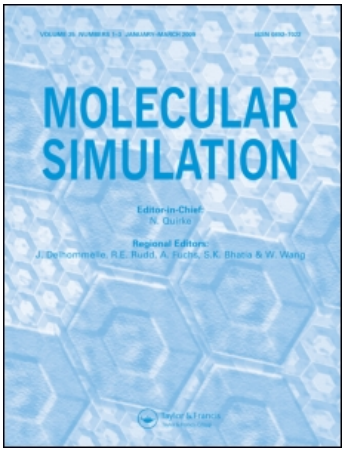

Molecular Simulation
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t713644482

Linked Lists and the Method of Lights in Molecular Dynamics Simulation - Search for the Best Method of Forces Evaluation in Sequential MD Codes

Witold Dzwinel[a]; Monika Bargiel[a]; Jacek Kitowski[a]; Jacek Mościński[a]
[a] Institute of Computer Science, Cracow, Poland

To cite this Article Dzwinel, Witold , Bargiel, Monika , Kitowski, Jacek and Mościński, Jacek(1989) 'Linked Lists and the Method of Lights in Molecular Dynamics Simulation - Search for the Best Method of Forces Evaluation in Sequential MD Codes', Molecular Simulation, 4: 4, 229 — 239
To link to this Article: DOI: 10.1080/08927028908022365
URL: http://dx.doi.org/10.1080/08927028908022365

Molecular Simulation, 1989, Vol. 4, pp. 229–239
Reprints available directly from the publisher
Photocopying permitted by license only

LINKED LISTS AND THE METHOD OF LIGHTS IN MOLECULAR DYNAMICS SIMULATION – SEARCH FOR THE BEST METHOD OF FORCES EVALUATION IN SEQUENTIAL MD CODES

WITOLD DZWINEL, MONIKA BARGIEL, JACEK KITOWSKI
and JACEK MOŚCIŃSKI

Institute of Computer Science, AGH al. Mickiewicza 30, 30-059 Cracow, Poland

(*Received May 1989, accepted August 1989*)

In this paper two methods of forces evaluation used in the MD codes are presented and compared against the classical *linked lists* algorithm [3,4] and its modified version [5]. The first algorithm, so called the *method of lights* is a sequential version of the CYBER 205 vector oriented code [6]. A new algorithm of forces evaluation is also proposed, which incorporates advantages of the *method of lights* and the *linked lists* technique.

KEY WORDS: Linked lists, method of lights, forces evaluation, list processing.

INTRODUCTION

The need for efficient MD codes is revealed in a number of papers. It is true of advanced architecture computers as well as standard sequential machines. Many users perform simulations with cheap table-top PCs, and have MD codes which of necessity need to take advantage of their limited computational power in the most effective way. Our C-language MD program [1] simulates the dynamics of simple liquid mixtures in three dimensional space. It is suitable for several thousand particles and can be used efficiently in a PC/workstation environment. With the assistance of a DSI-020 acceleration board one weekend is needed for a 3000 timestep simulation of the dynamics of 2916 argon and krypton atoms.

The bottleneck of the MD code throughput is the interparticle forces computation. For the short-range interactions (to which this paper is restricted) the cutoff radius (R_{cut}) and periodic boundary conditions application enable bulk dynamics to be simulated using a relatively small number of particles. The direct forces computation scales as N^2, (N = number of particles), to overcome this problem some sophisticated methods have been developed. One of them is Verlet's algorithm [2], which needs, however, large amount of operating memory for neighbours array storing, even for relatively small number of particles.

The very attractive *linked lists* concept, introduced by Hockney and Eastwood [3] and implemented by e.g. Smith [4], has o(N) complexity and is intended for simulation of large particle ensembles. Its modified C-language version, in which the *cube technique* concept was applied, was presented in the CCP5 Newsletter [5]. The *cube*

technique differs from the original *linked list* method in some additional conditions limiting the number of neighbours of each particle to those confined in the cube, $C\{\vec{r}(i), 2R_{cut}\}$, where $\vec{r}(i)$ and $2\ R_{cut}$ are the centre and the side length of the cube respectively. Thus the squared distance between two particles i and j is not evaluated unless the differences in particle coordinates satisfy $|r_\alpha(i) - r_\alpha(j)| < R_{cut}$ where $r_\alpha(i)$ and $r_\alpha(j)$ are the i-th and j-th particle coordinates for $\alpha \in \{x,y,z\}$ and $i,j = 1, \ldots, N$, $i \neq j$.

Another approach, the so-called *method of lights*, is proposed by Sullivan, Mountain and O'Connell [6]. Although this algorithm is adjusted to vector properties of the CYBER 205, we have used a similar approach to produce an efficient sequential C-language code.

The second algorithm proposed here, the *sorted linked list* method, incorporates advantages of the *method of lights* and the *linked lists* technique.

THE METHOD OF LIGHTS - SEQUENTIAL ALGORITHM

The original version of the *method of lights* [6] gives an efficient way to find the neighbours of each particle confined in the cube $C\{\vec{r}(i), 2\ R_{cut}\}$. It is done by the particle coordinates sorting along x,y and z directions and using some auxiliary arrays. Then the neighbour lists are applied for forces evaluation (similar to Verlet's algorithm).

The neighbour arrays are introduced to take the full advantage of the vector properties of CYBER 205. However, on sequential machines such an approach seems to be inconvenient. It is more memory consuming than the usual Verlet's algorithm, because on average only $\pi/6$ "cube neighbours" will be used for forces computation. Thus we used a modified version of the *method of lights* in which the neighbouring arrays are not applied and consequently the interparticle forces are evaluated simultaneously with particle neighbours determination.

The coordinate sorting determines auxiliary arrays L_α which contain particle indices ordered in respect to the α coordinate, such that

$$r_\alpha(L_\alpha(1)) \leqslant r_\alpha(L_\alpha(2)) \leqslant \ldots \leqslant r_\alpha(L_\alpha(N)). \tag{1}$$

Let n be the order index in the L_α array corresponding to the i particle index in the coordinate array, i.e.

$$i = L_\alpha(n). \tag{2}$$

The neighbourhood of each particle i in the α coordinate is determined by the pair of pointers b_α and e_α, which mean *beginning* and *end* of the particle neighbourhood respectively. They indicate the boundary indices of L_α array, so that the condition $|r_\alpha(i) - r_\alpha(j)| < R_{cut}$ is fulfiled for all particles $j = L_\alpha(b_\alpha), L_\alpha(b_\alpha + 1), \ldots, L_\alpha(e_\alpha - 1), L_\alpha(e_\alpha)$. These pointers are determined by means of the algorithm presented in scheme 3 [6] and stored in Tb_α and Te_α tables respectively. The computation box side lengths are equal to 1 (i.e. $r_\alpha(i) \in [0,1]$).

```
begin
        ad: = 0.0
        {initialize e_α and b_α for particle L_α(1)}
        for n: = 2 to N do
            begin
            while (((r_α(L_α(e_α)) + ad) − r_α(L_α(n))) < R_cut) do
                    begin
                    e_α: = e_α + 1
                    if (e_α > N) then
                        begin
                        e_α: = 1;                                   (3)
                        ad: = 1.0
                        end
                    end
            if (e_α = 1) then e_α: = N + 1
            Te_α(L_α(n)): = e_α − 1
            {similar code for b_α}
            .
            .
            Tb_α(L_α(n)): = b_α
            end
end.
```


For each particle i one can determine a set, $\omega(i)$, containing particles confined in the cube $C\{\vec{r}(i), 2R_{\text{cut}}\}$. The set $\omega(i)$ is determined using Tb_α and Te_α arrays as the intersection of sets of neighbouring particles for particular directions $\omega_\alpha(i)$.

$$\omega(i) = \omega_x(i) \cap \omega_y(i) \cap \omega_z(i). \tag{4}$$

The sets $\omega_\alpha(i)$ are defined (with necessary corrections resulting from periodic boundary conditions) by

$$\omega_\alpha(i) = \{j | r_\alpha(L_\alpha(b_\alpha)) < r_\alpha(j) < r_\alpha(L_\alpha(e_\alpha))\}. \tag{5}$$

Additional arrays R_α assign to each particle i its positions n in L_α tables, i.e.

$$R_\alpha(i) = R_\alpha(L_\alpha(n)) = n. \tag{6}$$

Let $Tb_x(i) \leqslant m \leqslant Te_x(i)$. Thus all particles $j = L_x(m)$ are neighbours of the particle i in respect of the x coordinate ($j \in \omega_x(i)$). In order to check if $j \in \omega(i)$ the following conditions are verified

$$\begin{aligned} Tb_y(i) &\leqslant R_y(j) \leqslant Te_y(i), \\ Tb_z(i) &\leqslant R_z(j) \leqslant Te_z(i). \end{aligned} \tag{7}$$

An important point of the algorithm are efficient periodic boundary conditions (*pbc*). B_α is defined as follows:

$$B_\alpha(i) = \begin{cases} \frac{1}{2} \text{ if } n < Tb_\alpha(i), \\ -\frac{1}{2} \text{ if } n > Te_\alpha(i), \\ 0 \text{ otherwise.} \end{cases} \tag{8}$$

Thus the coordinate differences $r_\alpha(i,j)$ between i and j particles are

$$r_\alpha(i,j) = r_\alpha(i) - r_\alpha(j) + [B_\alpha(i) - B_\alpha(j)] \tag{9}$$

where [ ] means the greatest integer function.

In order to avoid unnecessary computations the main loop over all particles is split into two separate parts (see scheme 10). The first one refers to these i particles for which $Tb_\alpha(i) > n$ (i.e. when *pbc* are to be used), while the second one concerns the remaining particles.

```
main loop
    begin
    {sorting and determination of arrays L_α, B_α, Tb_α, Te_α}
    for i: = 1 to N do F_α(i): = 0
    Φ*: = 0
    Ψ: = 0
    n: = 1
    i: = L_x(n)
    repeat
        s: = Ξ(i)
        block 1                                              (10)
        n: = n + 1
        i: = L_x(n)
    until (Tb_x(i) < n)
    bg: = n
    for n: = bg to N do
        begin
        i: = L_x(n)
        s: = Ξ(i)
        block 2
        end
    end,
```

where

s – kind of particle i,
Ξ – array of particles kinds,
$\Phi = \Phi^*/2$ – potential energy,
Ψ – virial,
$\vec{F}(i)$ – force acting on particle i,

```
block 1
    begin
    r_x(i): = r_x(i) + 1.0
    for m: = Tb_x(i) to N do
        block 3                                              (11)
    r_x(i): = r_x(i) - 1.0
    for m: = 1 to n - 1 do
        block 3
    end.
```

block 2
begin
for $m = Tb_x(i)$ to $n - 1$ do. (12)
block 3
end

In **block 3** the set $\omega(i)$ is determined by checking the conditions (7). These conditions are coupled into one relational expression (see schemes 13 and 14), alternatively using *and* or *or* operators depending on $B_\alpha(i)$ values.

block 3
begin
$j = L_x(m)$
if $(((R_y(j) < Te_y(i))\ op_y(i)\ (R_y(j) > Tb_y(i)))$ and
$((R_z(j) < Te_z(i))\ op_z(i)\ (R_z(j) > Tb_z(i)))$ and (13)
$\sum_\alpha r_\alpha^2(i,j) < R_{cut}^2)$ then **block common**
end,

where

$$op_\alpha(i) = \begin{cases} and \text{ if } B_\alpha(i) = 0, \\ or \text{ if } B_\alpha(i) \neq 0. \end{cases} \quad (14)$$

In block common forces, potential energy and virial are computed [1,4]

block common

begin
$s' := \Xi(j)$
$k := X(s,s')$
$l := [(r^2(i,j) - \varrho^2(k))/\Delta r^2]$
$\Delta u(k) := u(k,l+1) - u(k,l)$
$f_\alpha(i,j) := -\Delta u(k) \cdot r_\alpha(i,j)$
$F_\alpha(i) := F_\alpha(i) + f_\alpha(i,j)$
$F_\alpha(j) := F_\alpha(j) - f_\alpha(i,j)$
$\Phi^* := \Phi^* + (u(k,l+1) + u(k,l))$
$\Psi := \Psi - r^2(i,j) \cdot \Delta u(k)$
end,

where

s' – kind of particle j,
X – array of pair potential kinds,
$\varrho(k)$ – minimum distance from which potential is tabulated,
Δr – step of potential tabulation,
$u(k,l)$ – potential tables (see [1,4]),

It is shown theoretically [6], that the *method of lights* scales as $N^{5/3}$, assuming that coordinate array sorting consumes a negligible amount of the total CPU time. Since our sequential algorithm differs mainly from that described in [6] in the *pbc* treatment (which does not influence its complexity) it is estimated to be $o(N^{5/3})$ also.

SORTED LINKED LISTS METHOD

The *cube technique* and the *method of lights* have one common feature. They limit the number of neighbours of a given particle to those confined in the $C\{\vec{r}(i), 2R_{cut}\}$ cube. The *method of lights* gives an easy way to find them out but, on the other hand, linked lists based algorithms have less complexity equal to o(N). Joining the advantages of both the methods, a new efficient algorithm can be obtained. In the method, the neighbouring particles are determined as in the *linked-list* technique and next some of them (those not confined in $C\{\vec{r}(i), 2R_{cut}\}$) are removed from the distance evaluation on the basis of the boundary tables Tb_α and Te_α as in the *method of lights*. The modifications related to the *linked lists* method are listed below. The combined method scales as the *linked lists* one (i.e. o(N)), since the *method of lights* is considered for every link cell separately (with approximately fixed number of particles).

The arrays R_α, Tb_α and Te_α retain their meaning as used in the *method of lights* and $c_\alpha(i)$ means the integer α coordinate of the elementary cell containing the particle i. The cell coordinates are kept in auxiliary arrays, S_α, such that $S_\alpha(n) = c_\alpha(i)$. The list arrays LL_x and LL_z contain linked lists of particles within each cell, sorted decreasingly in respect to x and z coordinates respectively. The arrays HOC_x and HOC_z are "heads" for the corresponding linked lists. A (-1) entry in $HOC_\alpha(c)$ indicates that there are no particles in cell C. Another entry gives the address of the coordinates of the first particle in the list. Similarly, the link coordinate of a particle either gives the address of the coordinates of the next molecule in the list or is equal to -1 to indicate the end of the list.

Periodic boundary conditions are used with help of tables MIC_α

$$MIC_\alpha(c_\alpha) = \begin{cases} -NC_\alpha \text{ for } c_\alpha = -1, \\ 0 \text{ for } 0 \leqslant c_\alpha < NC_\alpha, \\ NC_\alpha \text{ for } c_\alpha = NC_\alpha. \end{cases} \tag{16}$$

NC is the number of cells in α direction equal to the computational box side length (i.e. in this method $r_\alpha(i) \in [0, N, C_\alpha]$). $NC(= NC_x \cdot NC_y \cdot NC_z)$ is the total number of cells. The cell index c is usually computed using the equation

$$c = c_x + NC_x \cdot (c_y + NC_y \cdot c_z). \tag{17}$$

In the most time consuming parts of the program Equation (17) is replaced by

$$c = NCB_x(c_x) + NCB_y(c_y) + NCB_z(c_z), \tag{18}$$

where number of real operatiaons is reduced and NCB_α are auxiliary tables conserving the meaning of Equation (17).

For the molecule i lying in cell $c(i)$ the neighbouring particles are those from cell $c(i)$ and 26 neighbouring cells. Due to the third Newton's law, only half of them are taken into account (all cells lying above the current one and half cells from the same level). In order to determine coordinates of the jc-th neighbouring cell ($jc = 1, \ldots, 13$), the tables NH_α are introduced. The value of $NH_\alpha(jc)$ means the shift of the jc neighbouring cell in α direction relative to the current cell. Thus NH_α tables elements take on values from $[-1, 1]$.

The primary loop for forces evaluation is organized over all cells of the computational box. Calculations of the individual molecule forces are done within the secon-

dary loop. Due to efficiency the secondary loop is split into three blocks [5]. The principal modifications of the *sorted linked lists* technique in comparison with the *linked lists* method refer to two out of the three blocks.

```
for c(i) = 0 to NC − 1 do
    begin
    block 1
    for jc = 1 to 9 do {cells with NH_z(jc) = 1}
        block 2
    for jc = 10 to 13 do {cells with NH_z(jc) = 0}
        block 3
    end
```
(19)

The first block, kept unchanged, is responsible for the interparticle forces evaluation in the reference cell only.

```
block 1
    begin
    i = HOC_z(c(i))
    while (i > −1) do
        begin
        s := Ξ(i)
        j := LL_z(i)
        while (j > −1) do
            begin
            r_α(i,j) := r_α(i) − r_α(j)
            r²(i,j) := Σ_α r²_α(i,j)
            if (r²(i,j) < R²_cut) then block common
            j := LL_z(j)
            end
        i := LL_z(i)
        end
    end
```
(20)

The second block is used for forces computation between particles i from the reference cell and particles j from cells above the reference one ($c_z(j) > c_z(i)$). Before forces evaluation for each particle i from the reference cell the conditions are checked whether the particle has its neighbouring particles in the $c(j)$ cell (i.e. if all the boundary indices exceed the reference cell – see statements marked by (*)). The internal loop to evaluate distances between particle i and particles j is performed only for these i for which those conditions are satisfied. Within this loop the **block common** is executed in respect to another complex condition (see statements marked by (***)). The loop marked by (**) finds the first particle j which is a neighbour of i with respect to z direction

block 2

```
    begin
    c_α(j) := c_α(i) + NH_α(jc) (for α ∈ {x,y,z})
    mic_α := MIC_α(c_α(j)) (for α ∈ {x,y,z,})
    c(j) := NCB_x(c_x(j)) + NCB_y(j)) + NCB_z(c_z(j))
    i := HOC_z(c(i))
    k := HOC_z(c(j))
(*) while (i > −1 and S_z(Te_z(i)) ≠ c_z(i)) do
        begin
(*)     if ((NH_x(jc) = 0 or S_x(l_x) ≠ c_x(i)) and
(*)        (NH_y(jc) = 0 or S_y(l_y) ≠ c_y(i))) then
           begin
(**)       while (R_z(k) > Te_z(i)) do k := LL_z(k)        (21)
           j := k
           s := Ξ(i)
           r_α(i) := r_α(i) − mic_α
           while (j > −1) do
                 begin
(***)            if (cond_x(i,j) and cond_y(i,j) and
                 Σ_α r_α²(i,j) < R²_cut) then block common
                 j := LL_z(j)
                 end
           end
        i := LL_z(i)
        end
    end,
```

where

$$I_\alpha = \begin{cases} Te_\alpha(i) \text{ if } NH_\alpha(jc) > 0, \\ Tb_\alpha(i) \text{ if } NH_\alpha(jc) < 0 \end{cases} \tag{22}$$

and

$$cond_\alpha(i,j) = \begin{cases} R_\alpha(j) \leqslant Te_\alpha(i) \text{ if } NH_\alpha(jc) > 0, \\ \text{'true'} \qquad\qquad \text{if } NH_\alpha(jc) = 0, \\ R_\alpha(j) \geqslant TB_\alpha(i) \text{ if } NH_\alpha(jc) < 0. \end{cases} \tag{23}$$

The third forces procedure block is responsible for forces computations between particles from the reference cell and from neighbouring cells such that $c_x(j) \leqslant c_x(i)$ and $c_z(j) = c_z(i)$. The conditional loop (*) performed for all pairs of cells $c(i)$ and $c(j)$ lets to eliminate from distances computation all i particles which have no neighbours in $c(j)$ in respect to x direction. The similar condition is used for y direction (**). The loop for distance calculation between particles i and j differs from the basic linked lists loop in checking additional conditions resulting from boundary pointers $Tb_\alpha(i)$ and $Te_\alpha(i)$ (***).

```
block 3
    begin
    c_α(j) := c_α(i) + NH_α(jc) (for α ∈ {x,y})
    mic_α := MIC_α(c_α(j)) (for α ∈ {x,y})
    c(j) := NCB_x(c_x(j)) + NCB_y(c_y(j)) + NCB_z(c_z(i))
    i := HOC_x(c(i))
    k := HOC_x(c(j))
(*) if (NH_x(jc) < 0) then
(*)       while (R_x(k) < Tb_x(i) or S_x(Tb_x(i)) = c_x(i)) do i = LL_x(i)
    while (i > -1) do
          begin
(**)      if (NH_y(jc) ≠ 0 and S_y(l_y) = c_y(i)) then
            begin
            s := Ξ(i)                                                    (24)
            r_α(i) := r_α(i) - mic_α (for α ∈ {x,y})
            j := k
(***)       while (j > -1 and cond_x(i,j)) do
                  begin
(***)             if (cond_y(i,j) and Σ_α r_α^2(i,j) < R_cut^2) then
                        block common
                  j  LL_x(j)
                  end
            end
          i := LL_x(i)
          end
    end.
```

Within every type of the secondary loop the **block common** is used as described by Equation (15).

RESULTS AND CONCLUSIONS

For all algorithms presented here C-language programs were developed and their timings were measured on the IBM PC/AT clone supported with DSI-020 acceleration board (MC68020/68881, 12.5 MHz, 1MB RAM). The test runs were carried out for Ar/Kr mixture in 116 K [7] and $R_{cut} = 2.5\sigma$ (Figure. 1), $R_{cut} = 1.5\sigma$ Figure. 2) and $R_{cut} = 3.5\sigma$ (Figure 3).

For the tested numbers of particles, algorithms based on the *method of lights* and the *sorted linked lists* method are almost twice as fast as the *linked lists* code. This advantage is more evident for $R_{cut} = 3.5\sigma$, when the *linked lists* method efficiency is decreased due to cell sizes enlargement forced by R_{cut} and consequently many unnecessary distance calculations with particles not confined in $C\{\vec{r}(i), 2R_{cut}\}$. Since the computational time per particle for the *method of lights* increases with N (which confirms its nonlinear complexity estimated in [6]) this algorithm is efficient for the moderate numbers of particles ($N = 1000$–2000). For very large N *linked lists* based techniques (for instance the *sorted linked lists* method) are prefered.

Estimated average time for forces computation in MD simulation step using

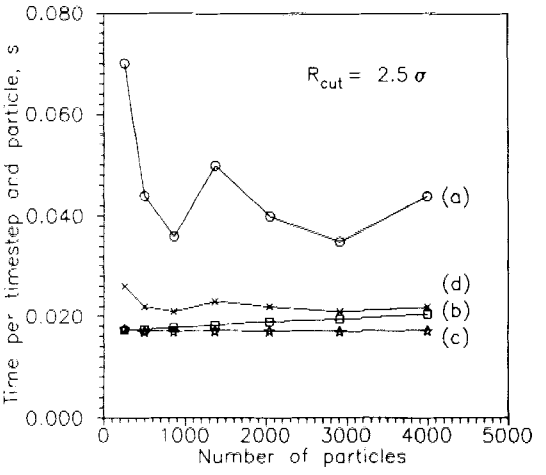

Figure 1 Comparison of time averages for MD timestep per article for $R_{cut} = 2.5\sigma$ and various forces procedures
(a) Smith algorithm [4], C-language code,
(b) *method of lights*, optimized sequential realization,
(c) *sorted linked lists* algorithm,
(d) *cube technique* [1,5].

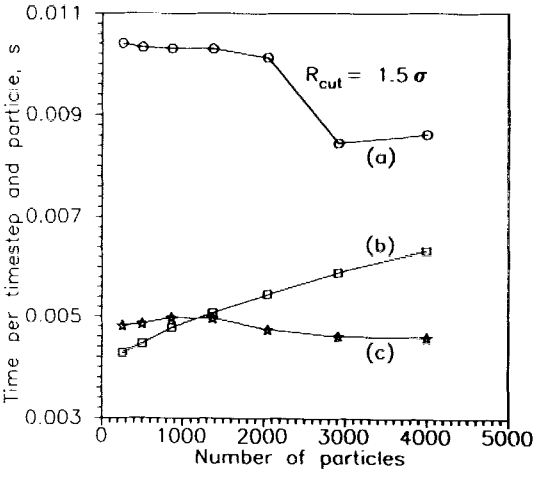

Figure 2 Comparison of time averages for MD timestep per particle for $R_{cut} = 1.5$ and various forces procedures
(a) Smith algorithm [4], C-language code,
(b) *method of lights*, optimized sequential realization,
(c)*sorted linked lists* algorithm.

theoretical "pure" Verlet's algorithm (without neighbours table updating, assuming no time for loops organization, etc.) is only about 20% less than obtained by the *sorted linked lists* method [8]. Bearing this in mind, is further significant improvement of sequential MD algorithms possible?

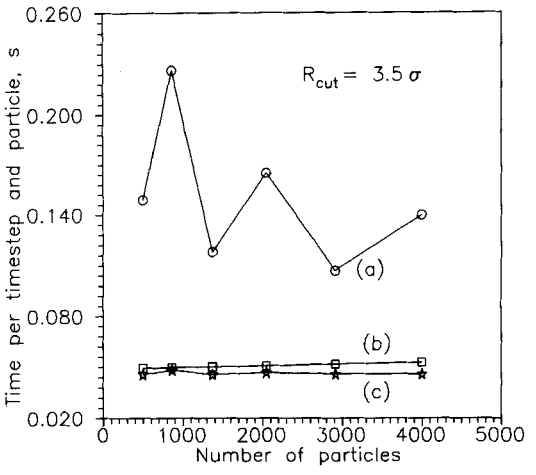

Figure 3 Comparison of time averages for MD timestep per particle for $R_{cut} = 3.5\sigma$ and various forces procedures
(a) Smith algorithm [4], C-language code,
(b) *method of lights*, optimized sequential realization,
(c) *sorted linked lists* algorithm.

Acknowledgements

We are grateful to Professor Jerzy Kapelewski for arranging partial financial support under Project number CPBP 01.08 D4.3.

References

[1] J. Mościński, W. Dzwinel, J. Kitowski and M. Bargiel, "C-language Molecular Dynamics Program for the Simulation of Monoatomic Molecular Mixtures", prepared for publication in *Comput. Phys. Commun.*
[2] L. Verlet, "Computer experiments on classical fluids", *Phys. Rev.*, **159**, 98 (1967).
[3] R.W. Hockney and J.W. Eastwood, "Computer Simulation Using Particles", McGraw-Hill, New York, 1981.
[4] W. Smith, "Fortran Code for the LINK-CELL Method" *CCP5 Information Quarterly for Computer Simulation of Condensed Phases*, informal Newsletter, Daresbury Laboratory, No. 20, 52 (1986).
[5] J. Mościński, W. Dzwinel and J. Kitowski, " C-language Code Running on PC?? How Forces() Procedure Looks Like?" *CCP5 Information Quarterly for Computer Simulation of Condensed Phases*, informal Newsletter, Daresbury Laboratory, No. 27, 26 (1988).
[6] F. Sullivan, R.D. Mountain and J. O'Connell, "Molecular dynamics on vector computers", *J. Comput. Phys.*, **61**, 138 (1985).
[7] W. Smith, "The program ADMIXT: A Molecular Dynamics Program for the Simulation of Monoatomoc Liquid Mixtures" (CCP Program Library, 1983).
[8] W. Dzwinel, Ph.D. thesis (Academy of Mining and Metallurgy, Kraców, 1988, in polish).