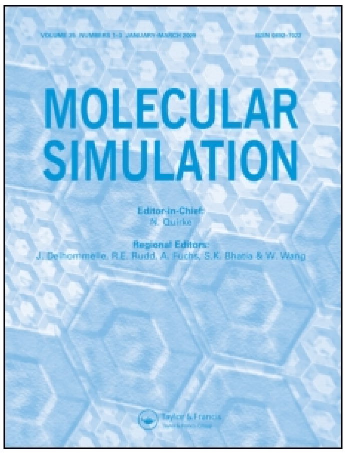
This article was downloaded by:
On: *14 January 2011*
Access details: *Access Details: Free Access*
Publisher *Taylor & Francis*
Informa Ltd Registered in England and Wales Registered Number: 1072954 Registered office: Mortimer House, 37-41 Mortimer Street, London W1T 3JH, UK



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

Large Timesteps in Molecular Dynamics Simulations

David Macgowan[ab]; David M. Heyes[ac]
[a] Research School of Chemistry, Australian National University, Canberra, A.C.T., Australia [b] BP Research Centre, Middlesex, United Kingdom [c] Department of Chemistry, Royal Holloway and Bedford New College, University of London, Surrey, United Kingdom

To cite this Article Macgowan, David and Heyes, David M.(1988) 'Large Timesteps in Molecular Dynamics Simulations', Molecular Simulation, 1: 5, 277 — 297
To link to this Article: DOI: 10.1080/08927028808080950
URL: http://dx.doi.org/10.1080/08927028808080950

PLEASE SCROLL DOWN FOR ARTICLE

Full terms and conditions of use: http://www.informaworld.com/terms-and-conditions-of-access.pdf

This article may be used for research, teaching and private study purposes. Any substantial or systematic reproduction, re-distribution, re-selling, loan or sub-licensing, systematic supply or distribution in any form to anyone is expressly forbidden.

The publisher does not give any warranty express or implied or make any representation that the contents will be complete or accurate or up to date. The accuracy of any instructions, formulae and drug doses should be independently verified with primary sources. The publisher shall not be liable for any loss, actions, claims, proceedings, demand or costs or damages whatsoever or howsoever caused arising directly or indirectly in connection with or arising out of the use of this material.

Molecular Simulation, 1988, Vol. 1, pp. 277–297 . © 1988 Gordon and Breach Science Publishers S.A.
Photocopying permitted by license only Printed in Great Britain
Reprints available directly from the publisher

LARGE TIMESTEPS IN MOLECULAR DYNAMICS SIMULATIONS

DAVID MACGOWAN[§]

Research School of Chemistry, Australian National University, G.P.O. Box 4, Canberra, A.C.T. 2601, Australia.

DAVID M. HEYES

Research School of Chemistry, Australian National University G.P.O. Box 4, Canberra, A.C.T. 2601, Australia; and Department of Chemistry, Royal Holloway and Bedford New College, University of London, Egham, Surrey TW20 0EX, United Kingdom.[¶]

(*Received October 1987*)

Downloaded At: 20:01 14 January 2011

Both equilibrium and nonequilibrium molecular dynamics simulations are carried out for two state points of the Lennard-Jones fluid, using leapfrog algorithms. In the equilibrium simulations we obtain internal energies, pressures, radial distribution functions and velocity autocorrelation functions. In the nonequilibrium simulations we obtain the relevant transport coefficients; additionally, the radial distribution function and velocity autocorrelation function in a shearing fluid are computed. It is found that, provided the accuracy of the particle trajectories is fully utilised in calculating their velocities, much larger timesteps than are customary can be used without significant drift in the results. We are thus able to take full advantage of the well known stability of the leapfrog algorithm and also of the even greater stability of its modifications for isokinetic simulations.

KEY WORDS: Timestep, molecular dynamics, leapfrog algorithm, isokinetic.

1 INTRODUCTION

Broadly speaking, the community of molecular dynamics simulators divides into supporters of the low order leapfrog algorithm [1] and those using higher order Gear predictor–corrector algorithms [2]. There are few workers simultaneously using both methods, though there have apparently been a few ‘conversions’ in both directions. Other finite difference schemes have been used in molecular dynamics from time to time but it is fair to say that none has attracted a wide and loyal following. In the present paper we shall concentrate almost exclusively on the leapfrog algorithm apart from brief reference to Gear algorithms for purposes of comparison.

It is helpful to use as large a time step as possible in molecular dynamics in order to travel through phase space with the smallest possible number of force evaluations per unit real time. Obviously, however, this should not be done unless a sufficiently

[§]Permanent address: BP Research Centre, Chertsey Road, Sunbury-on-Thames, Middlesex TW16 7LN, United Kingdom.
[¶]Permanent address.

accurate trajectory is obtained. The motivation for the present work was an attempt to make better use of rather limited computing power, but a larger time step will always permit more results to be obtained for any given computer and so the methods described here should extend the potential projects which can be tackled even for those with access to the most powerful supercomputers.

In comparing the relative merits of leapfrog and Gear algorithms in the context of molecular dynamics, we at first restrict attention to 'standard' (Newtonian, isoenergetic) equilibrium molecular dynamics (EMD). It is well known [3] that the local truncation errors of the Gear methods are smaller but the leapfrog algorithm is much more stable at large time steps for long term energy conservation, despite substantial short term energy fluctuations.

There are in fact several different forms in which the leapfrog algorithm has been implemented. It is unfortunate, as will become clear below, that Verlet [1] popularised the use of a very crude approximation for the velocity. This choice limits the time step which can be used more severely than the error inherent in the positions; the stability is not affected since the updating of the positions need not involve the velocities at all. Beeman [4] showed that energy is conserved much better when the velocity is calculated more accurately and gave an entirely different implementation of the leapfrog algorithm convenient for doing this. He also noted, however, that it is easy to use this higher order velocity approximation within the Verlet implementation.

Downloaded At: 20:01 14 January 2011

We shall be principally concerned here with molecular dynamics algorithms at constant 'temperature'. The main impetus behind the development of these was for nonequilibrium molecular dynamics (NEMD), but we strongly advocate their use for EMD as well. Not only is it almost invariably most convenient to do simulations at an *a priori* fixed temperature, but, as we shall see, the Gaussian isokinetic form of the leapfrog algorithm can be used with even larger timesteps than the isoenergetic form. We shall not consider here the Nosé thermostatted equations of motion [5,6] because, although these have their advantages, Gaussian thermostats have been found to be more computationally efficient [7].

In a recent paper Fincham [8] investigated the possibility of increasing the time step in leapfrog EMD above usual values. With the use of improved estimates of statistical uncertainty, he revealed an *apparent* systematic drift in the thermodynamic properties calculated as the time step was raised before the algorithm becomes unstable. We show here that this drift is caused by the use of Verlet's crude approximation to the velocities and is essentially removed when a more accurate finite difference approximation to the velocity is used. Alternative methods of implementing the isokinetic leapfrog algorithm are compared and reasons given for preferring one of these. We also present the first investigation of the use of large timesteps in NEMD. For simplicity, we restrict attention in this work to pure fluids; the generalisation to mixtures is trivial and some calculations have already been done [9].

2 CONSTANT ENERGY EMD SIMULATIONS

We consider N particles each of mass m in a cubic basic cell of volume Ω with the usual periodic boundary conditions. Isoenergetic EMD is the solution of the coupled differential equations

$$\mathbf{r}_i^{(2)} = \mathbf{a}_i(\{\mathbf{r}_j\}), \quad \mathbf{a}_i = \mathbf{F}_i/m, \tag{1}$$

where $\mathbf{F}_i$ represents the resultant force on particle i and $^{(k)}$ denotes the kth time derivative. Within the leapfrog algorithm, Equations (1) are approximated by

$$\mathbf{r}_i(n+1) = 2\mathbf{r}_i(n) - \mathbf{r}_i(n-1) + h^2\mathbf{a}_i(n) \tag{2}$$

where h is the timestep and the time arguments are given in timestep units. Notice that $\mathbf{r}_i$ is used to denote both the exact solution of the differential equations (1) and its finite difference approximation (2). In contrast, since we mention different approximations to the velocity, these are denoted below by different symbols and not by $\mathbf{r}_i^{(1)}$. It is common in practice [10] to replace Equation (2) by the equivalent pair of equations

$$\mathbf{r}_i(n+1) = \mathbf{r}_i(n) + h\mathbf{w}_i(n), \tag{3}$$

$$\mathbf{w}_i(n) = \mathbf{w}_i(n-1) + h\mathbf{a}_i(n), \tag{4}$$

where the auxiliary variable $\mathbf{w}_i(n)$ is an approximation to the velocity at time $n + \frac{1}{2}$. However, we wish to emphasise that (2) itself is a very stable way of propagating the particle positions irrespective of any consideration of velocities.

Often, following the fashion set by Verlet [1], the velocity at time n is approximated by

$$\mathbf{u}_i(n) = \tfrac{1}{2}[\mathbf{w}_i(n) + \mathbf{w}_i(n-1)]. \tag{5}$$

$\mathbf{u}_i(n)$ is an adequate approximation to the velocity at time n when a small timestep is used, but Equation (2) is of higher order accuracy than (5) and, if we wish to apply (2) at the largest possible time steps, it is necessary to use

$$\boldsymbol{v}_i(n) = \tfrac{1}{6}[2\mathbf{w}_i(n) + 5\mathbf{w}_i(n-1) - \mathbf{w}_i(n-2)]. \tag{6}$$

This is the lowest order expression for the velocity at time n which takes full advantage of the accuracy inherent in the position update (2). It is algebraically equivalent to the velocity approximation given by Beeman's predictor-corrector form [4] of the leapfrog algorithm

It is useful to indicate explicitly the leading truncation errors in the different approximations to the velocity:

$$\mathbf{u}_i(n) = \mathbf{r}_i^{(1)}(n) + \tfrac{1}{6}h^2\mathbf{r}_i^{(3)}(n), \tag{7}$$

$$\mathbf{w}_i(n) = \mathbf{r}_i^{(1)}(n+\tfrac{1}{2}) + \tfrac{1}{24}h^2\mathbf{r}_i^{(3)}(n+\tfrac{1}{2}), \tag{8}$$

$$\mathbf{v}_i(n) = \mathbf{r}_i^{(1)}(n) + \tfrac{1}{12}h^3\mathbf{r}_i^{(4)}(n). \tag{9}$$

Corresponding to Equations (7–9), we can define three approximate temperatures T^u, T^w and T^v by

$$(3N-4)k_BT^v = m\langle \Sigma\, v_i^2\rangle, \tag{10}$$

and its obvious analogues. Here < ... > denotes a time average (assumed equivalent to an ensemble average) and Σ represents a sum over all particles. Again retaining only leading order truncation errors, we have

$$(3N-4)k_B(T^u - T^v) = \tfrac{1}{3}h^2m\langle \Sigma\, \mathbf{r}_i^{(1)}\cdot\mathbf{r}_i^{(3)}\rangle, \tag{11}$$

$$(3N-4)k_B(T^w - T^v) = \tfrac{1}{12}h^2m\langle \Sigma\, \mathbf{r}_i^{(1)}\cdot\mathbf{r}_i^{(3)}\rangle. \tag{12}$$

Downloaded At: 20:01 14 January 2011

Table 1 Comparison of thermodynamic quantities for the constant total energy simulations. The Lennard-Jones (LJ) state point is $\varrho = 0.6$ and $T = 1.52$. All simulations start from near identical configurations for 60 LJ units. U and P are the configurational energy per particle and total pressure in LJ reduced units. The true standard errors based on the Fincham method are shown in brackets corresponding to the least significant digits; h is the time step. T^{u}, T^{w} and T^{v} are the temperatures defined in equations (7–10). The time steps were increased in increments of 0.005. The maximum time step given preceded a value at which the system became unstable. ES denotes the prediction of the Nicolas *et al.* equation of state [11].

h	$-U$	P	T^{u}	T^{w}	T^{v}
ES	3.942	0.749			
0.005	3.957(3)	0.737(12)	1.506(2)	1.508(2)	1.508(2)
0.010	3.950(3)	0.759(11)	1.505(2)	1.512(2)	1.515(2)
0.015	3.950(3)	0.755(16)	1.514(2)	1.530(2)	1.535(2)

Thus, when T^{v} is regarded as essentially exact T^{u} has an error four times as great as the error in T^{w}, so long as higher order truncation terms are negligible. In the context of the leapfrog algorithm T^{v} is as close as we can get to exact; since the leapfrog approximation to the position has error of order h^{4}, velocity approximations obtained from leapfrog positions have a minimum error of order h^{3}.

Results of some isoenergetic simulations for a Lennard-Jones fluid (potential parameters ε and σ) at the same state point as Fincham [8] ($k_{B}T/\varepsilon = 1.52$, $N\sigma^{3}/\Omega = 0.6$, hereafter referred to as state point A) are shown in Table 1. As in *all* simulations reported in this paper, $N = 108$, the potential was cutoff at 2.5σ and tail corrections were added to the thermodynamic properties. Here and subsequently results are quoted in Lennard-Jones reduced units without any particular symbol being used to denote these. The reduced time unit is $\sigma\sqrt{(m/\varepsilon)} \simeq 2$ps when the Lennard-Jones potential is used, with $\varepsilon/k_{B} = 119.8\,K$ and $\sigma = 0.3405$ nm, to model argon. Thus our time step of 0.005 corresponds to 10 fs in the real units used by Fincham. Notice that the difference between the temperatures achieved and the one aimed for is much greater than for the isokinetic simulations discussed below.

The principal points to note about these results are that the algorithm is stable up to a time step of 0.015 (in reduced units) but that, for the same T^{u}, T^{v} and T^{w} are higher at the larger timesteps. In view of Equations (11) and (12), this indicates that $\langle \Sigma\, r_{i}^{(1)} \cdot r_{i}^{(3)} \rangle$ is negative. Presumably previous workers with leapfrog algorithms have used T^{u} under the impression that it might have greater fluctuations than T^{v} but would have the same average value. In fact T^{u} is systematically lower than T^{v}. This does not matter for low time steps such as have tended to be used in the past, but it becomes important for the larger time steps which can be used, without instability, in the isokinetic algorithms to which we now turn our attention.

3 CONSTANT KINETIC ENERGY EMD SIMULATIONS

3.1 Basic algorithms

In Gaussian isokinetic EMD [12], Equation (1) is replaced by

$$\mathbf{r}_{i}^{(2)} = \mathbf{a}_{i}(\{\mathbf{r}_{j}\}) - \lambda(\{\mathbf{r}_{j}^{(1)}\})\mathbf{r}_{i}^{(1)}, \tag{13}$$

$$\lambda = M/2K \tag{14}$$

Downloaded At: 20:01 14 January 2011

where

$$M = \Sigma\, \mathbf{r}_i^{(1)} \cdot \mathbf{F}_i, \tag{15}$$

$$K = \Sigma\, \tfrac{1}{2} m r_i^{(1)2}. \tag{16}$$

The value of the multiplier λ is chosen to ensure that $K^{(1)} = 0$.

Isokinetic MD was originally performed with Gear algorithms [12]. Equations (13–16) cannot be solved with the conventional leapfrog algorithm because of the velocity dependent thermostat 'force'. The first workers to overcome this were Brown and Clarke [13] and a different method was later developed by Heyes [14]. We carry out simulations with both methods. In each, after updating the positions using Equation (3), we first calculate

$$\mathbf{u}_i^*(n) = \mathbf{w}_i(n-1) + \tfrac{1}{2} h \mathbf{a}_i(n) \tag{17}$$

and then incorporate the thermostat term with [14]

$$\mathbf{u}_i(n) = \mathbf{u}_i^*(n) - \tfrac{1}{2} h \lambda(\{\mathbf{u}_j(n)\}) \mathbf{u}_i(n), \tag{18}$$

or [13]

$$\mathbf{u}_i(n) = f^{u*}(n) \mathbf{u}_i^*(n). \tag{19}$$

In order for Equations (18) and (19) to be equivalent,

$$f^{u*}(n) = 1/[1 + \tfrac{1}{2} h \lambda(\{\mathbf{u}_j(n)\})]$$

but at this point Brown and Clarke use the required temperature T, or equivalently the required kinetic energy

$$K_0 = \tfrac{1}{2}(3N - 4) k_B T \tag{20}$$

explicitly at every time step by setting

$$f^{u*}(n) = [K_0 / K^*(n)]^{1/2}, \tag{21}$$

where

$$K^*(n) = \Sigma\, \tfrac{1}{2} m u_i^*(n)^2. \tag{22}$$

Heyes, on the other hand, maintains constant temperature by solving the implicit equations (18) to give

$$\lambda(\{\mathbf{u}_j(n)\}) = M^*(n) / [2K^*(n) - \tfrac{1}{2} h M^*(n)] \tag{23}$$

where

$$M^*(n) = \Sigma\, \mathbf{F}_i(n) \cdot \mathbf{u}_i^*(n). \tag{24}$$

Equation (4) is now replaced by [13]

$$\mathbf{w}_i(n) = [2 f^{u*}(n) - 1] \mathbf{w}_i(n-1) + h f^{u*}(n) \mathbf{a}_i(n) \tag{25}$$

or [14]

$$\mathbf{w}_i(n) = \mathbf{w}_i(n-1) + h[\mathbf{a}_i(n) - \lambda(\{\mathbf{u}_j(n)\}) \mathbf{u}_i(n)]. \tag{26}$$

(Although (25) and (26) are equivalent, their forms are best suited to the respective algorithms indicated.)

We refer to these two methods [13,14] as explicit [Equations (3), (17), (19), (21) and

Downloaded At: 20:01 14 January 2011

(25)] and implicit [Equations (3), (17), (18), (23) and (26)] respectively. For EMD, the latter has been solved explicitly even although it is implicit in the finite difference equation sense, leading to good stability properties. For some NEMD algorithms, the implicit method requires an iterative solution.

For completeness we consider, in addition to the explicit and implicit algorithms, two isokinetic schemes involving rescaling of the 'half time step velocities' $\mathbf{w}_i$. In both cases Equation (3) is retained. Replacing Equation (4) by

$$\mathbf{w}_i(n) = f^{w}(n-1)\mathbf{w}_i(n-1) + h\mathbf{a}_i(n), \tag{27}$$

where

$$f^{w}(n) = [K_0/\Sigma\, \tfrac{1}{2}mw_i(n)^2]^{1/2}, \tag{28}$$

we obtain a more transparent form of the earliest isokinetic scheme proposed by Woodcock in 1971 [15]. Alternatively, we may replace Equation (4) by

$$\mathbf{w}_i^*(n) = \mathbf{w}_i(n-1) + h\mathbf{a}_i(n) \tag{29}$$

followed by

$$\mathbf{w}_i(n) = f^{w*}(\mathrm{n})\mathbf{w}_i^*(n), \tag{30}$$

where

$$f^{w*}(\mathrm{n}) = [K_0/\Sigma\, \tfrac{1}{2}mw_i^*(n)^2]^{1/2}. \tag{31}$$

We denote these two algorithms by R and R^* respectively.

3.2 *Fixing* T^{v} at large timesteps

We shall first consider at some length the explicit and implicit algorithms. It should be emphasised that these algorithms lead directly to the fixing of T^{u}. In view of the remarks above concerning the inaccuracy of $\mathbf{u}_i(n)$ for large time steps, this clearly poses a slight problem in using these algorithms for large time steps. However, as in isoenergetic simulations, it turns out that the ratio $T^{v}:T^{w}:T^{u}$ is constant (for a given time step and state point). Therefore each of T^{v}, T^{w} and T^{u} is fixed but all to slightly different values. For both explicit and implicit simulations, the initial values input were positions $\mathbf{r}_i(0)$ and velocities $\mathbf{w}_i(-1)$ chosen to give $T^{w} = T$. This startup procedure results in $T^{w} = T$ (apart from slight drift) for all times in the implicit algorithm but in the explicit algorithm $T^{u} = T$ from the first velocity rescaling onward. We now consider separately ways of fixing $T^{v} = T$ in the explicit and implicit algorithms.

The only possible way of fixing $T^{v} = T$ in the explicit algorithm is to replace K_0 by some suitable K_0^{u} in Equation (21). The simplest procedure for establishing K_0^{u} is to carry out a separate short run giving $T^{u} = T$ and obtain from this run an approximation T_s^{v} to T^{v}. Then a new run can commence with

$$K_0^{u} = K_0(T/T_s^{v}). \tag{31}$$

This will approximately achieve our aim but since T^{v}/T^{u} is a (weak) function of state point, some iteration may be required if an extremely accurate temperature is needed. A more convenient procedure is to adjust K_0^{u} continuously within one run. Then we use the expression

$$K_0^{u} = K_0(T_R^{u}/T_R^{v})(T/T_R^{v}), \tag{32}$$

Downloaded At: 20:01 14 January 2011

where the subscripts R indicate that the temperatures are running averages weighted toward recent values. (In fact it is satisfactory to use just the temperatures at the immediately previous timestep.) The reason for the extra second factor in (32) compared to (31) is unknown but it seems to be necessary in order to achieve $T^v = T$ precisely.

The only way in which the temperature is introduced into the implicit algorithm is through the initial condition on $\mathbf{w}_i(-1)$. As with the explicit algorithm, a short pilot run can be carried out and then the production run will be started with K_0 replaced by

$$K_0^w = K_0(T_s^w/T_s^v) \tag{33}$$

in Equation (20). Here T_s^w will be very close to T whereas T_s^u in the explicit algorithm [see Equation (31)] was identically equal to T. This form of the implicit algorithm has much smaller temperature drift than the Gear predictor–corrector algorithm used, with much smaller time steps, in the earliest Gaussian isokinetic simulations [12]. (In isokinetic Gear simulations, the temperature is periodically and quite frequently rescaled.)

Nevertheless, in a long run with our implicit algorithm the temperature may drift slightly. This can be counteracted by replacing (14) with

$$\lambda = [M + \alpha(K - K_0)]/2K \tag{34}$$

which slowly relaxes K toward K_0 according to $K^{(1)} = -\alpha(K - K_0)$. Here $\alpha > 0$ is an arbitrary relaxation rate which can either be made very small in order to stabilise a long production run or can be made quite large in order to adjust quickly to the correct temperature in a pilot run. Any temperature adjustment of this type is of course not possible within the framework of the explicit algorithm. There the temperature is absolutely fixed which could mask algorithm instability. Here, the choice of a *small* value of α avoids this possibility.

Specifically for the modified leapfrog algorithm, the velocity update is still given by Equation (26), but (23) is replaced by

$$\lambda(\{\mathbf{u}_j(n)\}) = 2A/[B + (B^2 + h^2\alpha K_0^u A)^{1/2}], \tag{35}$$

$$A = M^*(n) + \alpha(K^*(n) - K_0^u), \tag{36}$$

$$B = 2K^*(n) - \tfrac{1}{2}hM^*(n) + h\alpha K_0^u. \tag{37}$$

Of course, K_0^u is used here rather than K_0 because the algorithm fixes T^u directly. Again, the value for K_0^u could either be obtained from Equation (31) after a short pilot run *of the explicit algorithm* or be adjusted continuously using Equation (32).

The fixing of T^v in the R and R^* algorithms is achieved by obvious analogous procedures to those for the explicit algorithm. Since $T^v - T^w$ is not nearly so great as $T^v - T^u$ the corrections involved are not as important.

Although possible, the fixing of T^v using an algorithm which basically fixes T^u is an unwelcome complication. The reader may well wonder why we do not use an algorithm directly fixing T^v. Tests were made of several algorithms similar to the ones above but aimed at directly producing a prescribed T^v rather than a prescribed T^u. All were found to be unstable even at quite low time steps and we attribute this to the

Downloaded At: 20:01 14 January 2011

Downloaded At: 20:01 14 January 2011

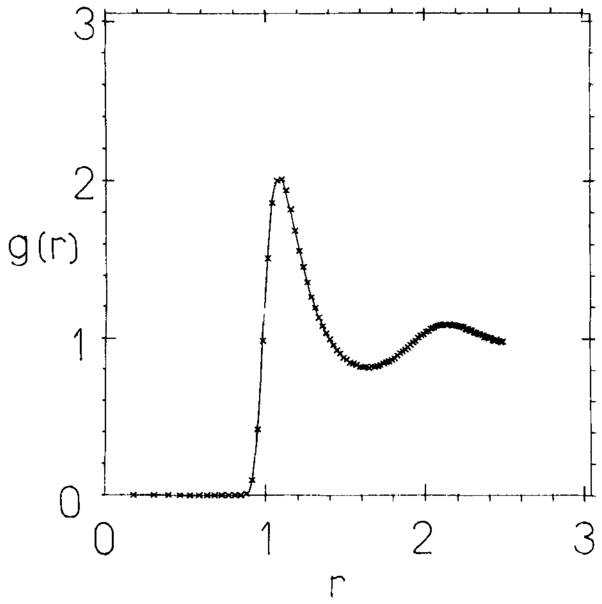


Fig.1A

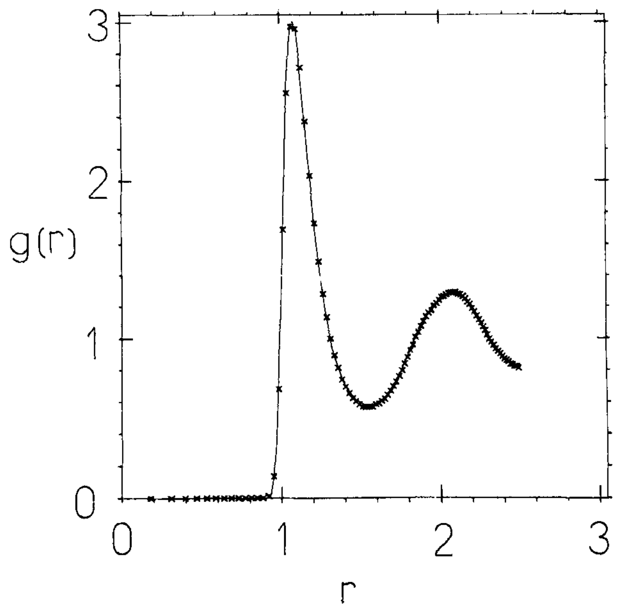


Fig.1B

Figure 1 Radial distribution functions obtained using isokinetic EMD. The duration of the simulations was 180 reduced time units. (A) State point A, line h = 0.005, crosses *h* = 0.02; (B) state point B, line *h* = 0.005, crosses h = 0.03 (see section 2 for definition of reduced units).

Downloaded At: 20:01 14 January 2011

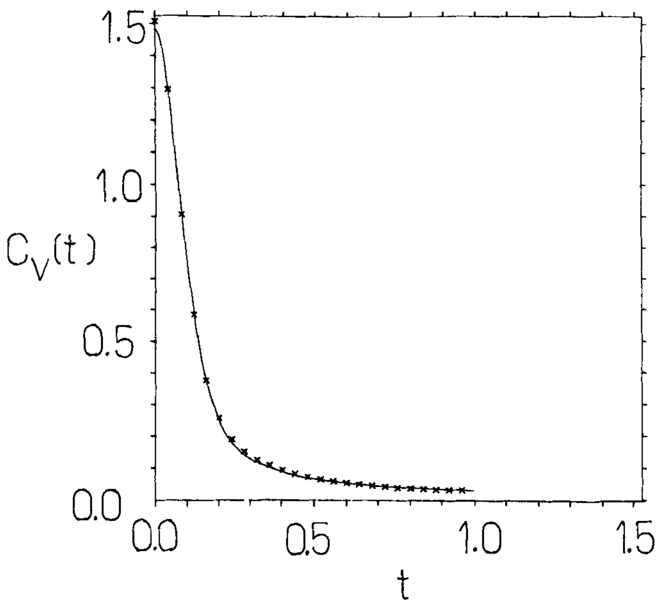


Fig. 2A

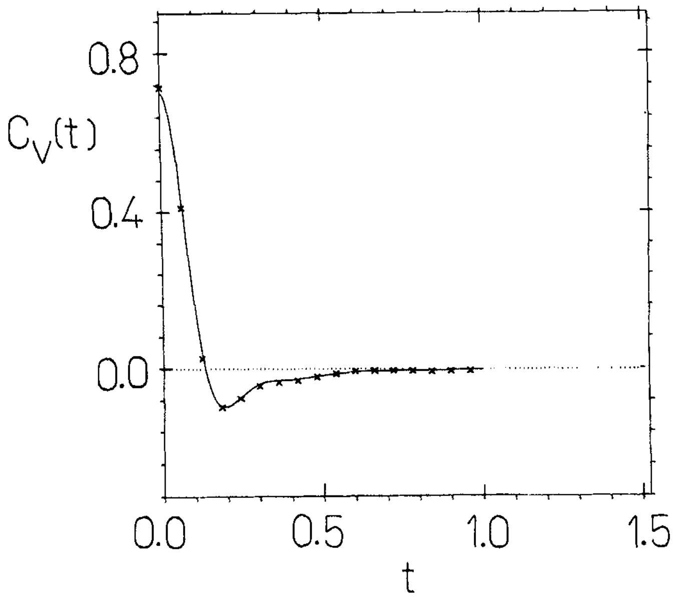


Fig. 2B

Figure 2 Velocity autocorrelation functions. All details as for Figure 1.

inclusion of $\mathbf{r}_i(n - 2)$ in the resultant position updating equation which replaces (2). We have also tried isokinetic modifications of the well known predictor–corrector forms [16] of the ordinary leapfrog algorithm but all of these attempts have also resulted in loss of stability. In particular, Equations (28) of [16] with $c = \frac{1}{2}$ do not work stably for Gaussian isokinetic molecular dynamics contrary to what might be deduced from the unsubstantiated remark following those equations. This is presumably because the stability analysis used in [16] refers specifically to velocity independent forces.

3.3 Results

For the isokinetic calculations we have carried out runs for the widely studied state point $k_B T/\varepsilon = 0.722$, $N\sigma^3/\Omega = 0.8442$ (state B), near the triple point as well as for the Fincham state point (state A). The system was equilibrated to an arbitrary point in phase space, i.e. a particular set of $\mathbf{r}_i(0)$ and $\mathbf{w}_i(-1)$ for one fixed value of time step h, which was used as the basis for the comparisons between the isokinetic algorithms. Each production run was preceded by a short equilibration of approximately 10 of the time steps appropriate to that run. Thus the starting conditions were not exactly the same; since the input data are separated by a half time step, it is almost impossible to achieve identical starting conditions in leapfrog simulations with different time steps.

Before commenting on the relative merits of these schemes, it is worth emphasising that the comparisons are made over a limited region of the accessible phase space and over time spans less than 180 reduced units. Therefore agreement is better than would be obtained by choosing starting conditions that were widely spaced in time for each different time step (e.g. by running each new time step on from the final state of the run at the previous time step). The advantage of the present approach is to allow useful conclusions based on rather limited computer resources.

Using the implicit algorithm, we have evaluated the radial distribution functions (RDFs) and velocity autocorrelation functions (VAFs) for our two state points. These two properties are perhaps the most familiar probes of static and dynamic liquid structure respectively. The RDFs are shown in Figure 1 and the VAFs, obtained by considering all particles and adopting a time origin at each time step, in Figure 2. Each plot shows two results for simulations of equal (real time) duration but considerably different time step. It can be seen from Figures 1 and 2 that RDFs and VAFs are unaffected by the timestep. The approximate temperature fixed to the required value in these runs was T^v but we also noted that it is difficult to detect any difference in these functions when the less accurate approximations to the temperature were held constant.

The better statistics for thermodynamic properties (shown in Tables 2A and 2B for states A and B respectively) make them a stricter test. We present the thermodynamic properties, together with T^u, T^w and T^v for the explicit, implicit, R and R* algorithms at various time steps. The errors (shown in parentheses when non-negligible) are obtained from the formulae given by Fincham [8] but the statistical inefficiency has been obtained by treating subaverages over 200 time steps as independent. This reproduces the statistical inefficiencies of Fincham with reasonable accuracy. Results are obtained up to the maximum possible time steps before instability occurs but this maximum is only approximate due to incrementing the time step by 0.005.

It can be seen that there is no significant difference between the implicit and explicit algorithms provided the same approximation to the temperature is maintained constant with increasing time step. We now concentrate on the implicit algorithm in commenting on the variation of internal energy and pressure with time step. It is

Downloaded At: 20:01 14 January 2011

Table 2A As for Table 1 except that isokinetic simulations were performed on the state point $\varrho = 0.6$ and $T = 1.52$. E and I refer to the explicit and implicit solution of the Gaussian thermostat from equations (17–26). The R rows refer to simulations performed using the velocity rescaling methods of equations (27–31). The rows with the superscript a) refer to computations of duration 180 LJ units, the remainder being for 60 LJ reduced units.

Method	h	$-U$	P	T^u	T^w	T^v
ES		3.942	0.749			
E	0.005	3.944(4)	0.775(13)	1.520	1.522	1.523
E	0.010	3.943(5)	0.802(13)	1.520	1.528	1.530
E	0.015	3.935(4)	0.811(17)	1.520	1.538	1.542
E	0.020	3.929(5)	0.839(16)	1.520	1.554	1.560
I	0.005	3.948(4)	0.765(13)	1.520	1.522	1.522
I	0.010	3.948(4)	0.794(14)	1.519	1.526	1.529
I	0.015	3.935(6)	0.834(20)	1.518	1.535	1.540
I	0.020	3.932(3)	0.823(11)	1.517	1.549	1.556
E	0.005	3.951(3)	0.748(11)	1.518	1.519	1.520
E	0.010	3.949(4)	0.770(15)	1.510	1.518	1.520
E	0.015	3.946(4)	0.776(17)	1.499	1.516	1.520
E	0.020	3.941(3)	0.762(11)	1.482	1.514	1.520
I	0.005	3.944(4)	0.773(14)	1.517	1.519	1.520
I	0.010	3.949(4)	0.777(12)	1.510	1.518	1.520
I	0.015	3.944(5)	0.754(8)	1.499	1.515	1.520
I	0.020	3.949(4)	0.731(8)	1.483	1.513	1.520
I^a	0.005	3.946(2)	0.773(8)	1.518	1.519	1.520
I^a	0.020	3.951(3)	0.743(7)	1.483	1.513	1.520
R	0.010	3.937(7)	0.792(32)	1.513	1.520	1.522
R	0.015	3.940(8)	0.771(30)	1.503	1.520	1.524
R	0.020	3.938(6)	0.754(26)	1.490	1.520	1.527
R*	0.015	3.951(8)	0.738(26)	1.503	1.520	1.525
R*	0.015	3.947(5)	0.747(20)	1.503	1.520	1.525
R*	0.020	3.935(8)	0.746(26)	1.490	1.520	1.527
R*	0.025	3.945(8)	0.762(27)	1.471	1.520	1.528

Downloaded At: 20:01 14 January 2011

found that these properties are independent of time step up to maximum stable time steps of $h_{max} \simeq 0.02$ for state A and $h_{max} \simeq 0.03$ for state B, provided T^v rather than T^u is fixed to T. For state B at $h = 0.03$, T^w has very large fluctuations in the explicit algorithm and there is some slight evidence of drift in the implicit algorithm. The R and R^* algorithms can be used up to the same maximum time steps as the implicit and explicit algorithms.

Clearly T^w is quite close to T^v whilst T^u is consistently lower (by $\sim 5\%$ for $h = 0.025$). The relative values of the different temperatures are consistent for small h with T^v being essentially exact and T^u having an error four times as great as T^w [see Equations (11) and (12)]. For larger h, higher order corrections distort this simple relation. It should be emphasised that T^w is much closer to T^v than is T^u, so that it may sometimes not be considered necessary to eliminate the small discrepancy between T^w and T^v. This gives the simple R and R^* an advantage if one is not prepared to make the adjustments necessary to fix T^v; either R or R^* should certainly be preferred to the explicit algorithm. The implicit algorithm retains the advantage of not introducing the desired temperature explicitly at each time step; thus no instabilities could be masked, an important consideration when trying to use a time step as close as possible to h_{max}.

Table 2B As for Table 2A except that isokinetic simulations were performed on the state point $\varrho = 0.8442$ and $T = 0.722$.

Method	*h*	$-U$	*P*	T^u	T^w	T^v
ES		6.084	0.118			
I	0.005	6.113(3)	−0.002(18)	0.721	0.722	0.723
I	0.010	6.099(4)	0.087(22)	0.720	0.725	0.727
I	0.015	6.093(4)	0.111(21)	0.719	0.731	0.734
I	0.020	6.080(4)	0.177(28)	0.719	0.740	0.746
I	0.025	6.070(7)	0.232(37)	0.721	0.755	0.763
I	0.005	6.106(4)	0.045(21)	0.720	0.721	0.722
I	0.010	6.102(4)	0.046(23)	0.715	0.720	0.722
I	0.015	6.103(4)	0.043(22)	0.708	0.719	0.722
I	0.020	6.113(3)	0.006(17)	0.697	0.717	0.722
I	0.025	6.106(6)	0.020(35)	0.684	0.716	0.723
I	0.030	6.120(3)	−0.060(12)	0.668	0.715	0.723
I[a]	0.005	6.100(2)	0.068(12)	0.720	0.721	0.722
I[a]	0.030	6.120(2)	−0.054(11)	0.668	0.715	0.722
R	0.020	6.107(5)	−0.018(32)	0.702	0.722	0.727
R	0.025	6.118(6)	−0.065(36)	0.690	0.722	0.729
R	0.030	6.113(5)	−0.034(29)	0.676	0.723	0.730
R*	0.030	6.113(5)	−0.011(31)	0.675	0.722	0.730
R*	0.030	6.112(5)	−0.034(28)	0.675	0.722	0.730

The ratio of $h_{\max}$ at state point A to $h_{\max}$ at state point B (near the triple point) is about $\frac{2}{3}$. We anticipated that $h_{\max}$ should be nearly constant *in units of* $\sigma(m/k_BT)^{1/2}$, i.e. $h_{\max}T^{1/2}$ constant in our units. On this basis, we would predict the ratio to be $(0.722/1.52)^{1/2} \simeq 0.69$. Considering the very approximate determination of $h_{\max}$, this is quite consistent with the observed value.

4 NEMD SIMULATIONS

In equilibrium simulations, the choice of external constraint is largely a subjective balance between the convenience of isokinetic dynamics and the simplicity of isoenergetic dynamics. In nonequilibrium simulations, it is essential to remove heat from the system or the work done by the external force will cause the state point to change with time. Therefore isoenergetic equations of motion lose their advantage in simplicity and convenience is a decisive factor in favour of isokinetic methods. Accordingly, only isokinetic nonequilibrium algorithms will be considered.

4.1 Thermal conductivity

The basic differential equation for the NEMD thermal conductivity algorithm of Evans [17] is

$$\mathbf{r}_i^{(2)} = \mathbf{a}_i(\{\mathbf{r}_j\}) + \mathbf{C}_i(\{\mathbf{r}_j^{(1)}\})\cdot\mathbf{F}_e - \lambda(\{\mathbf{r}_j^{(1)}\})\mathbf{r}_i^{(1)}, \quad (39)$$

where

Downloaded At: 20:01 14 January 2011

$$\mathbf{C}_i(\{\mathbf{r}_j^{(1)}\}) = C_i^K(\{\mathbf{r}_j^{(1)}\})\mathbf{1} + \mathbf{C}_i^{\phi}, \tag{40}$$

$$C_i^K(\{\mathbf{r}_j^{(1)}\}) = \tfrac{1}{2}mr_i^{(1)2} - K/N \tag{41}$$

$$\mathbf{C}_i^{\phi} = \frac{1}{2}\left[\sum_j (\phi_{ij}\mathbf{1} - \mathbf{r}_{ij}\mathbf{F}_{ij}) - \frac{1}{2N}\sum_j\sum_k (\phi_{jk}\mathbf{1} - \mathbf{r}_{jk}\mathbf{F}_{jk})\right] \tag{42}$$

where λ is given by Equations (34), except that now

$$M = \Sigma\, \mathbf{r}_i^{(1)}\cdot[\mathbf{F}_i + m\mathbf{C}_i(\{\mathbf{r}_j^{(1)}\})\cdot\mathbf{F}_e], \tag{43}$$

rather than being given by Equation (15). K is still given by Equation (16), $\mathbf{r}_{ij} = \mathbf{r}_j - \mathbf{r}_i$, ϕ_{ij} is the interaction energy of the pair (i, j) and $\mathbf{F}_{ij}$ is the force on i due to j.

The thermal conductivity κ is given by

$$\kappa = T^{-2} \lim_{F_e \to 0} \langle \mathbf{J}_q \cdot \mathbf{F}_e \rangle / F_e^2 \tag{44}$$

where the heat current $\mathbf{J}_q$ is defined by [18]

$$\Omega\mathbf{J}_q = \frac{1}{2}\sum_i \left[\mathbf{r}_i^{(1)}(mr_i^{(1)2} + \sum_j \phi_{ij}) - \mathbf{r}_i^{(1)}\cdot\sum_j \mathbf{r}_{ij}\mathbf{F}_{ij}\right]. \tag{45}$$

In implementing an implicit leapfrog form of this algorithm, Equation (17) is replaced by

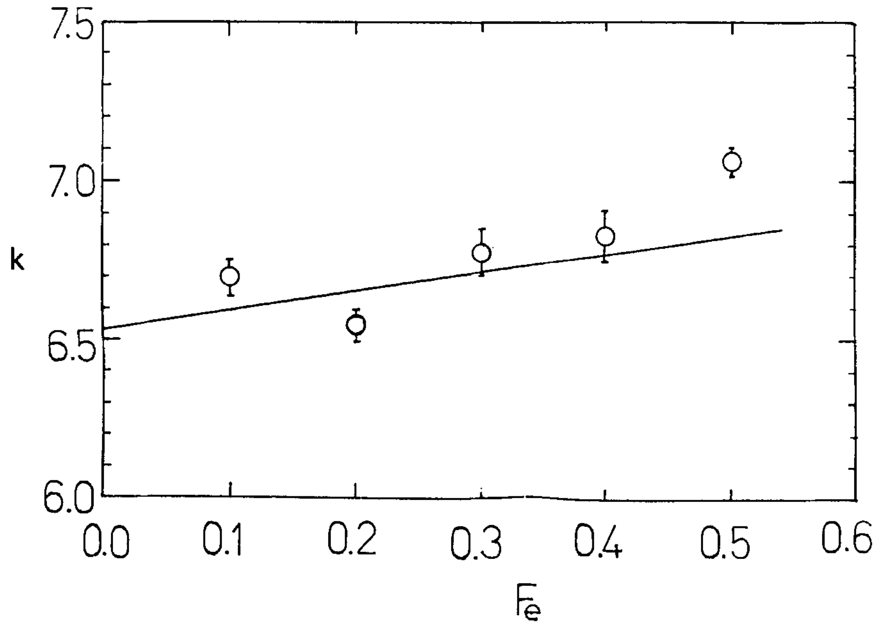


Figure 3 Results of the NEMD thermal conductivity algorithm for state point B. The maximum length of the simulations was 3200 reduced time units for $F_e = 0.1$, with substantially shorter runs for higher F_e values, and the time step was 0.02.

Downloaded At: 20:01 14 January 2011

$$\mathbf{u}_i^*(n) = \mathbf{w}_i(n-1) + \tfrac{1}{2}h[\mathbf{a}_i(n) + \mathbf{C}_i^{\phi}(n)\cdot\mathbf{F}_e] \tag{46}$$

and (18) is replaced by

$$\mathbf{u}_i(n) = \mathbf{u}_i^*(n) + \tfrac{1}{2}h[C_i^K(\{\mathbf{u}_j(n)\})\cdot\mathbf{F}_e - \lambda(\{\mathbf{u}_j(n)\})\mathbf{u}_i(n)]. \tag{47}$$

Equation (47) is solved by iteration. The initial guess is to replace $\{\mathbf{u}_j(n)\}$ by $\{\mathbf{u}_j^*(n)\}$ and three iterations are found to give adequate convergence. Note that once $\mathbf{w}_i(n)$ is evaluated using

$$\mathbf{w}_i(n) = \mathbf{w}_i(n-1) + h[\mathbf{a}_i(n) + \mathbf{C}_i^{\phi}(n)\cdot\mathbf{F}_e - \lambda(\{\mathbf{u}_j(n)\})\mathbf{u}_i(n)] \tag{48}$$

it is possible to obtain $\mathbf{v}_i(n)$ and hence $\mathbf{J}_q(\{\mathbf{v}_i(n)\})$ rather than the less accurate $\mathbf{J}_q(\{\mathbf{u}_i(n)\})$. However, it is $\mathbf{J}_q(\{\mathbf{u}_i(n)\})$ which is implicit in the adiabatic dissipation for the algorithm.

Results are shown in Figure 3 above for various values of F_e at state point B. The linearly extrapolated result for $F_e = 0$ is $\kappa = 6.53$, in excellent agreement with the original result $\kappa = 6.55$ of Evans obtained using a Gear algorithm [17]. However, the present results show considerably more scatter than those of Evans.

4.2 Shear viscosity

Steady isokinetic homogeneous shear flow can be achieved by a simple boundary driven algorithm:

$$\mathbf{r}_i^{(2)} = \mathbf{a}_i(\{\mathbf{r}_j\}) - \lambda(\{\mathbf{v}_j\})\mathbf{v}_i, \tag{49}$$

where

$$\mathbf{v}_i = \mathbf{r}_i^{(1)} - \gamma y_i\hat{\mathbf{x}}. \tag{50}$$

Here γ is the rate of change of the imposed velocity in the x-direction with y. In place of the usual periodic boundary conditions, Equations (49) are used in conjunction with shearing periodic boundaries. The total "laboratory frame" kinetic energy then includes a contribution due to the shearing motion in addition to the local thermal kinetic energy which determines the temperature.

It is for this reason that the $\mathbf{r}_i^{(1)}$ of the previous algorithms is replaced by the peculiar velocity $\mathbf{v}_i$ in Equation (49). The kinetic energy which is held constant is now

$$K = \Sigma\tfrac{1}{2}mv_i^2, \tag{51}$$

rather than that defined by Equation (16). In order to achieve this it is necessary to calculate λ from

$$\lambda = [M - \gamma L + \alpha(\dot{K} - K_0)]/2K \tag{52}$$

where, now,

$$M = \Sigma\, \mathbf{v}_i\cdot\mathbf{F}_i, \tag{53}$$

$$L = \Sigma\, mv_{xi}v_{yi}. \tag{54}$$

Here we have explicity included the temperature adjustment parameter α but this may be set to zero if desired.

It is often convenient to replace Equations (49) with two sets of first order equations

$$\mathbf{r}_i^{(1)} = \mathbf{v}_i + \gamma y_i\hat{\mathbf{x}}, \tag{55}$$

Downloaded At: 20:01 14 January 2011

$$v_i^{(1)} = \mathbf{a}_i - \gamma v_{yi}\hat{\mathbf{x}} - \lambda(\{\mathbf{v}_j\})\mathbf{v}_i. \tag{56}$$

For historical reasons [2,19] these are known as the Sllod equations. For steady shear, they are entirely equivalent to Equations (49) but the Sllod equations have advantages for frequency dependent shear. In conjunction with either Equations (49) or Equations (55) and (56) the initial condition must be chosen so that the average peculiar velocity is zero and $K = K_0$.

The shear viscosity is given by

$$\eta(\gamma) = -\langle P_{xy}\rangle/\gamma \tag{57}$$

where P_{xy} is the xy component of the pressure tensor $\mathbf{P}$ given by [18]

$$\Omega\mathbf{P} = \sum_i \left[m\mathbf{v}_i\mathbf{v}_i - \frac{1}{2}\sum_j \mathbf{r}_{ij}\mathbf{F}_{ij} \right]. \tag{58}$$

The homogeneous shear algorithms are the only synthetic nonequilibrium algorithms where nonlinear results are believed to have any significance – hence the form of Equation (57) as opposed to Equation (44). The usual linear or Newtonian shear viscosity is obtained upon taking the limit $\gamma \to 0$.

We now proceed to give two ways of implementing the homogeneous shear algorithm in an implicit leapfrog form. For consistency with the previous notation, we denote three different finite difference approximations to the peculiar velocity $\mathbf{v}_i$ by $\mathbf{U}_i$, $\mathbf{V}_i$ and $\mathbf{W}_i$. These correspond respectively to the approximations $\mathbf{u}_i$, $\mathbf{v}_i$ and $\mathbf{w}_i$ for $\mathbf{r}_i^{(1)}$. The two forms of the leapfrog algorithm are most easily related to the two different forms of the shearing equations and so we identify them as the one second order (1S) and two first order (2F) algorithms respectively.

The 1S algorithm is based on Equation (49) and, as for the EMD algorithm, includes Equations (3) and (17). Equation (18) is replaced by

$$\mathbf{U}_i(n) = \mathbf{U}_i^*(n) - \tfrac{1}{2}h\lambda(\{\mathbf{U}_j(n)\})\mathbf{U}_i(n), \tag{59}$$

where

$$\mathbf{U}_i(n) = \mathbf{u}_i(n) - \gamma y_i(n)\hat{\mathbf{x}}. \tag{60}$$

The solution of the implicit Equations (59) is given by the following obvious analogues of Equations (35–37):

$$\lambda(\{\mathbf{U}_j(n)\}) = 2A/[B + (B^2 + h^2\alpha K_0^U A)^{1/2}], \tag{61}$$

$$A = M^*(n) - \gamma L^*(n) + \alpha(K^*(n) - K_0^U), \tag{62}$$

$$B = 2K^*(n) - \tfrac{1}{2}hM^*(n) + h\alpha K_0^U. \tag{63}$$

Finally, in place of (26), we have

$$\mathbf{w}_i(n) = \mathbf{w}_i(n-1) + h[\mathbf{a}_i(n) - \lambda(\{\mathbf{U}_j(n)\})\mathbf{U}_i(n)]. \tag{64}$$

With this method, as for the EMD implicit algorithm, we avoid iteration. We also avoid an ambiguity which occurs in the 2F algorithm.

Nevertheless, it seems that the 2F algorithm which we now describe is more stable in practice. Here the entire algorithm is cast in terms of the peculiar velocities $\mathbf{U}_i$ and $\mathbf{W}_i$. Let us suppose that we know $\{\mathbf{r}_i(n)\}$ and $\{\mathbf{W}_i(n-1)\}$ and consider the steps necessary to obtain $\{\mathbf{W}_i(n)\}$. The first step is to calculate

$$\mathbf{U}_i^*(n) = \mathbf{W}_i(n-1) + \tfrac{1}{2}h\mathbf{a}_i(n) \tag{65}$$

Downloaded At: 20:01 14 January 2011

followed by

$$\mathbf{U}_i(n) = \mathbf{U}_i^*(n) - \tfrac{1}{2}h[\gamma U_{yi}(n)\hat{\mathbf{x}} + \lambda(\{\mathbf{U}_j(n)\})\mathbf{U}_i(n)], \tag{66}$$

which must be solved iteratively. The iteration procedure is similar to that for Equations (47): The initial guess is to replace $\{\mathbf{U}_j(n)\}$ by $\{\mathbf{U}_j^*(n)\}$ and again three iterations are found to give adequate convergence. Note that in the iteration procedure it is essential to calculate the y-component before the x-component. Finally, $\mathbf{W}_i(n)$ is evaluated using

$$\mathbf{W}_i(n) = \mathbf{W}_i(n-1) + h[\mathbf{a}_i(n) - \gamma U_{yi}(n)\hat{\mathbf{x}} - \lambda(\{\mathbf{U}_j(n)\})\mathbf{U}_i(n)]. \tag{67}$$

We are now faced with advancing the positions to $\{\mathbf{r}_i(n+1)\}$. From Equation (55) it is clear that this should be done with an equation of the form

$$\mathbf{r}_i(n+1) = \mathbf{r}_i(n) + h[\mathbf{W}_i(n) + \gamma Y_i(n)\hat{\mathbf{x}}]. \tag{68}$$

Since $\mathbf{W}_i(n)$ is an approximation to the peculiar velocity at time $n + \frac{1}{2}$, $Y_i(n)$ should ideally be an approximation to y_i at that same time. Given that the finite difference scheme only provides y_i at full time steps, a natural choice would be

Table 3A As for Table 2A except that the effects of an applied shear rate, γ, are investigated. The equations of motion for inducing shear flow are implemented using the 2F and 1S algorithms described in the text. The implicit method for implementing the Gaussian isokinetic multipliers was used. The state point is $\varrho = 0.6$, $T = 1.52$ and $\gamma = 1.0$. All runs were for 60 LJ time units.

Method	h	$-U$	P	η	T^u	T^w	T^v
2F	0.005	3.913(4)	0.855(13)	0.774(15)	1.513	1.515	1.516
2F	0.010	3.909(4)	0.838(17)	0.811(13)	1.507	1.515	1.518
2F	0.015	3.910(3)	0.840(13)	0.807(12)	1.497	1.515	1.520
2F	0.020	3.913(4)	0.833(15)	0.790(12)	1.482	1.515	1.522
1S	0.005	3.910(4)	0.871(15)	0.801(17)	1.518	1.520	1.521
1S	0.010	3.908(3)	0.852(14)	0.805(16)	1.514	1.521	1.524
1S	0.015	3.913(4)	0.858(18)	0.801(15)	1.506	1.523	1.529
1S	0.020	3.904(4)	0.871(13)	0.788(21)	1.497	1.529	1.538

Table 3B As for Table 3A except that the state point is $\varrho = 0.8442$, $T = 0.722$. The rows denoted by the superscript a) indicate simulations performed for 180 LJ time units; the remainder were conducted for 60 LJ time units.

Method	h	$-U$	P	η	T^u	T^w	T^v
2F	0.005	5.899(6)	1.030(31)	2.061(27)	0.720	0.721	0.722
2F	0.010	5.887(5)	1.105(27)	2.117(28)	0.715	0.720	0.722
2F	0.015	5.902(6)	1.013(31)	2.088(29)	0.707	0.719	0.722
2F	0.020	5.900(5)	1.030(27)	2.063(24)	0.696	0.717	0.722
2F	0.025	5.901(5)	1.020(25)	2.094(26)	0.682	0.716	0.723
2F[a]	0.005	5.897(3)	1.043(18)	2.072(17)	0.720	0.721	0.722
2F[a]	0.025	5.901(3)	1.011(17)	2.096(17)	0.682	0.716	0.722
1S	0.005	5.896(4)	1.056(21)	2.051(22)	0.720	0.722	0.722
1S	0.010	5.897(5)	1.049(25)	2.039(25)	0.716	0.721	0.723
1S	0.015	5.890(5)	1.077(25)	2.115(23)	0.709	0.720	0.724
1S	0.020	5.896(5)	1.038(27)	2.052(25)	0.699	0.720	0.725
1S	0.025	5.895(6)	1.058(32)	2.121(28)	0.687	0.721	0.728

Downloaded At: 20:01 14 January 2011

$$Y_i(n) = \tfrac{1}{2}[y_i(n) + y_i(n+1)], \tag{69}$$

but in this work the simpler and apparently very crude approximation

$$Y_i(n) = y_i(n) \tag{70}$$

has been used. No difference could be detected in short test runs between results obtained using Equations (69) and (70) respectively.

We will now consider the extent to which the homogeneous shear algorithms can be used with large time steps. The fixing of T^v rather than T^u is done in the same way as for isokinetic equilibrium runs. However, the effect of small changes in temperature is not so important at high shear rates as in equilibrium because of the dominating effect on the molecular trajectories of the imposed shear. Tables 3A and 3B present the thermodynamic properties and shear viscosity based on the above 2F and 1S algorithms. The shear rate chosen is unity in reduced units. This generates $\sim 50\%$ shear thinning at state point B but insignificant shear thinning at state point A. Presumably, since shear thinning indicates significant deviation from the equilibrium trajectories, the maximum time step for stability of the algorithm should only be reduced if there is significant shear thinning. This is indeed observed: The maximum time step for state B is reduced to 0.025 (from 0.03) but the maximum time step for state A remains at 0.02.

The shear viscosity and thermodynamic properties are statistically indistinguishable for time steps up to 0.02 at both state points. However, apart from blowing up beyond a certain time step the 1S algorithm has some drift even at very small timesteps. This can be counteracted with a moderately large α temperature adjustment

Downloaded At: 20:01 14 January 2011

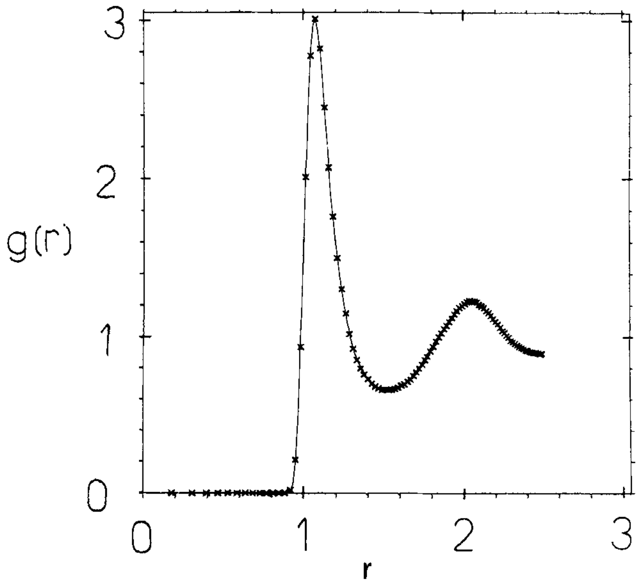


Figure 4 Radial distribution function for state point B under homogeneous shear $\gamma = 1.0$. Otherwise as for Figure 1(B) except that crosses are for h = 0.025, the shear algorithm being unstable at h = 0.03.

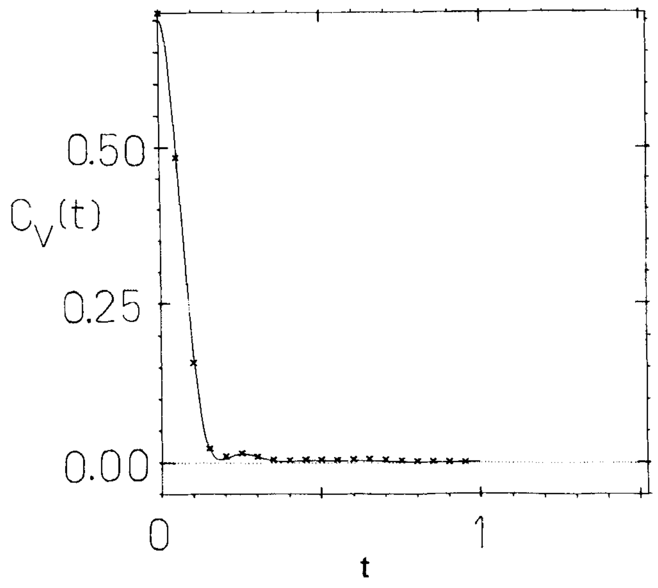


Figure 5 Peculiar velocity autocorrelation function for state point B under homogeneous shear $\gamma = 1.0$ (averaged over all three components even although the system is now anisotropic). Otherwise as for Figure 4.

but this removes the advantage of the implicit over the explicit algorithm. Therefore the 2F algorithm which does not have this drift is to be preferred. Figures 4 and 5 show that there is no noticeable time step dependence of the RDF and VAF for state point B.

4.3 Self diffusion

We write the equations of motion for the colour current algorithm of Evans et al. [12] in the following constant current form:

$$\mathbf{r}_i^{(1)} = \mathbf{v}_i + (-1)^i I\hat{\mathbf{z}}, \tag{71}$$

$$\mathbf{v}_i^{(1)} = \tilde{\mathbf{a}}_i - \lambda(\{\mathbf{v}_j\})\mathbf{v}_i. \tag{72}$$

Here the 'colour' of particle i is $(-1)^i$, $NI = \Sigma\,(-1)^i z_i^{(1)}$ is the 'colour' current,

$$\tilde{\mathbf{a}}_i = \mathbf{a}_i - (-1)^i \Theta\hat{\mathbf{z}}, \tag{73}$$

$$\Theta = N^{-1}\sum_j (-1)^j a_{zj}, \tag{74}$$

and λ is given by Equations (34), (51) and (53). Again these two first order sets of equations can be replaced by a single set of second order equations

$$\mathbf{r}_i^{(2)} = \tilde{\mathbf{a}}_i - \lambda(\{\mathbf{v}_j\})\mathbf{v}_i. \tag{75}$$

These equations of motion must be solved with initial conditions of the correct peculiar kinetic energy and separate zero average peculiar velocities for both 'colours' of particle.

Downloaded At: 20:01 14 January 2011

Fig. 6A

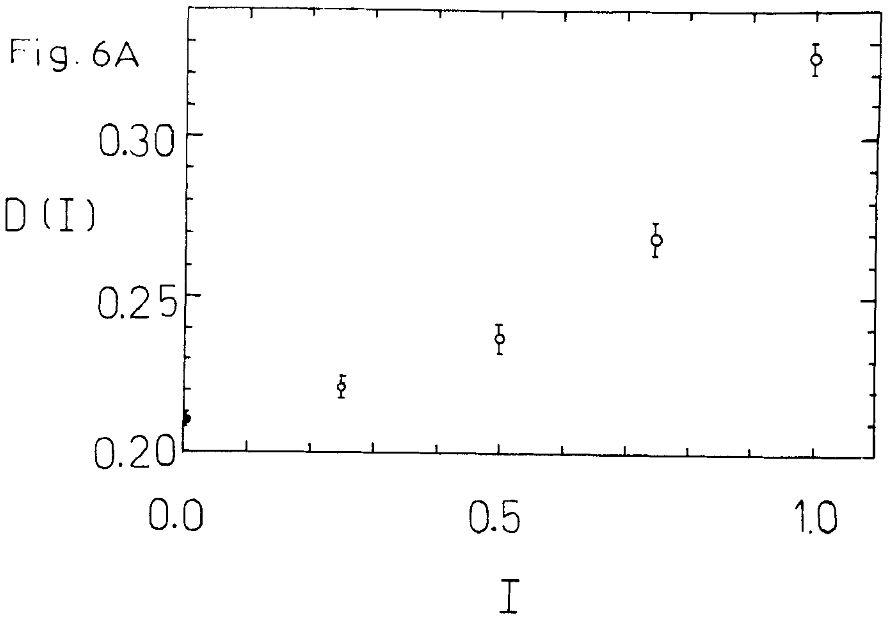


Fig. 6B

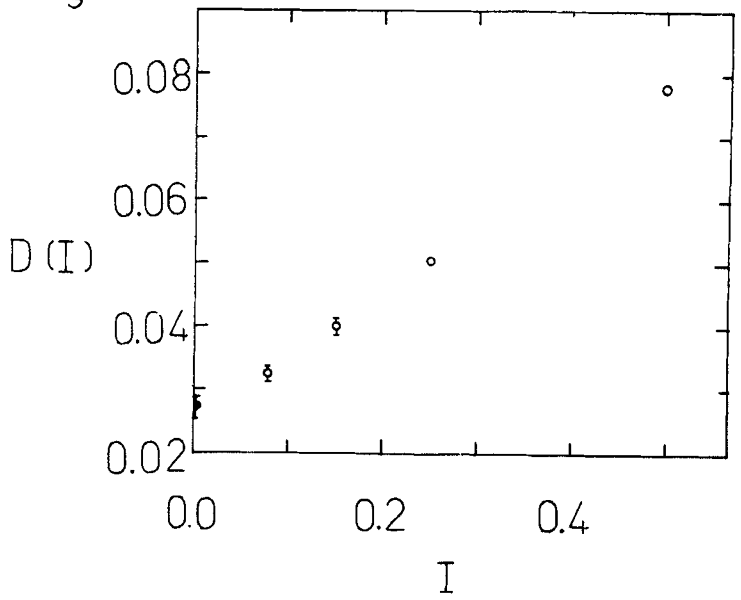


Figure 6 NEMD results for the self diffusion coefficient for (A) state point A and (B) state point B. The results shown on the $I = 0$ axis are obtained by integrating the velocity autocorrelation functions from Figure 2. The maximum lengths of the NEMD simulations are 1800 reduced time units for the lowest values of I with substantially shorter runs for higher I values. The time steps were 0.015 or 0.02 for state point A and 0.025 for state point B.

Downloaded At: 20:01 14 January 2011

The motion is like that of positive and negative charges in a uniform electric field except that these particles have no mutual Coulomb interactions. The self diffusion coefficient is given by

$$D = -\frac{N-1}{N} k_B T \lim_{I \to 0} I/\langle \Theta \rangle, \tag{76}$$

this being essentially the flux divided by the driving force although in this algorithm, unlike the previous ones, the flux is fixed and the steady state average of the driving force necessary to maintain that flux comes out from the simulation.

There are two forms of the leapfrog algorithm equivalent to the 1S and 2F shear algorithms. We do not give details of these since they closely follow the shear case. The diffusion algorithms are simpler because the terms by which laboratory and peculiar velocities differ are constants. This removes the ambiguity of the 2F algorithm and makes the two algorithms algebraically identical. Therefore it is no surprise that the 1S and 2F algorithms for diffusion behave similarly, neither having the drift problem seen for the 1S shear algorithm. Test runs showed that 1S and 2F gave statistically identical results, as expected.

Figure 6 shows results for the diffusion algorithm at the state points A and B. Extrapolation to $I = 0$ gives good agreement with the diffusion coefficient obtained from the mean square displacement in an equilibrium simulation, i.e. by integrating the VAFs shown in Figure 2. These equilibrium simulation results for D are shown on the $I = 0$ axis of Figure 6.

Downloaded At: 20:01 14 January 2011

5 CONCLUSIONS

We have shown that isokinetic EMD and NEMD equations of motion can be successfully integrated using modified leapfrog algorithms. Three types of test have been performed: changes of time step, comparisons between EMD and NEMD results, and comparisons with previous published results. Stability and accuracy of isokinetic leapfrog results is maintained up to time steps much higher than is the case with Gear algorithms and also somewhat higher than for isoenergetic leapfrog. This improves computational efficiency by increasing the rate of coverage of phase space per unit CPU time. The maximum time step possible depends on the state point studied. It is quite strongly affected by temperature and by strong external fields and it is expected that there may also be a weaker density dependence.

We gave a detailed comparison of the implicit, explicit, R and R* methods for EMD but only considered the implicit method for NEMD. We personally prefer the implicit method because it does not explicitly fix the temperature at each time step, but it would be possible to use analogues of the explicit, R and R* algorithms for NEMD. Where iteration was necessary in the implicit NEMD algorithms it would also be necessary in the corresponding alternative algorithms. We find that the homogeneous NEMD algorithms are inferior to EMD in terms of computer time required to calculate the linear transport coefficient. Only in the case of homogeneous shear, where finite driving forces are meaningful, or for some special state points where long time tail effects are important, do we recommend use of the NEMD algorithms other than as an occasional independent check on EMD results.

For the largest time steps and near equilibrium it is important that the system temperature is obtained accurately. Previous simulations where lower order accuracy has been used for the velocities are generally still quite valid because they have used small time steps but we believe that use of $\mathbf{v}_i$ to approximate the velocities should be preferred in all future leapfrog molecular dynamics simulations. If adjustment of T^v to the correct value is avoided in interests of simplicity then the best algorithms to use will be implicit, R and R^* which fix T^w. The explicit algorithm is to be avoided since it fixes T^u and is in every respect inferior to the R algorithm though the latter predated it by 13 years.

There may be circumstances in which highly accurate single trajectories are needed. Then we would advocate the use of Gear algorithms which are clearly superior in terms of local truncation error. Even at low time steps, such as 0.005, the energy fluctuations in the isoenergetic leapfrog molecular dynamics and kinetic energy fluctuations in isokinetic leapfrog molecular dynamics are quite large.

References

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

[2] D. J. Evans and G. P. Morriss, "Non-Newtonian molecular dynamics", *Comp. Phys. Rep.* **1,** 298 (1984).

[3] See, for example, H. J. C. Berendsen and W. F. Van Gunsteren, "Practical algorithms for dynamical simulations", Enrico Fermi International Summer School of Physics, Volume 97: Molecular Dynamics Simulations of Statistical Mechanical Systems, edited by G. Ciccotti and W. G. Hoover, North Holland, Amsterdam, 1987.

[4] D. Beeman, "Some multistep methods for use in molecular dynamics calculations", *J. Comp. Phys.* **20,** 130 (1976).

[5] S. Nosé, "A molecular dynamics method for simulations in the canonical ensemble", *Mol. Phys.* **52,** 255 (1984); "A unified formulation of the constant temperature molecular dynamics method", *J. Chem. Phys.* **81,** 511 (1984).

[6] W. G. Hoover, "Canonical dynamics: Equilibrium phase-space distributions", *Phys. Rev. A* **31,** 1695 (1985).

[7] D. J. Evans and B. L. Holian, "The Nosé-Hoover thermostat", *J. Chem. Phys.* **83,** 4069 (1985).

[8] D. Fincham, "Choice of timestep in molecular dynamics simulation", *Comp. Phys. Comm.* **40,** 263 (1986).

[9] D. MacGowan, "Time correlation functions in a binary liquid mixture", *Phys. Rev. A* **36,** 1367 (1987).

[10] D. Fincham, "Programs for the molecular dynamics simulation of liquids. I. Spherical molecules with short-ranged interactions", *Comp. Phys. Comm.* **21,** 247 (1980).

[11] J. J. Nicolas, K. E. Gubbins, W. B. Streett and D. J. Tildesley, "Equation of state of the Lennard-Jones fluid", *Mol. Phys.* **37,** 1429 (1979).

[12] D. J. Evans, W. G. Hoover, B. H. Failor, B. Moran and A. J. C. Ladd, "Nonequilibrium molecular dynamics via Gauss's principle of least constraint", *Phys. Rev. A* **28,** 1016 (1983).

[13] D. Brown and J. H. R. Clarke, "A comparison of constant energy, constant temperature and constant pressure ensembles in molecular dynamics simulations of atomic liquids", *Mol. Phys.* **51,** 1243 (1984).

[14] D. M. Heyes, "The nature of extreme shear thinning in simple liquids", *Mol. Phys.* **57,** 1265 (1986).

[15] L. V. Woodcock, "Isothermal molecular dynamics calculations for liquid salts", *Chem. Phys. Lett.* **10,** 257 (1971).

[16] A. R. Janzen and J. W. Leech, "Periodic multistep methods in molecular dynamics", *Comp. Phys. Comm.* **32,** 349 (1984).

[17] D. J. Evans, "Homogeneous NEMD algorithm for thermal conductivity – application of non-canonical linear response theory", *Phys. Lett. A* **91,** 457 (1982).

[18] J. H. Irving and J. G. Kirkwood, "The statistical mechanical theory of transport processes. IV. The equations of hydrodynamics", *J. Chem. Phys.* **18,** 817 (1950).

[19] D. J. Evans, "Nonequilibrium molecular dynamics", Enrico Fermi International Summer School of Physics, Volume 97: Molecular Dynamics Simulation of Statistical Mechanical Systems, edited by G. Ciccotti and W. G. Hoover, North Holland, Amsterdam, 1987.

Downloaded At: 20:01 14 January 2011