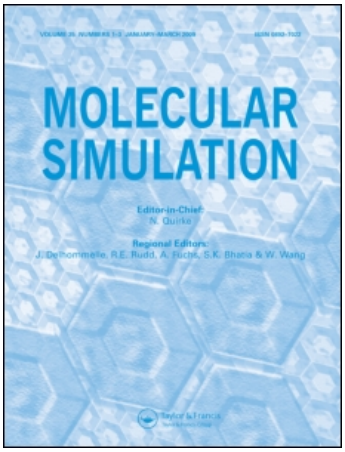

Molecular Simulation
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t713644482

A Parallel Algorithm for Nonequilibrium Molecular Dynamics Simulation of Shear Flow on Distributed Memory Machines

David P. Hansen[a]; Denis J. Evans[a]
[a] Research School of Chemistry, Australian National University, Canberra, Australia

To cite this Article Hansen, David P. and Evans, Denis J.(1994) 'A Parallel Algorithm for Nonequilibrium Molecular Dynamics Simulation of Shear Flow on Distributed Memory Machines', Molecular Simulation, 13: 6, 375 — 393
To link to this Article: DOI: 10.1080/08927029408022000
URL: http://dx.doi.org/10.1080/08927029408022000

Molecular Simulation, 1994, Vol. 13, pp. 375-393
Reprints available directly from the publisher
Photocopying permitted by license only

A PARALLEL ALGORITHM FOR NONEQUILIBRIUM MOLECULAR DYNAMICS SIMULATION OF SHEAR FLOW ON DISTRIBUTED MEMORY MACHINES

DAVID P. HANSEN and DENIS J. EVANS

Research School of Chemistry, Australian National University, Canberra, A. C. T. 0200, Australia

(Received May 1994, accepted June 1994)

An algorithm is described which allows Nonequilibrium Molecular Dynamics (NEMD) simulations of a fluid undergoing planar Couette flow (shear flow) to be carried out on a distributed memory parallel processor using a (spatial) domain decomposition technique. Unlike previous algorithms, this algorithm uses a co-moving, or Lagrangian, simulation box. Also, the shape of the simulation box changes throughout the course of the simulation. The algorithm, which can be used for two or three dimensional systems, has been tested on a Fujitsu AP1000 Parallel computer with 128 processors.

KEY WORDS: Shearflow, NEMD, distributed memory.

1. INTRODUCTION

There have been an increasing number of papers in the literature dealing with strategies for carrying out Molecular Dynamics (MD) simulations on distributed memory machines [1–12]. These strategies include systolic loop algorithms, replicated data algorithms and domain decomposition algorithms.

Systolic loop algorithms [5, 6, 8] cycle all the data through every processor, calculating the forces on particles as the data is passed. Replicated data algorithms [5] assume each processor has a copy of the positions and momenta of all particles in the system, while assigning each particle to an individual processor. The forces on each particle can then be calculated by that processor. While these algorithms can be very good if small systems are being simulated, they quickly become expensive in terms of memory and communication as the size of the system increases. Scalability is an important attribute in algorithms for multiprocessor machines. This allows both the system size and the machine size to be increased, without significant time and communication penalties and without having to make significant changes to the code.

Domain decomposition algorithms subdivide the simulation box into sub-areas with each processor being assigned a sub-area of the simulation box [2, 3, 5–7, 9–12]. Each processor is then responsible for calculating the forces on particles within its sub-area. If the interparticle forces are short ranged, then particles will only interact with other particles in neighbouring sub-areas. Therefore some particle information will need to be sent to 'neighbouring' processors. However the communication

patterns used in this case are well known since each processor will be limited to near-neighbour communication.

The algorithm described in this paper is primarily for use in simulating fluid systems where the interparticle forces are short ranged. Long ranged forces, such as gravitational and coulombic forces, are excluded. In systems which use longranged potentials systolic loop or replicated data algorithms may be more useful. Simulation strategies for solids, where particle diffusion is absent, are much easier.

For the simulation of equilibrium systems using the domain decomposition technique, the most effective way of dividing the simulation box is into cubes, for a three dimensional system [11, 12], or squares for a two-dimensional fluid. With the simulation box divided into cubes and each processor assigned a cubic sub-area, the surface area to volume ratio is at a minimum. This minimises communication between processors.

Previously, algorithms for simulating fluids undergoing shear flow using a domain decomposition technique have been proposed for three dimensional systems [7]. The simulation box is subdivided so that each processor is assigned an ortho-rhombic column, whose long axis is parallel to the streaming velocity. The shear is then maintained using the Lees-Edwards periodic boundary conditions [13]. Using this method the communication patterns are unchanged from those of an equilibrium system which has been subdivided into columns. However the use of columns rather than cubes increases the inter processor communication that results from particle diffusion transverse to the streaming velocity. The high transverse surface area of these columns exacerbates this problem.

However, if the system is divided into cubes, then particle convection which results from the shear flow would require even higher communication rates than those for the column scheme. This is because particle displacement caused by diffusion is only proportional to the square root of time, whereas convective displacement is proportional to time.

The algorithm presented in this paper solves these problems by using a co-moving, or Lagrangian, simulation box which also changes shape during the simulation. This gives us an algorithm for simulating fluids under shear which is of comparable efficiency to the most efficient domain decomposition algorithms at equilibrium.

2. THE FUJITSU AP1000

A program using the algorithm described in this paper was developed and tested on a Fujitsu AP1000 supercomputer. The Fujitsu AP1000 is a distributed memory parallel processor. It consists of a host processor and between 16 and 1024 cell processors. The AP1000 is classified as a Multiple Instruction Multiple Data (MIMD) machine as each cell processor can operate independently of all others. Each cell processor has its own memory and can send or receive data from other cell processors or the host processor. All external I/O must be done through the host processor. Together, the cell processors are known as a Cellular Array Processor.

The AP1000 has three separate high performance communication networks, the Broadcast network, or B-net, the Status and Synchronisation network, or S-net and the Torus network, or T-net [14]. Having three separate networks improves performance

by avoiding interference between the three type of message traffic. The B-net allows the host, or a cell processor, to broadcast data to all cell processors. The S-net allows cells or the host to test what state the cell processors are in. The T-net allows point to point communication between individual cell processors. The T-net is a two dimensional torus topology network and hence each cell processor has four immediately adjacent neighbours (Figure 2.1). The provided message passing libraries allow information to be sent from one specific cell processor to another specific cell processor, or to a neighbouring cell processor by using NORTH, SOUTH, EAST or WEST to refer to the four immediately adjacent neighbours.

There is also a library of functions provided which allow for data from all processors to be combined in some way. These include summing values from all processors, finding the maximum or minimum value of all processors, as well as global AND and OR functions.

3. ALGORITHM

Planar Couette Flow

In these simulations the thermostatted SLLOD algorithm for fluids undergoing planar Couette flow is used. This algorithm is well known [15, 16]. The equations of motion are

$$\dot{\mathbf{q}}_i = \frac{\mathbf{P}_i}{m} + \mathbf{i}\gamma y_i$$

$$\dot{\mathbf{p}}_i = \mathbf{F}_i - \mathbf{i}\gamma \mathbf{p}_y - \alpha \mathbf{p}, \tag{1}$$

where $\mathbf{q}_i$ is the position vector, $\mathbf{p}_i$ is the momentum vector and y_i the y-position of particle i, and $\mathbf{F}_i$ the total force exerted by all other particles on particle i. The strain rate, or shear rate, γ, can be expressed as $\gamma = \partial u_x/\partial y$, where u_x is the linear streaming velocity of particle i (we assume the Reynolds number is low). The thermostatting term, $\alpha \mathbf{p}$, is added to remove the viscous heat which is produced irreversibly by the shear and

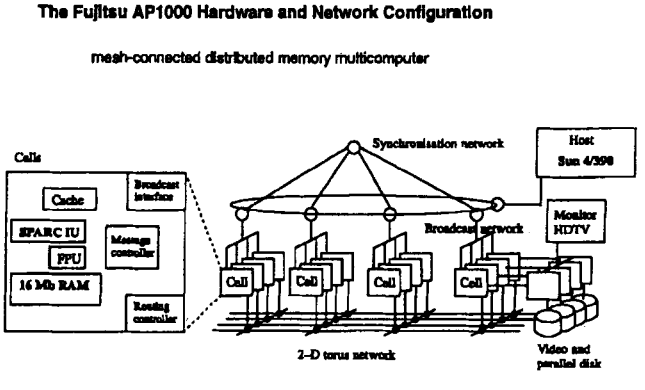

Figure 2.1 Network connection of cell processors in the AP1000 (taken from [14]).

allows the system to (possibly) reach a steady state. The thermostat used in this case is the Gaussian isokinetic thermostat,

$$\alpha = \frac{\sum_{i=1}^{N} (\mathbf{F}_i \cdot \mathbf{p}_i - \gamma p_{xi} p_{yi})}{\sum_{i=1}^{N} \mathbf{p}_i^2} \tag{2}$$

where α is the thermostat multiplier.

The integration of the equations of motion is an important aspect of a molecular dynamics simulation. In these simulations the Gear-Predictor algorithm is used to integrate the equations of motion. The Gear-Predictor is a higher-order integration scheme than various commonly used Verlet methods.

Lees-Edwards Periodic Boundary Conditions

The SLLOD equations of motion together with the Lees-Edwards Periodic Boundary Conditions [13] give an exact description of planar Couette flow, arbitrarily far from equilibrium. (Note at high Reynolds number the flow may become turbulent). The Lees-Edwards, or *sliding brick*, periodic boundary conditions, as represented in Figure 3.1, are used to drive the shear flow. The periodic images of the simulation box are translated horizontally from one layer to the next. The motion of the image box with respect to the simulation box, defines the strain rate, $\gamma = \partial u_x / \partial_y$, where u_x is the induced linear streaming velocity of particle i.

The laboratory velocity of each particle is the sum of two parts: a thermal or peculiar velocity and a streaming velocity. The motion of the images particles in the image boxes above and below the simulation box will impose the linear streaming velocity, $\mathbf{u}_i$, on each particle in the simulation box. If the streaming velocity of the simulation box in the x direction is s_x, then the row of image boxes immediately above the simulation box will have a streaming velocity of $s_x + \gamma L \mathbf{n}_x$, where γ is the shear rate, L is the length of a side

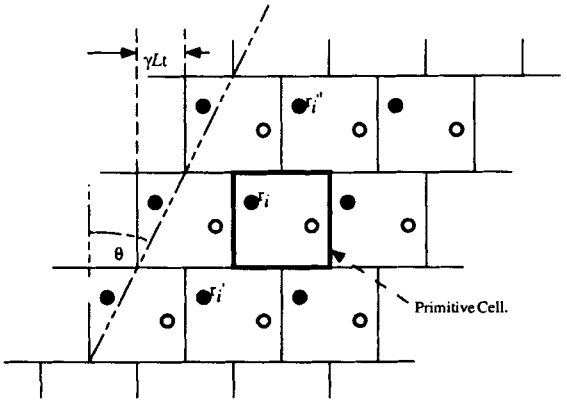

Figure 3.1 Lees-Edwards periodic boundary conditions.

of the simulation box and $\mathbf{n}_x$ is the unit vector in the x direction. Similarly the row of image cells below the simulation box will have a streaming velocity of s_x-$\gamma L\mathbf{n}_x$. From this the position of the image particles in the image simulation boxes can be found. The image of a particle in the image box above at time t_1 will be

$$\mathbf{r}'_i(t_1) = (\mathbf{r}_i + \gamma L\mathbf{n}_x t_1)$$

and below

$$\mathbf{r}'_i(t_1) = (\mathbf{r}_i - \gamma L\mathbf{n}_x t).$$

Under conventional periodic boundary conditions, when a particle moves out of the simulation box, through any side, it is replaced by a particle entering through any side, it is replaced by a particle entering through the opposite side. The new particle will have the same laboratory momenta, but with a position

$$\mathbf{r} = (\mathbf{r}) \bmod L.$$

This remains true under Lees-Edwards periodic boundary conditions except when the particle moves out of the simulation box through a y face. In this case the replacing image particle will not have the same laboratory velocity, nor necessarily the same x coordinate. If particle i in figure 2.1, moves out the top side of the simulation box, it is replaced by $\mathbf{r}''_i$, and

$$\mathbf{r}''_i = (\mathbf{r}_i - \gamma \mathbf{L}\mathbf{n}_x t_1) \bmod \mathbf{L}$$

When calculating the new laboratory velocity of particle i the streaming velocity of the image particle must be considered and hence the new laboratory velocity will be

$$\dot{\mathbf{r}}''_i = (\dot{\mathbf{r}}_i - \gamma \mathbf{L}\mathbf{n}_x)$$

Similarly if particle i moves out the bottom side of the simulation box to a position, $\mathbf{r}_i$ it will be replaced by a particle with position $\mathbf{r}'_i$. The new position of particle i, $\mathbf{r}'_i$, will be

$$\mathbf{r}'_i = (\mathbf{r}_i + \gamma \mathbf{L}\mathbf{n}_x t_1) \bmod \mathbf{L}.$$

Similarly the new velocity will be

$$\dot{\mathbf{r}}'_i = (\mathbf{r}_i + \gamma \mathbf{L}\mathbf{n}_x).$$

From these equations of motion we see that although the laboratory velocities change as indicated above, the SLLOD momenta, $\mathbf{p}_i$, do not change.

These Lees-Edwards periodic boundary conditions are sufficient to define a *boundary driven* algorithm for planar Couette flow. Alternatively, they can be combined with the thermostatted SLLOD equations of motion to provide a homogeneous algorithm for shear flow.

Potential

The fluids used in these simulations were either Lennard-Jones fluids, using the Lennard-Jones potential:

$$\Phi = \begin{cases} 4\varepsilon\left[\left(\dfrac{\sigma}{r}\right)^{12} - \left(\dfrac{\sigma}{r}\right)^{6}\right], & r \le r_c \\ 0, & r > r_c \end{cases} \tag{4}$$

where

$$r_c = 2.5\sigma,$$

or the Weeks-Chandler-Andersen (WCA) potential:

$$\Phi = \begin{cases} 4\varepsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right] + \varepsilon, & r \leq r_c \\ 0, & r > r_c \end{cases} \tag{5}$$

where $r_c = 2^{1/6}\sigma$,
or a soft sphere fluid using the soft sphere potential:

$$\Phi = \begin{cases} \varepsilon\left(\frac{\sigma}{r}\right)^{12}, & r \leq r_c \\ 0, & r > r_c \end{cases} \tag{6}$$

where $r_c = 1.5\sigma$.

The force between two particles can be found from the potential functions. For the Lennard-Jones potential, the cut-off, r_c, in the potential can be made longer, however the algorithm which we describe is best suited to short range potentials.

Calculation of Forces

The calculation of forces will be described for a two dimensional fluid, however the description is trivially extended to three dimensions.

The calculation of the force on each particle due to all other particles is the most CPU-time expensive part of an MD simulation. The distance between each pair of particles needs to be calculated, and if the distance is less than the cut-off distance, r_c, then the force between them is calculated. Algorithms which consider every pair of particles are $O(N^2)$ and rapidly become expensive for large systems. To decrease the time spent during the forces calculation it is necessary to minimise the number of particle pairs for which the distance is calculated. The most efficient method possible would be $O(N.N_N/2)$, where N_N is the maximum number of interacting neighbours of any one particle, e.g for the WCA potential in a two-dimensional fluid, $N_N = 6$. There are a number of methods which can be used to minimise the number of particle pairs for which the distance need to be calculated. Newton's 3rd law (Action and Reaction) can be used so that the force between particle i and particle j need only be calculated once, i.e. $F_{ij} = -F_{ji}$.

Since the cut-off distance, r_c, is small compared to the size of the simulation box then the distance between a large number of particle pairs will be greater than the cut-off distance. In these cases the use of so called "cell code" can be used to minimise the number of pairs of particles which need be compared. The simulation box is divided into smaller *cells*, with the size of each cell large enough so that each particle can only interact with particles in the same cell or in immediately neighbouring cells. Only particles in neighbouring cells need then be considered when calculating the distances between particle pairs.

Another way of minimising computation is to use a neighbour list. A neighbour list is a list of particle pairs which are within a distance of $r_c + r_d$, where $0 < r_d \ll r_c$. The neighbour list can then be used to find particle pairs which are within r_c of each other and then the force between them can be calculated. Only when any particle has moved more than $r_d/2$ from its position at the time at which the last neighbour list was formed, need a new neighbour list be formed. The computational cost can be reduced by forming the neighbour list using cell code, an O(N) operation. It would therefore seem that a combined algorithm is optimal.

On a scalar machine a neighbour list is built in the following way. Figure 3.2 shows how the simulation box is split into square cells of sidelength $r_c + r_d$. The distance between all particle pairs in neighbouring cells, and between particle pairs within the home cell, is then calculated. If the distance is less than or equal to $r_c + r_d$ they are added to the neighbour list of particle pairs. Using Newton's Third Law, we only need compare cells once. For cells not along the top or left edges of the simulation box, particles in each cell are compared with particles in cells above and cells to the left, as well as all particles in the home cell. e.g. all particles in cell 11 of Figure 3.2 are compared with all particles in cells 6, 7, 8, 10 and all other particles in cell 11. For cells on the left and top edges of the simulation box, the particles are compared with the images of the cells along the bottom and right edges, as can be seen for cells 1 and 4 in Figure 3.2. All particles in cell 1 are compared with the images of all particles in cells 4, 16, 13 and 14. Particles in cell 4 are compared with the images of the particles in cells 15, 16 and 13 as well as particles in cell 3. A complete neighbour list of particle pairs which are separated by a distance of less than $r_c + r_d$ has then been built.

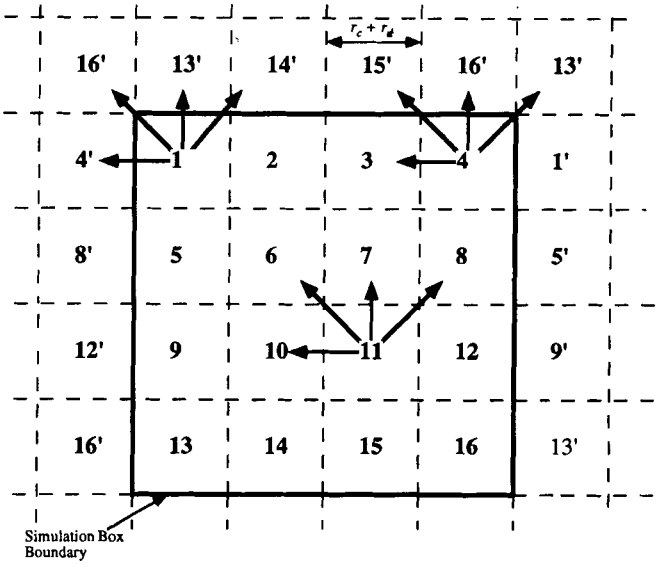

Figure 3.2 Cell code is used to decrease the number of particle pairs which need to be compared.

These methods, which minimise the amount of computation which must be used in order to calculate the total force on each particle, will be included in the algorithm for the distributed memory machine.

Once the forces on each particle are calculated, the equations of motion can be solved numerically and the time-averaged thermodynamic and transport properties of the system calculated.

3.1. Equilibrium MD on distributed memory machines

In order to use a domain decomposition technique the physical space of the simulation box must be mapped onto the processors of the AP1000. The model used in this algorithm is to divide the simulation box geometrically into sub-areas, and then map each sub-area onto a processor. All particles which are within that sub-area are assigned to that processor, which becomes those particles' *home processor.*

For the reasons discussed in the introduction it is best to subdivide a two dimensional system into squares, and a three dimensional system into cubes. Figure 3.3a shows a two dimensional system subdivided into squares and how those sub-areas are the mapped onto the processors. In this case neighbouring processors are assigned neighbouring sub-areas, which allows the relative addressing of the two-dimensional Torus network to be used. Figure 3.3b shows how the sub-areas of a three dimensional system may be mapped onto the processors. This means that the relative addressing of the two-dimensional Torus network cannot be used in the X and Z directions. Absolute addresses must be calculated to find the processors which are simulating neighbouring

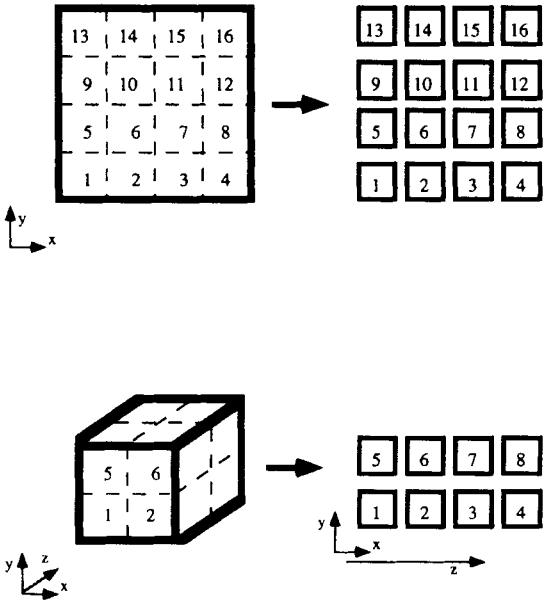

Figure 3.3 The mapping of the physical space on to the processors. Here seen for a two dimensional fluid (a) and a three dimensional fluid (b).

sub-areas of the simulation box in these directions. However, once these absolute addresses have been calculated they remain the same for the duration of the simulation.

Due to this decomposition of the simulation box, some interacting particle pairs will be in neghbouring sub-areas, which are being simulated on different processors. Hence, each processor must know which processors are assigned the neighbouring sub-areas. The necessary particle information must then be passed to the necessary neighbouring processors in order for the total force on each particle to be calculated.

Calculation of Forces

As discussed earlier, for systems with large numbers of particles using a neighbour list is the most efficient way of calculating the force on each particle. Since some interacting particle pairs will be on neighbouring processors, coordinate information about some of the particles involved in these particle pairs must be passed to the necessary processors in order for a neighbour list to be built and the total force on each particle to be calculated.

Here we will describe the algorithm for building the neighbour list for a two dimensional fluid. This can be trivially extended to three dimensions, with the exception that absolute addresses and not the relative addressing within the Torus network must be used.

For a two dimensional fluid the simulation box is sub-divided into squares, with each processor assigned a square sub-area and the two-dimensional Torus network of the AP1000 used, with neighbouring sub-areas being assigned to neighbouring processors (Figure 3.3a).

As in the scalar program "cell code" is used to build the neighbour list, hence the simulation box is divided into cells. So that each processor has all necessary information to build a neighbour list, particle information must be sent to neighbouring processors. All particles which are in cells on the EAST side of the sub-area of each processor are sent to the processor to the EAST. These are then received by that processor to the EAST, i.e. each processor receives a list of particles from the processor to the WEST. These particles are then added to the list of particles on that processor. All particles in the cells on the SOUTH edge of the sub-area on each processor are then sent SOUTH. These are then received by that processor to the SOUTH, i.e. each processor receives a list of particles from the processor to the NORTH. These particles are then added to the list of particles on that processor. This sending order can be seen in Figure 3.4.

It is now possible to build a complete neighour list containing all particle pairs which are within a distance of $r_c + r_d$. For cells which are not on the edge of the sub-area of a processor, the neighbour list is built in the normal way, as can be seen for cell 7 of Figure 3.5. However it is not possible to calculate the distance between particles in cells on the EAST edge of the sub-area (4, 8, 12, & 16) with particles in cells to the NORTH-EAST, as would normally happen. This must be done on the processor to the EAST, hence the distance between particles in cells on the WEST edge of the sub-area and particles in cells to the SOUTH-WEST. e.g. particles in cell 5 and cell 12a. This is not possible for cell 13 and the neighbour list is complete except for the calculation of the distance between particles in cell 4 and particles in cell 13 on the processor to the

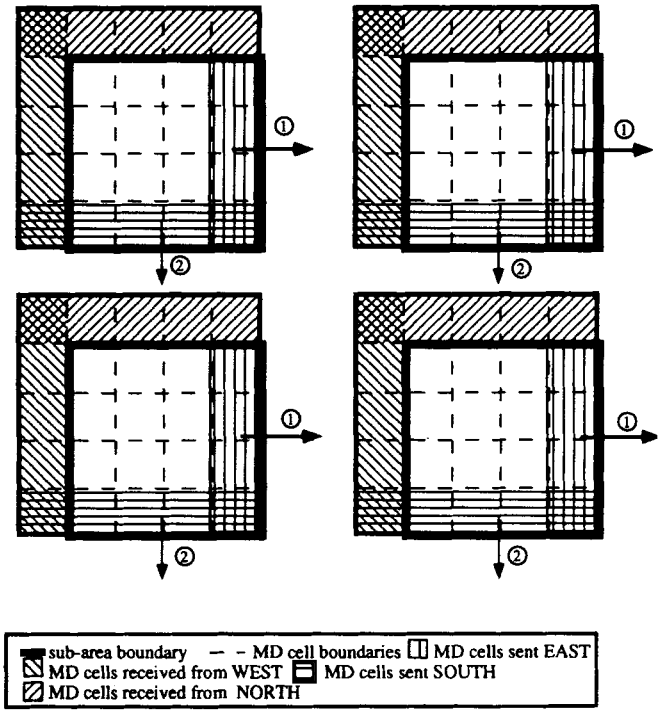

Figure 3.4 The sending order for sending the necessary coordinate information to neighbouring processors so that a neighbour list can be built.

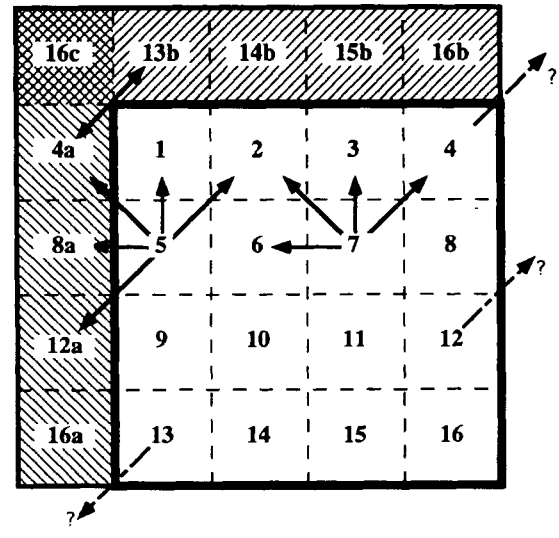

Figure 3.5 Building of the neighbour list on a processor. Information from neighbouring processors has been received.

NORTH EAST. Since the particle information in cell 4 has been sent EAST and cell 13 from the processor to the NORTH EAST has been sent SOUTH, this interaction can take place on the neighbouring processor to the EAST. In Figure 3.5, the interaction of cell 4a and cell 13b is taking place on this processor instead of the processor to the WEST. By following this algorithm a complete neighbour list can be built.

When the particle information is sent to the neighbouring processors and a neighbour list built, the force on each particle can be calculated from the particle coordinates. Not all the force information about each particle will be on the particle's *home processor*. In order to calculate the new position and momentum of each particle this force information must be sent back to those particle's *home processors*. This force information is sent back to the home processor in the reverse order of how the coordinate information was sent. For the particles received from the NORTH, the force information is sent NORTH, and added to the force information for particles on that processor. Then for particles initially received from the WEST, the force information is sent back to the WEST (now the *home processor*) and added, so that now the total force on each particle is known by each particle's *home processor*.

At each timestep the necessary particle information must be sent to neighbouring processors, the force information calculated and sent back to *home processors*. For this reason a list of particles, whose information must be sent to neighbouring processors is kept and only updated when a new neighbour list is formed. The home processor can then calculate the new positions and momentum for each particle using the integration of the equations of motion.

Once the total force on each particle is calculated the simulation can proceed as for the scalar program.

Migration of particles

During the simulation, particles will migrate across the sub-areas of the simulation box assigned to each processor. It would seem that a new neighbour list would be required every time a particle crosses a sub-area boundary. However it is not necessary to move a particle every time it leaves the sub-area assigned to its *home processor*. It is only when a new neighbour list is formed that it is necessary to check if any particles have moved out of the sub-area assigned to its *home processor*. If a particle has moved out of the sub-area assigned to it's *home processor* the particle is then moved to a new *home processor*.

Accumulation of thermodynamic data

Thermodynamic and transport data are computed on individual processors for their individual host particles. The data from all processors must be combined to form the required N-particle sums. This can be easily done by using the global sum library functions which are provided.

Load Balancing of the processors

It is important that the processors all have a similar number of particles assigned to them in order for the machine to be *load balanced*. This minimises the time for which

some processors are idle, allowing for the most efficient use of the machine. Hence each processor is assigned an equal sub-area of the simulation box. This may require slight adjustments in the value of r_d, so that when the simulation box is divided into cells for building the neighbour list there is a whole number of cells on each processor. This works well for fluids which are relatively homogeneous with respect to density.

Shear Flow on Multi Processor Machines

This algorithm presents several problems when the Lees-Edwards periodic boundary conditions are added in order to simulate the shear flow. Because of the 'sliding' nature of the images of the simulation box, the image simulation boxes no longer line up directly underneath each other, but change as the simulation proceeds, as can be observed in Figure 3.6. Neighbouring processors no longer represent neighbouring sub-areas. If the current algorithm is used for a large simulation, particles near the top of the simulation box will be moving so fast due to the applied shear that they could move through the sub-area of a processor within a single timestep. In order to overcome these complications, a complex time dependent communication pattern would have to be employed.

The solution to this problem is to use a simulation box which is a Lagrangian parallelogram (or rhomboid in 3D), instead of a static square (or cube in 3D). This idea was used by Evans [17]. The simulation box, as can be seen in Figure 3.7, changes shape throughout the course of the simulation until the shear angle becomes equal to 45°. The simulation box is then realigned to $-45°$. This means that the sub-areas which are assigned to each processor would be co-moving, or Lagrangian, parallelograms.

If a neighbour list is employed, and cell code is used to build that neighbour list, then the cells into which the simulation box is divided would also be co-moving parallelograms which change shape during the simulation. If the size of these cells remained the same, $r_c + r_d$, then as the shear angle increases particle pairs within $r_c + r_d$ of each other need be no longer in neighbouring cells, as seen in Figure 3.8a. There must be a slight alteration to the size of the cells. The size of the cells must be increased to $1/(\cos 45°)$ $(r_c + r_d)$ i.e. $\sqrt{2}(r_c + r_d)$, so that even at the largest possible shear angle, 45°, particles

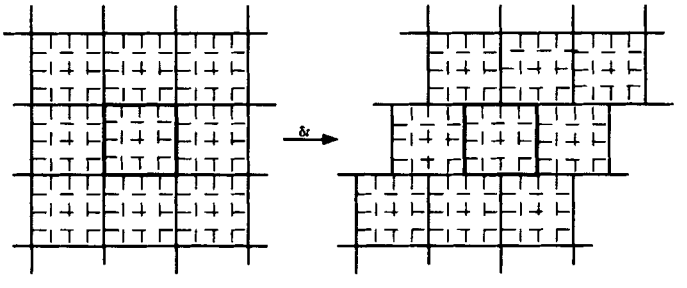

Figure 3.6 The simulation box is surrounded by periodic images, and the sliding nature of the Lees-Edwards periodic boundary conditions can be seen over time.

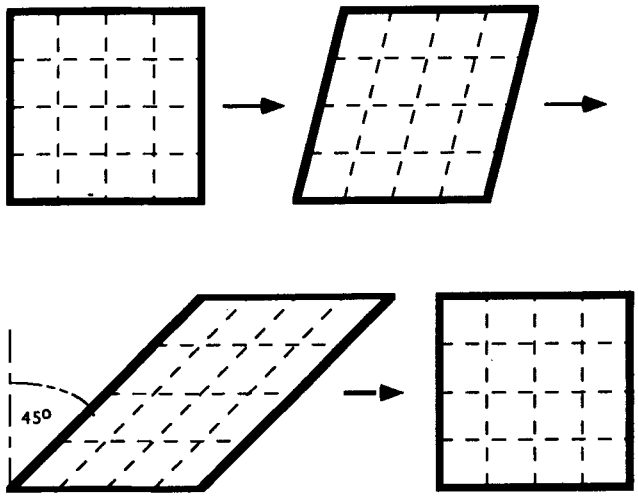

Figure 3.7 The simulation box will continue to change shape throughout the simulation, until it reaches an angle of 45°, when it is re-aligned $-45°$.

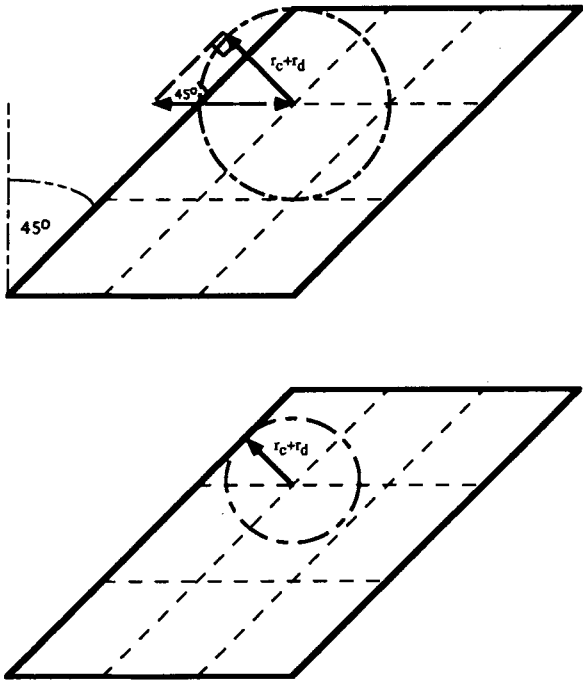

Figure 3.8 The size of the cell must be increased so that all particle pairs are within $r_c + r_d$ of each other.

which will interact will be in neighbouring cells, as can be seen in Figure 3.8b. At first glance it may seem more efficient to set the size of the cells to $1/(\cos\theta)(r_c + r_d)$, where θ is the shear angle at the time of building the neighbour list. However, in order for the machine to be *load balanced*, it is more efficient to divide the simulation box into the same number cells throughout the simulation.

These changes allow neighbour lists to be formed and the simulation to proceed as for the Equilibrium MD case.

4. PROGRAM EVALUATION

The purpose of the algorithm presented in this paper is to allow scalable simulations of shear flow in large two and three dimensional systems on a distributed memory machine, thus obtaining the appropriate transport and thermodynamic data.

Scaling the algorithm to large systems

In order to simulate large numbers of particles efficiently, the algorithm must be scalable, so that as the number of particles, N, increases, the time taken per timestep remains proportional to N. Tables 1 and 2 show the average time per timestep for two and three dimensional systems of different sizes. These average times were taken over a 1000 timestep simulation once the system had reached a steady state. For the two dimensional systems the soft sphere potential with $r_c = 1.5\sigma$ was used with $T^* = 1.0$, $\rho^* = 0.9238$. For the three dimensional systems the WCA potential was used at the Lennard-Jones triple point, $T^* = 0.722$, $\rho^* = 0.8442$. These simulations used a shear rate $\gamma = 0.01$. Figures 4.1 and 4.2 show the number of particles per processor, N_p, plotted against the time per timestep for each sized system. The resulting linear plots show that the algorithm scales very well.

The communication costs within a simulation must also be taken into account. Tables 4.1 and 4.2 include the percentage of time used for communication within the

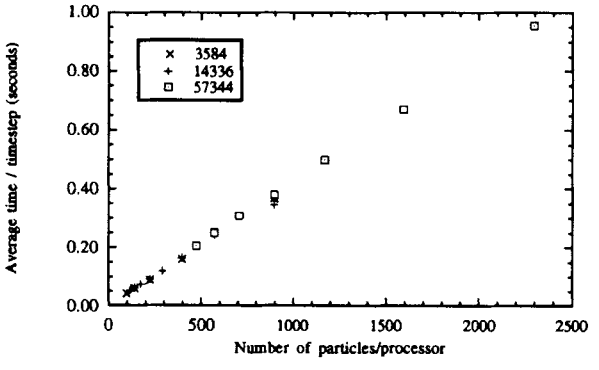

Figure 4.1 The number of particles per processor vs time per timestep for a two dimensional fluid.

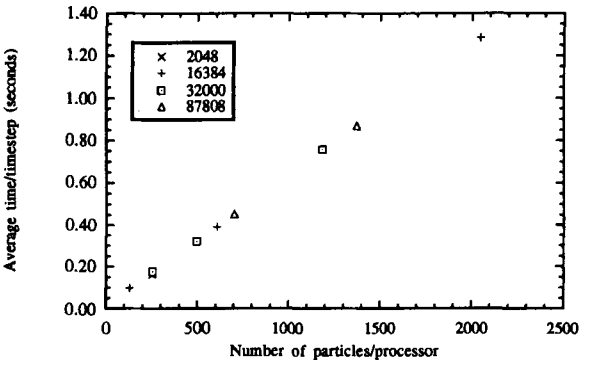

Figure 4.2 The number of particles per processor vs time per timestep for a three dimensional fluid.

Table 4.1 The average time per timestep for various two dimensional systems.

N	N_p	N/N_p	*Time/timestep*	*% comm*
3584	4	896.0	0.368	3.5
3584	9	398.2	0.159	5.2
3584	16	224.0	0.089	7.4
3584	25	143.4	0.059	8.5
3584	36	99.6	0.042	9.8
14336	16	896.0	0.347	2.9
14336	25	573.4	0.244	3.5
14336	36	398.2	0.163	6.5
14336	49	292.6	0.120	7.5
14336	64	224.0	0.092	8.2
14336	81	177.0	0.074	8.7
14336	100	143.4	0.062	9.5
14336	121	118.5	0.053	9.9
57344	25	2293.8	0.955	1.9
57344	36	1592.9	0.670	2.2
57344	49	1170.3	0.500	2.6
57344	64	896.0	0.381	3.0
57344	81	707.9	0.309	3.3
57344	100	573.4	0.251	3.6
57344	121	473.9	0.205	4.0

same simulations. These percentages show that as the number of particles on each processor increases, the amount of time spent in communication decreases. Hence, communication costs will become relatively insignificant as the number of particles is increased.

It must be stressed that the Lees-Edwards periodic boundary conditions do add a considerable amount of time to the simulation over the equilibrium case. Since the size of the cells used in cell code for generating the neighbour list are increased, the number of particles sent to neighbour processors also increases, and hence communication costs, and the amount of time spent in force subroutines, increase.

Table 4.2 The average time per timestep for various three dimensional systems.

N	N_p	N/N_p	*Time/timestep*	*% comm*
2048	8	256.0	0.1628	11.1
16384	8	2048.0	1.2837	5.0
16384	27	606.8	0.3898	7.9
16384	64	256.0	0.1724	12.5
16384	125	131.0	0.0977	13.0
32000	27	1185.2	0.7570	6.5
32000	64	500.0	0.3200	9.0
32000	125	256.0	0.1750	11.3
87808	64	1372.0	0.8680	6.3
87808	125	702.5	0.4520	7.5

5. THE NUMBER DEPENDENCE OF THE VISCOMETRIC PROPERTIES OF SIMPLE FLUIDS

Thermodynamic and transport properties obtained using this method of simulating large systems undergoing Planar Couette flow can be compared with previous results. Table 5.1a shows results for a two dimensional fluid system of 14336 particles at $T^* = 1.0, \rho^* = 0.9238$. The soft sphere potential with cut off at 1.5σ was used. Table 5.1a shows results for a two dimensional fluid system of 57344 particles at $T^* = 1.0$, $\rho^* = 0.9238$. The soft sphere potential with cut off at 1.5σ was used. These show good agreement with previous results. These length of the simulations depended on the size of the system. Smaller systems ran for between 100 and 500 thousand timesteps, while larger systems ran for approximately 100 thousand timesteps. Table 5.2(a),(b) and (c) show results for three dimensional fluid systems of $N = 2048$, 16384, 32000 and 87808

Table 5.1a Results for a two dimensional fluid system of $N = 14336$ particles at $T^* = 1.0$, $\rho^* = 0.9238$. The soft sphere potential with cut off at 1.5σ was used.

γ	η (this work)	p (this work)	η ([18])	p ([18])
0.01	3.94 (12)	10.894 (1)	3.87 (3)	10.895 (1)
0.1	3.64 (3)	10.934 (2)	3.63 (2)	10.935 (1)
0.316	3.12 (1)	11.123 (2)	3.11 (1)	11.124 (2)
1.0	2.397 (6)	11.952 (1)	2.392 (3)	11.941 (1)

Table 5.1b Results for a two dimensional fluid system of $N = 57334$ particles at $T^* = 1.0$, $\rho^* = 0.9238$. The soft sphere potential with cut off at 1.5σ was used.

γ	η (this work)	p (this work)	η ([18])	p ([18])
0.01	4.1 (4)	10.893 (5)	3.8 (1)	10.895 (1)
0.1	3.65 (3)	10.934 (4)	3.65 (2)	10.932 (1)
0.316	3.11 (1)	11.122 (3)		
1.0	2.39 (1)	11.950 (4)	2.401 (2)	11.956 (3)

Table 5.2 (a) Viscosity results for three dimensional fluid systems at $T^* = 0.722$, $\rho^* = 0.8244$ for the WCA potential. Included are results for four different system sizes and results from previous work. For the largest system, results for lower shear rates are included.

γ \N	2048 [15]	2048	16384	32000	87808
0.005					2.27 (23)
0.01					2.30 (1)
0.02					2.33 (7)
0.04	2.340 (12)				2.33 (3)
0.09	2.295 (20)	2.32 (3)	2.29 (3)	2.30 (5)	2.30 (8)
0.16	2.228 (14)	2.26 (3)	2.25 (2)	2.244 (6)	2.245 (4)
0.36	2.092 (7)				2.092 (2)
0.64	1.949 (6)	1.96 (1)	1.94 (3)	1.939 (6)	1.938 (2)
0.81					1.869 (1)
1.00	1.814 (11)	1.81 (1)	1.799 (4)	1.808 (3)	1.808 (2)
1.44	1.699 (7)	1.70 (2)	1.694 (6)	1.695 (3)	1.6948 (9)

Table 5.2 (b) Pressure results for three dimensional fluid system at $T^* = 0.722$, $\rho^* = 0.8244$ for the WCA potential. Included are results for four different system sizes and results from previous work. For the largest system, results for lower shear rates are included.

γ \N	2048 [15]	2048	16384	32000	87808
0.005					6.392 (1)
0.01					6.3912 (9)
0.02					6.3918 (8)
0.04	6.397 (1)				6.393 (2)
0.09	6.406 (2)	6.407 (3)	6.405 (2)	6.404 (4)	6.4069 (6)
0.16	6.436 (4)	6.437 (3)	6.436 (5)	6.4321 (5)	6.433 (1)
0.36	6.560 (3)				6.555 (1)
0.64	6.787 (7)	6.784 (18)	6.8 (6)	6.781 (3)	6.78 (2)
0.81					6.9412 (8)
1.00	7.145 (10)	7.142 (6)	7.134 (5)	7.137 (2)	7.1387 (5)
1.44	7.672 (10)	7.68 (3)	7.660 (9)	7.661 (6)	7.6627 (9)

particles at $T^* = 0.722$, $\rho^* = 0.8224$. The WCA potential with cut off at $2^{1/6}\sigma$ was used. Again good agreement is observed with previous results. Also included are results from previous studies for the 2048 particle system. Good agreement is observed with the previous work and between the system sizes.

An advantage of using larger systems is that better statistics should result. The planar couette flow results were extended for the largest system, $N = 87808$, into the low shear regime. Obtaining qualitative and quantitative results for fluids at these lower shear rates is important as these shear rates are much more compatible with shear rates which can be measured in actual experiments. Table 5.2 includes the results for the lower shear rates. Figure 5.1 shows the viscosity as a function of the square root of the strain rate. Although the error bars are large, the results tend to suggest that the viscosity begins to level out at lower shear rates, rather than continuing in a linear relationship with the square of the strain rate.

Table 5.2 (c) Internal energy results for three dimensional fluid system at $T^* = 0.722$, $\rho^* = 0.8244$ for the WCA potential. Included are results for four different system sizes and results from previous work. For the largest system, results for lower shear rates are included.

γ \N	2048 [15]	2048	16384	32000	87808
0.005					1.8083 (2)
0.01					1.8083 (1)
0.02					1.8082 (1)
0.04	1.8092 (2)				1.8085 (3)
0.09	1.8105 (2)	1.8107 (4)	1.8103 (3)	1.8101 (7)	1.8106 (1)
0.16	1.8154 (2)	1.8152 (5)	1.8149 (7)	1.8145 (1)	1.8147 (2)
0.36	1.8345 (6)				1.8333 (2)
0.64	1.8706 (10)	1.8701 (28)	1.871 (9)	1.8696 (5)	1.8694 (2)
0.81					1.8958 (1)
1.00	1.9301 (20)	1.9293 (12)	1.9281 (7)	1.9289 (3)	1.9289 (1)
1.44	2.0208 (25)	2.022 (6)	2.019 (1)	2.0189 (9)	2.0192 (1)

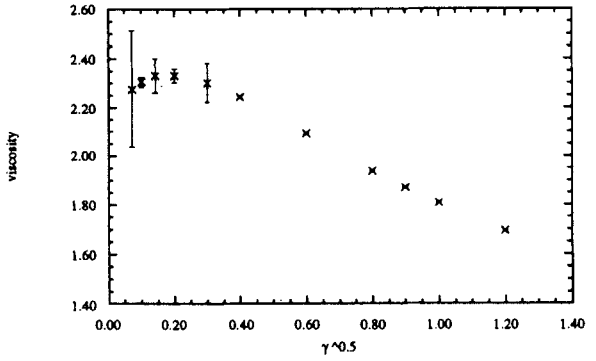

Figure 5.1 The viscosity of the three dimensional WCA fluid, for $N = 87808$ at $T^* = 0.7222$, $\rho^* = 0.8442$ as a function of the square root of the strain rate.

6. CONCLUSION

The advantage of this domain decomposition technique of simulating liquids undergoing shear flow is that it allows two dimensional systems to be decomposed into squares and three dimensional systems to be sub-divided into cubes. This allows more processors to be used to simulate the same number of particles, thereby increasing the speed of the computation.

Although the idea of using dynamic, or Lagrangian, simulation box is in this case applied to the simulation of shear flow in a liquid, the idea will be generally useful in simulating a number of nonequilibrium systems. There is a major problem in developing MD methods for systems where the streaming velocity is more complex than in planar Couette flow e.g. Poiseuille flow. For small channel widths and high flow rates where the Navier-Stokes relations break down, $u_x(y)$ is not known in advance. What is needed is an adaptive Lagrangian method.

Acknowledgement

The authors would like to thank the ANU Computer Science Department and Fujitsu for their generous grant of time on the AP1000.

References

[1] D. C. Rapaport, "Microscale hydrodynamics: Discrete-particle simulation of evolving flow patters", *Phys. Rev. A*, **36**, 3288 (1987).
[2] D. C. Rapaport, "Large-scale molecular dynamics simulation using vector and parallel computers", *Comput. Phys. Rep.*, **9**, 1 (1988).
[3] S. Y. Liem, D. Brown and J. H. R. Clarke, "Molecular dynamics simulation on distributed memory machines", *Compu. Phys. Comm.*, **67**, 261 (1991).
[4] S. Y. Liem, D. Brown and J. H. R. Clarke, "Investigation of the homogeneous-shear nonequilibrium molecular dynamics method", *Phys. Rev. A*, **45**, 3706-3713 (1992).
[5] W. Smith, "Molecular dynamics on hypercube parallel computers", *Comput. Phys. Comm.*, **62**, 229 (1991).
[6] H. G. Petersen and J. W. Perram, "Molecular dynamics on transputer arrays 1. Algorithm design, programming issues, timing experiments and scaling projections", *Mol. Phys*, **67**, 849 (1989).
[7] S. Cheynoweth, U. C. Klomp and L. E. Scales, "Simulation of organic liquids using psuedo-pairwise forces on a toriodal transputer array", *Comput. Phys. Comm.*, **62**, 297 (1991).
[8] A. R. C. Raine, D. Fincham and W. Smith, "Systolic loop methods for molecular dynamics simulation using multiple transputers", *Comput. Phys. Comm.*, **55**, 13 (1989).
[9] F. Burge and S. L. Formili, "Concurrent molecular dynamics simulation of spinodal phase transition on transputer arrays", *Comput. Phys. Comm.*, **60**, 31 (1990).
[10] M. R. S. Pinches and D. J. Tildesley, "Large scale molecular dynamics on parallel computers using the link-cell algorithm", *Mol. Sim.*, **6**, 51 (1991).
[11] D. Brown, J. H. R. Clarke, M. Okuda and T. Yamazaki, "A domain decomposition parallelization strategy for molecular dynamics simulation on distributed memory machines", *Comput. Phys. Comm.*, **74**, 67 (1993).
[12] K. Esselink, B. Smit and P. A. J. Hilbers, "Efficient parallel implementation of molecular dynamics on a toroidal network. Part 1. Parallelizing strategy", *J. Comp. Phys.*, **106**, 101 (1990).
[13] A. W. Lees and S. F. Edwards, "The computer study of transport processes under extreme conditions", *J. Phys. C*, **5**, 1921 (1971).
[14] H. Ishihata, T. Horie, I. Inano, T. Shimizu and S. Kato, "CAP-II Architecture". First Fujitsu-ANU CAP workshop, Kawasaki (1990).
[15] D. J. Evans, G. P. Morriss and L. M. Hood, "On the number dependence of viscosity in three dimensional fluids", *Mol. Phys.*, **68**, 637 (1989).
[16] D. J. Evans and G. P. Moriss, "Nonequilibrium statistical mechanics of liquids", Academic Press (1990).
[17] D. J. Evans, "The frequency dependent shear viscosity of methane", *Mol. Phys.*, **37**, 1745 (1979).
[18] L. M. Hood, D. J. Evans and S. T. Cui, "Number dependence of viscosity in two dimensional fluids", *Mol. Sim.*, **9**, 307 (1992).