personal computers represented by the Pentium 120, it is not necessary to use more expensive, sophisticated, and not easily manageable computers, such as the Cray, Cyber or parallel computers [13,14] to simulate systems large enough to permit the study of topics such as, critical phenomena, phase transitions, or Path Integrals [15].

***Acknowledgements***

This work has been supported in part by the Spanish DGICYT, Project No. PB95-0254 The cooperation of the University of Extremadura Computer Centre is gratefully acknowledged.

## *References*


[1] Allen, M. P. and Tildesly, D. J. (1989). *Computer Simulation of Liquids*, Clarendon, Oxford.

[2] Fincham, D. and Heyes, D. M. (1985). "Recent advances in molecular dynamics computer simulation", *In Dynamical Processes in Condensed Matter*, M.W. Evans, Ed., Wiley, New York, 493–575.

[3] Morales, J. J. and Toxvaerd, S. (1992)."The cell-neighbour table method in molecular dynamics simulations", *Comput. Phys. Commun.*, **71**, 71.

[4] Verlet, L. (1967). "Computer experiments on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules", *Phys. Rev.*, **159**, 98.

[5] Heyes, D. M. and Smith, W. (1987) "Cray vectorized link-cell code" *CCP5* No.26, and Heyes, D. M., (1988) "Correction to Cray vectorized link-cell code", *CCP5* No. **28**, 63.

[6] Grest, G. S., Dünweg, B. and Kremer, K. (1989). "Vectorised link-cell Fortran code for molecular dynamics simulations for a large number of particles", *Comput. Phys. Commun.*, **55**, 269.

[7] For a general review, see, for example, Fincham, D. (1989). "Molecular dynamics on vector and parallel computers. An annotated bibliography", *CCP5* No.31.

[8] Morales, J. J. (1993) "Path Integral Theory: An improved simulation for the forces in semiclassic systems", *J.Comput. Chem.*, **14**, 728.

[9] Weeks, J. D., Chandler, D. and Andersen, H. C. (1971). "Role of repulsive forces in determining the equilibrium structure of simple liquids", *J.Chem. Phys.*, **54**, 5237.

[10] Toxvaerd, S. (1991). "Algorithms for canonical molecular dynamics simultions", *Mol. Phys.*, **72**, 159.

[11] Morales, J. J. (1994). "Size and time dependence of the elastic constants of a two-dimensional solid near melting", *Phys. Rev. E*, **49**, 5127.

[12] Morales, J. J. and Nuevo, M. J. (1992). "Comparison of link-cell and neighbourhood tables on a range of computers", *Comput. Phys. Commun.*, **69**, 223.

[13] Fincham, D. (1987). "Parallel computers and molecular simulation," *Molecular Simulation*, **1**, 1.

[14] Pinches, M. R. S., Tildesley, D. J. and Smith, W. (1991). "Large scale molecular dynamics on parallel computers using the link-cell algorithm", *Molecular Simulation*, **6**, 51.

[15] Morales, J. J. and Nuevo, M. J. (1995) "Path Integral molecular dynamics methods: Application to Neon", *J.Comput. Chem.*, **16**, 105.