CHAPTER XX

MOLECULAR DYNAMICS SIMULATIONS OF POLYMERS

Vagelis A. Harmandaris[1,2] and Vlasis G. Mavrantzas[1,*]

[1]Institute of Chemical Engineering and High-Temperature Chemical Processes, ICE/HT-FORTH, Patras, GR 26500

[2]Department of Chemical Engineering, University of Patras, Patras, GR 26500

I. The Molecular Dynamics Technique

Molecular dynamics (MD) is a powerful technique for computing the equilibrium and dynamical properties of classical many-body systems. Over the last fifteen years, due to the rapid development of computers, polymeric systems have been the subject of intense study with MD simulations [1].

At the heart of this technique is the solution of the classical equations of motion, which are integrated numerically to give information for the positions and velocities of atoms in the system [2], [3], [4]. The description of a physical system with the classical equations of motion rather than quantum-mechanically is a satisfactory approximation as long as the spacing $h\nu$ between successive energy levels described is $h\nu < k_BT$. For a typical system at room temperature this holds for $< \sim 0.6\text{x}10^{13}$ Hz, i.e. for motions of time periods of about $t >\sim 1.6\text{x}10^{-13}$ sec or 0.16 ps.

* Author to whom correspondence should be addressed, e-mail: vlasis@iceht.forth.gr, tel.: +30-2-610-965 214, fax: +30-2-610-965 223.

A simple flow diagram of a standard MD algorithm is shown in Figure [1] and includes the following steps:

1. First, a model configuration representing a molecular-level snapshot of the corresponding physical system is chosen or constructed, and is initialized (initial positions, velocities of each particle within the system).
2. Then the total force acting on each particle within the system is computed. For polymer systems such a force has two components: intermolecular (from atoms belonging to different polymer chains) and intramolecular (from atoms belonging to thee same chain).
3. The integration of the equations of motion follows with an appropriate method. The most popular of them will be described in detail in the next section.
4. Actual measurements are performed (positions, velocities, energies, etc, are stored) after the system has reached equilibration, periodically every N_k steps.
5. After completion of the central loop (N steps), averages of the measured quantities and of the desired properties are calculated and printed.

II. Classical Equations of Motion

As stated above, at the heart of an MD simulation is the solution of the classical equations of motion. Let us consider a system consisting of N interacting molecules described by a potential energy function V. Let us also denote as q_k and $\dot{q}_k$ the generalized coordinates describing the molecular configuration and their time derivatives, respectively. The classical equations of motion for this system can be formulated in various ways [5]. In the Lagrangian formulation, the trajectory $\mathbf{q}(t)$ $(=q_1(t), q_2(t), \ldots, q_k(t),\ldots)$ satisfies the following set of differential equations:

$$\frac{\partial L}{\partial q_k} = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_k}\right) \tag{1}$$

where L is the Lagrangian of the system. This is defined in terms of the kinetic, K, and potential energy, V, as $L = L(\mathbf{q}, \dot{\mathbf{q}}, t) \equiv K - V$. The generalized momenta p_k conjugate to the generalized coordinates q_k are defined as

$$p_k = \frac{\partial L}{\partial \dot{q}_k}. \tag{2}$$

Alternatively, one can adopt the Hamiltonian formalism, which is cast in terms of the generalized coordinates and momenta. These obey Hamilton's equations

$$\dot{q}_k = \frac{\partial H}{\partial p_k}, \qquad \dot{p}_k = -\frac{\partial H}{\partial q_k} \tag{3}$$

where H is the Hamiltonian of the system, defined through the equation

$$H(\mathbf{p}, \mathbf{q}) = \sum_k \dot{q}_k p_k - L \tag{4}$$

If the potential V is independent of velocities and time, then H becomes equal to the total energy of the system: $H(\mathbf{p}, \mathbf{q}) = K(\mathbf{p}) + V(\mathbf{q})$ [5]. In Cartesian coordinates, Hamilton's equations of motion read:

$$\dot{\mathbf{r}}_i \equiv \mathbf{v}_i = \frac{\mathbf{p}_i}{m_i}, \quad \dot{\mathbf{p}}_i = -\nabla_{\mathbf{r}_i} \mathrm{V} \equiv -\frac{\partial V}{\partial \mathbf{r}_i} = \mathbf{F}_i \tag{5}$$

hence

$$m_i \ddot{\mathbf{r}}_i \equiv m_i \dot{\mathbf{q}}_i = \mathbf{F}_i \tag{6}$$

where $\mathbf{F}_\mathrm{i}$ is the force acting on atom i. Solving the equations of motion then involves the integration of the $3N$ second-order differential equations (6) (Newton's equations).

The classical equations of motion possess some interesting properties, the most important one being the conservation law. If we assume that K and V do not depend explicitly on time, then it is straightforward to verify that $\dot{H} = dH/dt$ is zero, i.e. the Hamiltonian is a constant of the motion. In actual calculations this conservation law is satisfied if there exist no explicitly time- or velocity-dependent forces acting on the system.

A second important property is that Hamilton's equations of motion are reversible in time. This means that, if we change the signs of all the velocities, we will cause the molecules to retrace their trajectories backwards. The computer-generated trajectories should also possess this property.

There are many different methods for solving ordinary differential equations of the form of eq. (6). Criteria for the proper choice of an algorithm include the following:

- Algorithm must not require an expensively large number of force evaluations per integration time step. Many common techniques for the solution of ordinary differential equations (such as the 4^{th} order Runge-Kutta method) become inappropriate, since they do not fulfill this criterion.
- Algorithm should satisfy the energy conservation law. It is also desirable that it be time reversible and conserve volume in phase space (be symplectic).
- Algorithm should permit the use of a large time step *dt*.
- Algorithm should be *fast* and require *little memory*.

Concerning the solution of equations of motion for very long times, it is clear that no algorithm provides an essentially exact solution. But this turns out to be not a serious problem, because the main objective of an MD simulation is not to trace the exact configuration of a

system after long time, but rather to predict thermodynamic properties as time averages and calculate time correlation functions descriptive of the dynamics.

In the following we briefly describe the two most popular families of algorithms used in MD simulations for the solution of classical equations of motion: the higher order methods and the Verlet algorithms.

II.A. Higher-Order (Gear) Methods

The basic idea in the higher-order methods is to use information about positions and their first, second, … n^{th} time derivatives at time t in order to estimate positions and their first, second, .., n^{th} time derivatives at time $t+dt$ [2]. If we consider the Taylor expansion of the position vectors of a given particle at time $t+dt$ including terms up to 4th order we have

$$\mathbf{r}^p(t+dt) = \mathbf{r}(t) + dt\mathbf{v}(t) + \frac{dt^2}{2}\ddot{\mathbf{r}}(t) + \frac{dt^3}{6}\dddot{\mathbf{r}}(t) + \frac{dt^4}{24}\ddot{\dddot{\mathbf{r}}}(t) + \ldots \tag{7}$$

$$\mathbf{v}^p(t+dt) = \mathbf{v}(t) + dt\ddot{\mathbf{r}}(t) + \frac{dt^2}{2}\dddot{\mathbf{r}}(t) + \frac{dt^3}{6}\ddot{\dddot{\mathbf{r}}}(t) + \ldots \tag{8}$$

$$\ddot{\mathbf{r}}^p(t+dt) = \ddot{\mathbf{r}}(t) + dt\dddot{\mathbf{r}}(t) + \frac{dt^2}{2}\ddot{\dddot{\mathbf{r}}}(t) + \ldots \tag{9}$$

$$\dddot{\mathbf{r}}^p(t+dt) = \dddot{\mathbf{r}}(t) + dt\ddot{\dddot{\mathbf{r}}}(t) + \ldots \tag{10}$$

In the above equations, the superscript p denotes "predicted" values and dots denote time derivatives. Eqs. (7)-(10) do not generate classical trajectories, since we have not as yet introduced the equations of motion. To do this we estimate the size of the error incurred by the expansion, $\Delta\mathbf{x}$, by calculating the forces (or, equivalently, the accelerations) at the predicted positions

$$\Delta\mathbf{x} \equiv \ddot{\mathbf{r}}(\mathbf{r}^p(t+dt)) - \ddot{\mathbf{r}}^p(t+dt) = -diag(1/m_1, 1/m_2, ..., 1/m_N)\nabla_{\mathbf{r}}\mathrm{V}(\mathbf{r}^p(t+dt)) - \ddot{\mathbf{r}}^p(t+dt) \tag{11}$$

The error is accounted for and corrected in a "corrector" step, that is

$$\mathbf{r}^c(t+dt) = \mathbf{r}^p(t+dt) + c_0\Delta\mathbf{x} \tag{12}$$

$$\mathbf{v}^c(t+dt) = \mathbf{v}^p(t+dt) + c_1\Delta\mathbf{x} \tag{13}$$

$$\ddot{\mathbf{r}}^c(t+dt) = \ddot{\mathbf{r}}^p(t+dt) + c_2\Delta\mathbf{x} \tag{14}$$

$$\dddot{\mathbf{r}}^c(t+dt) = \dddot{\mathbf{r}}^p(t+dt) + c_3\Delta\mathbf{x} \tag{15}$$

where c_i, $i = 1,..,n$ are constants. The values of c_i are such that they yield an optimal compromise between desired level of accuracy and algorithm stability [2].

The general scheme of an algorithm based on the predictor-corrector method goes as follows:

(a) predict positions and their first, second, …, n^{th} time derivatives at time $t+dt$ using the values of these quantities at time t.

(b) compute forces using the predicted positions and then the corresponding error $\Delta\mathbf{x}$ from the differences between accelerations as calculated from forces and accelerations as predicted by the prediction scheme.

(c) correct the predicted positions and their first, second, …, n^{th} time derivatives guided by $\Delta\mathbf{x}$.

II.B. Verlet Methods

Algorithms in this family are simple, accurate and, as we will see below, time reversible. Thus, the Verlet methods are the most widely used methods for integrating the classical equations of motion. The initial form of the Verlet equations [3] is obtained by utilizing a Taylor expansion at times $t-dt$ and $t+dt$

$$\mathbf{r}(t+dt) = \mathbf{r}(t) + dt\mathbf{v}(t) + \frac{dt^2}{2}\ddot{\mathbf{r}}(t) + \frac{dt^3}{6}\dddot{\mathbf{r}}(t) + \mathcal{O}(dt^4) \tag{16}$$

$$\mathbf{r}(t-dt) = \mathbf{r}(t) - dt\mathbf{v}(t) + \frac{dt^2}{2}\ddot{\mathbf{r}}(t) - \frac{dt^3}{6}\dddot{\mathbf{r}}(t) + \mathcal{O}(dt^4) \tag{17}$$

Summing the two equations gives

$$\mathbf{r}(t+dt) = 2\mathbf{r}(t) - \mathbf{r}(t-dt) + dt^2\ddot{\mathbf{r}}(t) + \mathcal{O}(dt^4) \tag{18}$$

with $\ddot{\mathbf{r}}(t)$ calculated from the forces at the current positions.

Two modifications of the Verlet scheme are of wide use. The first is the "leap-frog" algorithm [3] where positions and velocities are not calculated at the same time; velocities are evaluated at half-integer time steps:

$$\mathbf{r}(t+dt) = \mathbf{r}(t) + dt\mathbf{v}(t+\frac{dt}{2}) \tag{19}$$

$$\mathbf{v}(t+\frac{dt}{2}) = \mathbf{v}(t-\frac{dt}{2}) + dt\ddot{\mathbf{r}}(t)\,. \tag{20}$$

In order to calculate the Hamiltonian H at time t, the velocities at time t are also calculated as averages of the values at $t + dt/2$ and $t - dt/2$:

$$\mathbf{v}(t) = \frac{1}{2}\left(\mathbf{v}(t+\frac{dt}{2}) + \mathbf{v}(t-\frac{dt}{2}) \right) \tag{21}$$

The problem of defining the positions and velocities at the same time can be overcome by casting the Verlet algorithm in a different way. This is the velocity-Verlet algorithm [3],[6], according to which positions are obtained through the usual Taylor expansion

$$\mathbf{r}(t+dt) = \mathbf{r}(t) + dt\mathbf{v}(t) + \frac{dt^2}{2}\ddot{\mathbf{r}}(t) \tag{22}$$

whereas velocities are calculated through

$$\mathbf{v}(t+dt) = \mathbf{v}(t) + \frac{dt}{2}\left[\ddot{\mathbf{r}}(t) + \ddot{\mathbf{r}}(t+dt)\right]. \tag{23}$$

with all accelerations computed from the forces at the configuration corresponding to the considered time. To see how the velocity-Verlet algorithm is connected to the original Verlet method we note that, by eq. (22),

$$\mathbf{r}(t+2dt) = \mathbf{r}(t+dt) + dt\mathbf{v}(t+dt) + \frac{dt^2}{2}\ddot{\mathbf{r}}(t+dt)\,. \tag{24}$$

If eq. (22) is written as

$$\mathbf{r}(t) = \mathbf{r}(t+dt) - dt\mathbf{v}(t) - \frac{dt^2}{2}\ddot{\mathbf{r}}(t)\,, \tag{25}$$

then, by addition, we get

$$\mathbf{r}(t+2dt) + \mathbf{r}(t) = 2\mathbf{r}(t+dt) + dt\left[\mathbf{v}(t+dt) - \mathbf{v}(t)\right] + \frac{dt^2}{2}\left[\ddot{\mathbf{r}}(t+dt) - \ddot{\mathbf{r}}(t)\right]. \tag{26}$$

Substitution of eq. (23) into eq. (26) gives

$$\mathbf{r}(t+2dt) + \mathbf{r}(t) = 2\mathbf{r}(t+dt) + dt^2\ddot{\mathbf{r}}(t+dt) \tag{27}$$

which is indeed the coordinate version of the Verlet algorithm. The calculations involved in one step of the velocity algorithm are schematically shown in Ref. 2 [Figure 3.2, page 80].

A sample code of the velocity-Verlet integrator is shown in algorithm 1. In this algorithm, *N* is the total number of atoms in the system and the subroutine *get_forces* calculates the total force on every atom within the system.

```
…….

do i = 1, N
     r(i) = r(i) +dt*v(i) +dt*dt/2*F(i)        ! update positions at t+dt using
                                                 velocities and forces at t
     v(i) = v(i) + dt/2*F(i)                   ! update velocities at t+dt using
                                                 forces at t
end do

call get_forces (F)                            ! calculate forces at t+dt

do i = 1, N
     v(i) = v(i) + dt/2*F(i)                   ! update velocities at t+dt
                                                 using forces at t+dt
end do
```

Algorithm 1: Velocity-Verlet integration method

In general, higher-order methods are characterized by a much better accuracy than the Verlet algorithms, particularly at small times. Their biggest drawback is that they are not reversible in time, which results in other problems, such as insufficient energy conservation, especially in very long-time MD simulations. On the other hand, the Verlet methods are not essentially exact for small times but their inherent time reversibility guarantees that the energy conservation law is satisfied even for very long times [4]. This feature renders the Verlet methods, and particularly the velocity-Verlet algorithm, the most appropriate one to use in long atomistic MD simulations.

III. MD in other statistical ensembles

The methods described before address the solution of Newton's equations of motion in the microcanonical (*NVE*) ensemble. In practice, there is usually the need to perform MD simulations under specified conditions of temperature and/or pressure. Thus, in the literature

there exist a variety of methodologies for performing MD simulations under isochoric of isothermal conditions [2],[3]. Most of these constitute a reformulation of the Lagrangian equations of motion to include the constraints of constant *T* and/or *P*. Among them the most widely used is the Nosé-Hoover method.

III.A. The Nosé-Hoover Thermostat

To constrain temperature, Nosé [7] introduced an additional degree of freedom, *s*, in the Lagrangian. The parameter *s* plays the role of a heat bath whose aim is to damp out temperature deviations from the desired level. This necessitates adding to the total energy an additional potential term of the form

$$V_s = gk_BT \ln s \tag{28}$$

and an additional kinetic energy term of the form

$$K_s = \frac{Q}{2}\left(\frac{\dot{s}}{s}\right)^2 = \frac{p_s^{\,2}}{2Q}. \tag{29}$$

In the above equations, *g* is the total number of degrees of freedom. In a system with constrained bond lengths, for example, $g = 3\,N_{atoms} - N_{bonds} - 3$, with N_{atoms} and N_{bonds} standing for the total numbers of atoms and bonds respectively; the value of 3 subtracted in calculating *g* takes care of the fact that the total momentum of the simulation box is constrained to be zero by the periodic boundary conditions. Q and p_s represent the "effective mass" and momentum, respectively, associated with the new degree of freedom *s*. Equations of motion are derived from the Lagrangian of the extended ensemble, including the degree of freedom *s*. Their final form, according to Hoover's analysis [8], is

$$\dot{\mathbf{r}}_i = \frac{\mathbf{p}_i}{m_i} \tag{30}$$

$$\dot{\mathbf{p}}_i = -\frac{\partial V}{\partial \mathbf{r}_i} - \frac{\dot{s}}{s}\mathbf{p}_i \tag{31}$$

$$\dot{p}_s = \left(\sum_{i=1}^{N} \frac{\mathbf{p}^2{}_i}{m_i} - g k_B T \right) / Q \,, \qquad p_s = Q\frac{\dot{s}}{s} \,. \tag{32}$$

An important result in Hoover's analysis is that the set of equations of motion is unique, in the sense that no other equations of the same form can lead to a canonical distribution.

The total Hamiltonian of the system, which should be conserved during the MD simulation, is

$$H_{Nose-Hoover} = \sum_{i=1}^{N} \frac{\mathbf{p}^2{}_i}{m_i} + V(\mathbf{r}^N) + g k_B T \ln s + \frac{p_s{}^2}{2Q} \,. \tag{33}$$

To construct MD simulations under constant *P*, an analogous reformulation of the Lagrangian was proposed by Andersen [9]. The constant-pressure method of Andersen allows for isotropic changes in the volume of the simulation box. Later Hoover [8] combined this method with the isothermal MD method described above to provide a set of equations for MD simulations in the *NPT* ensemble. Parrinello and Rahman [10] extended Andersen's method to allow for changes not only in the size, but also in the shape of the simulation box. This is particularly important in the simulation of solids (e.g., crystalline polymers) since it allows for phase changes in the simulation involving changes in the dimensions and angles of the unit cell.

III.B. The Berendsen Thermostat - Barostat

Berendsen proposed a simpler way for performing isothermal and/or isobaric MD simulations without the need to use an extended Lagrangian, by coupling the system into a temperature and/or pressure bath [11]. To achieve this, the system is forced to obey the following equations

$$\frac{dT}{dt} = \left(T - T_{ext}\right)/\tau_T \tag{34}$$

and

$$\frac{dP}{dt} = \left(P - P_{ext}\right)/\tau_p \tag{35}$$

where T_{ext} and P_{ext} are the desired temperature and pressure values and τ_T and τ_P are time constants characterizing the frequency of the system coupling to temperature and pressure baths. T and P are the instantaneous values of temperature and pressure, calculated from the momenta and configuration of the system [2]. The solution of these equations forces velocities and positions to be scaled at every time step by factors χ_T and x_P, respectively, with

$$x_T = \left(1 + \frac{dt}{\tau_T}\left(\frac{T}{T_{ext}} - 1\right)\right)^{1/2} \tag{36}$$

$$x_P = 1 - \beta_T \frac{dt}{\tau_P}\left(P - P_{ext}\right) \tag{37}$$

and β_T being the isothermal compressibility of the system.

The method proposed by Berendsen is much simpler and easier to program than that proposed by Nosé and Hoover. It suffers, however, from the fact that the phase-space probability density it defines does not conform to a specific statistical ensemble (e.g., *NVT, NPT*). Consequently, there exists no Hamiltonian that should be conserved during the MD simulation.

III.C. MD in the $NTL_x\sigma_{yy}\sigma_{zz}$ ensemble

To further illustrate how extended ensembles can be designed to conduct MD simulations under various macroscopic constraints, we discuss here the $NTL_x\sigma_{yy}\sigma_{zz}$ ensemble. $NTL_x\sigma_{yy}\sigma_{zz}$ is an appropriate statistical ensemble for the simulation of uniaxial tension experiments on solid

polymers [12] or relaxation experiments in uniaxially oriented polymer melts [13]. This ensemble is illustrated in Figure 2. The quantities that are kept constant during a molecular simulation in this ensemble are the following:

- the total number of atoms in the system N,
- the temperature T,
- the box length in the direction of elongation L_x and
- the time average values of the two normal stresses σ_{yy} and σ_{zz}.

The $NTL_x\sigma_{yy}\sigma_{zz}$ ensemble can be viewed as a hybrid between the NVT ensemble in the x direction and the isothermal-isobaric (NPT) ensemble in the y and z directions. The temperature T is kept fixed at a prescribed value by employing the Nosé-Hoover thermostat; the latter introduces an additional dynamical variable in the system, the parameter s, for which an evolution equation is derived. Also kept constant during an MD simulation in this ensemble is the box length L_x in the x direction; on the contrary, the box lengths in the other two directions, L_y and L_z, although always kept equal, are allowed to fluctuate. This is achieved by making use in the simulation of an additional dynamical variable, the cross-sectional area $A(=L_yL_z)$ of the simulation cell in the yz-plane, which obeys an extra equation of motion involving the instantaneous average normal stress $(\sigma_{yy}+\sigma_{zz})/2$ in the two lateral directions y and z, respectively; $(\sigma_{yy}+\sigma_{zz})/2$ remains constant on average and equal to $-P_{ext}$ throughout the simulation.

The derivation of the equations of motion in the $NTL_x\sigma_{yy}\sigma_{zz}$ ensemble has been carried out in detail by Yang *et al.* [12], and goes as follows: Consider a system consisting of N atoms with $\mathbf{r}_{ik}$ being the position of atom i belonging to polymer chain k. The bond lengths are kept fixed, with $\mathbf{g}_{ik}$ denoting the constraint forces on atom i. The Lagrangian is written as a function of the

"extended" variables $\{\tilde{\mathbf{R}}_k, \mathbf{x}_{i\kappa}, A, s\}$ where $\tilde{\mathbf{R}}_k$ is the scaled (with respect to the box edge lengths) position of the center of mass of every chain k, and $\mathbf{x}_{ik}$ is the position of atom i in chain k measured relative to the chain center of mass. This ensemble is "extended" in the sense that it invokes the additional variables A and s, makes use of a scaled coordinate system and is formulated with respect to a "virtual" time t'. The equations of motion are derived from the extended Lagrangian by exactly the same procedure as for the other statistical ensembles. The final equations are further recast in terms of real coordinates and real time and have the following form:

$$m_i \ddot{r}_{xik} = F_{xik} + g_{xik} - \frac{\dot{s}}{s} p_{xik}, \tag{38}$$

$$m_i \ddot{r}_{yik} = F_{yik} + g_{yik} - \frac{\dot{s}}{s} p_{yik} + \frac{m_i R_{yk}}{2A}\left(\ddot{A} - \frac{\dot{A}^2}{2A}\right), \tag{39}$$

$$m_i \ddot{r}_{zik} = F_{zik} + g_{zik} - \frac{\dot{s}}{s} p_{zik} + \frac{m_i R_{zk}}{2A}\left(\ddot{A} - \frac{\dot{A}^2}{2A}\right), \tag{40}$$

$$Q\ddot{s} = Q\frac{\dot{s}^2}{s} + s\left[\sum_k \sum_i \frac{p^2_{xik} + p^2_{yik} + p^2_{zik}}{m_i} - (g+1)k_B T\right], \tag{41}$$

$$W\ddot{A} = W\frac{\dot{s}\dot{A}}{s} + s^2 L_x\left[\frac{1}{2}\left((-\sigma_{yy}) + (-\sigma_{zz})\right) - P_{ext}\right], \tag{42}$$

where the forces with two indices indicate center of mass forces, while those with three indices are forces on atoms within a particular polymer chain. $\mathbf{R}_k$ denotes the center of mass of molecule k, while Q and W are inertial constants govern the fluctuations in the temperature the two normal stresses σ_{yy} and σ_{zz}, respectively. The total Hamiltonian of the extended system, derived from the Lagrangian, has the form:

$$H_{NTL_x\sigma_{yy}\sigma_{zz}} = \sum_i \frac{p_i^{\,2}}{2m_i} + V(\mathbf{r}) + \frac{Q}{2}\left(\frac{\dot{s}}{s}\right)^2 + (g+1)\frac{\ln s}{\beta} + \frac{W}{2}\left(\frac{\dot{A}}{s}\right)^2 + P_{ext}L_x A\,. \qquad (43)$$

The first term on the right hand side represents the kinetic energy, the second term is the potential energy and the last four terms are the contributions due to the thermostat and the fluctuating box cross-sectional area in the plane *yz*. Conservation of $H_{NTL_x\sigma_{yy}\sigma_{zz}}$ is a good test for the simulation.

For the solution of equations of motion, a modification of the velocity-Verlet algorithm proposed by Palmer [14] can be followed.

IV. Liouville Formulation of Equations of Motion - Multiple Time Step Algorithms

In section II.A we presented the most popular algorithms for integrating Newton's equations of motion, some of which are not reversible in time. Recently, Tuckerman *et al*. [15],[16] have shown how one can systematically derive time reversible MD algorithms from the Liouville formulation of classical mechanics.

The Liouville operator L of a system of N degrees of freedom is defined in Cartesian coordinates as

$$iL = \sum_{i=1}^{N}\left[\dot{\mathbf{r}}_i \frac{\partial}{\partial \mathbf{r}_i} + \mathbf{F}_i \frac{\partial}{\partial \mathbf{p}_i}\right]. \qquad (44)$$

If we consider the phase-space of a system, Γ≒{**r**,**p**}, the evolution of the system from time 0 to time t, can be found by applying the evolution operator

$$\Gamma(t) = \exp(iLt)\,\Gamma(0)\,. \qquad (45)$$

The next step is to decompose the evolution operator into two parts such that

$$iL = iL_1 + iL_2 \quad \text{with } iL_1 = \sum_{i=1}^{N}\left[F_i \frac{\partial}{\partial \mathbf{p}_i}\right], \quad iL_2 = \sum_{i=1}^{N}\left[\dot{\mathbf{r}}_i \frac{\partial}{\partial \mathbf{r}_i}\right]. \tag{46}$$

For this decomposition, a short-time approximation to the evolution operator can be generated via the Trotter theorem [16] as

$$\begin{aligned} \exp(iLt) = \exp(i(L_1 + L_2)t/P)^P = \\ (\exp(iL_1(dt/2))\exp(iL_2 dt)\exp(iL_1(dt/2)))^P + \mathrm{O}(t^3/P^2) \end{aligned} \tag{47}$$

where $dt = t / P$. Thus, the evolution operator becomes

$$\exp(iLdt) = \exp(iL_1 \frac{dt}{2})\exp(iL_2 dt)\exp(iL_1 \frac{dt}{2}) + \mathrm{O}\left(dt^3\right) \tag{48}$$

The evolution of the system at time t using the above factorization, eq. (48), is described through the following scheme [16]

$$\mathbf{r}(dt) = \mathbf{r}(0) + dt\mathbf{v}(0) + \frac{dt^2}{2m}\mathbf{F}\left[\mathbf{r}(0)\right] \tag{49}$$

$$\mathbf{v}(dt) = \mathbf{v}(0) + \frac{dt^2}{2m}\left(\mathbf{F}\left[\mathbf{r}(0)\right] + \mathbf{F}\left[\mathbf{r}(dt)\right]\right) \tag{50}$$

which can be derived using the identity $\exp\left[a\left(\partial/\partial g(x)\right)\right]x = g^{-1}\left[g(x) + a\right]$. The result is the well-known velocity-Verlet integration scheme, described before, which is now derived in a different way.

Based on the previous factorization a very efficient algorithm can be developed, through the use of different time steps for integrating the different parts of the Liouville operator. This is the so-called reversible REference System Propagator Algorithm (rRESPA).

IV.A. The rRESPA algorithm

In the rRESPA algorithm, the above factorization is employed together with an integration of each part of the Liouville operator with a different time step. In addition, the forces **F** are also decomposed into fast (short range) forces $\mathbf{F}^f$, and slow (long range) forces $\mathbf{F}^s$, according to $\mathbf{F}(\mathbf{r}) = \mathbf{F}^f(\mathbf{r}) + \mathbf{F}^s(\mathbf{r})$. The total evolution operator is broken up into $iL = iL_1 + iL_2 + iL_3$ with

$$iL_1 = \sum_{i=1}^{N}\left[\mathbf{F}_i^f(\mathbf{r})\frac{\partial}{\partial \mathbf{p}_i}\right], \qquad iL_2 = \sum_{i=1}^{N}\left[\dot{\mathbf{r}}_i\frac{\partial}{\partial \mathbf{r}_i}\right], \qquad iL_3 = \sum_{i=1}^{N}\left[\mathbf{F}_i^s(\mathbf{r})\frac{\partial}{\partial \mathbf{p}_i}\right]. \tag{51}$$

The heart of the rRESPA algorithm is that the equations of motion are integrated by using two different time steps, i.e. a Multiple Time Step (MTS) method: the slow modes (slow forces, iL_3) are integrated with a larger time step, Δt, whereas the fast modes (fast forces and velocities iL_1, iL_2) with a smaller time step, δt ($\delta t = \Delta t/n$). In this case the evolution operator becomes [16]

$$\exp(iL\Delta t) = \exp(iL_3\frac{\Delta t}{2})\left[\exp(iL_1\frac{\delta t}{2})\exp(iL_2\delta t)\exp(iL_1\frac{\delta t}{2})\right]^n \exp(iL_3\frac{\Delta t}{2}) + \mathrm{O}(\Delta t^3). \tag{52}$$

The force calculated n times (fast force) is called the reference force. A sample code of an MTS integrator is given in Algorithm 2.

```
…….

do i  = 1, N
    v(i) = v(i) + Δt/2*F^s(i)                    ! update velocities using
                                                 ! slow forces at t
end do

do j = 1, n
  do i  = 1, N
    v(i) = v(i) + δt/2*F^f((j-1)δt)              ! update velocities using
                                                 ! fast forces at t+(j-1)δt
    r(i) = r(i) +δt*v(i)                         ! update positions at t+jδt
  end do

   call fast_forces (F^f)                        ! get fast forces at t+jδt

  do i  = 1, N
    v(i) = v(i) + δt/2*F^f(jδt)
  end do
end do

call slow_forces (F^s)                           ! get slow forces at t+Δt

do i = 1, N
    v(i) = v(i) + Δt/2*F^s(i)                    ! update velocities using
                                                 ! slow forces at t+Δt
end do

…….
```

Algorithm 2: rRESPA integration

IV.B. rRESPA in the NVT ensemble

For MD simulation in the *NVT* ensemble, a modification of the rRESPA algorithm has been proposed. The method uses a modification of the Lagrangian of the system based on the Nosé-Hoover approach, described in section II.B. The difference from the standard rRESPA scheme described before is that in this case the total Liouville operator is decomposed as

$$iL = iL_1 + iL_2 + iL_3 + iL_{NH} \tag{53}$$

with

$$iL_1 = \sum_{i=1}^{N}\left[\mathbf{F}_i^f(\mathbf{r})\frac{\partial}{\partial \mathbf{p}_i}\right], \qquad iL_2 = \sum_{i=1}^{N}\left[\dot{\mathbf{r}}_i\frac{\partial}{\partial \mathbf{r}_i}\right], \qquad iL_3 = \sum_{i=1}^{N}\left[\mathbf{F}_i^s(\mathbf{r})\frac{\partial}{\partial \mathbf{p}_i}\right]. \quad (54)$$

Also,

$$iL_{NH} = -\sum_{i=1}^{N}\upsilon_\xi \mathbf{v}_i \bullet \frac{\partial}{\partial \mathbf{v}_i} + \upsilon_\xi \frac{\partial}{\partial \xi} + G\frac{\partial}{\partial \upsilon_\xi} \quad (55)$$

where

$$G = \left(\sum_{i=1}^{N}\frac{\mathbf{p}^2{}_i}{m_i} - gk_BT\right)/Q\,, \qquad \upsilon_\xi = \dot{\xi}\,, \quad (56)$$

and ξ is a transformation of the additional degree of freedom s, $\log s = N\xi$.

Two modifications of the standard RESPA method exist, depending on the application of the extended operator $\exp(iL_{NH}t)$. The first variant of RESPA is useful when the evolution prescribed by the operator $\exp(iL_{NH}t)$ is slow compared to the time scale associated with the reference force. It is formed by writing

$$\exp(iL\Delta t) = \exp(iL_{NH}\frac{\Delta t}{2})\exp(iL_3\frac{\Delta t}{2})\left[\exp(iL_1\frac{\delta t}{2})\exp(iL_2\delta t)\exp(iL_1\frac{\delta t}{2})\right]^n \exp(iL_3\frac{\Delta t}{2})\exp(iL_{NH}\frac{\Delta t}{2}) + \mathrm{O}(\Delta t^3) \quad (57)$$

and is named XO-RESPA (eXtended system Outside-Reference System Propagator Algorithm). In general, XO-RESPA can be applied to systems characterized by fast vibrations, as the time scale associated with the extended system variable is usually chosen to be quite slow compared with these motions.

If the motion prescribed by the operator $\exp(iL_{NH}t)$ occurs on the same time scale as that generated by the "fast" forces, then a useful RESPA algorithm must include the application of this operator for the small time step *dt*. The evolution operator takes then the form

$$\begin{aligned}\exp(iL\Delta t) = \exp(iL_{NH}\frac{\delta t}{2})\exp(iL_3\frac{\Delta t}{2})\exp(-iL_{NH}\frac{\delta t}{2}) \\ \left[\exp(iL_{NH}\frac{\delta t}{2})\exp(iL_1\frac{\delta t}{2})\exp(iL_2\delta t)\exp(iL_1\frac{\delta t}{2})\exp(iL_{NH}\frac{\delta t}{2})\right]^n . \\ \exp(-iL_{NH}\frac{\delta t}{2})\exp(iL_3\frac{\Delta t}{2})\exp(iL_{NH}\frac{\delta t}{2}) + \mathrm{O}(\Delta t^3)\end{aligned} \tag{58}$$

The resulting integrator is named XI-RESPA (eXtended system Inside-REference System Propagator Algorithm).

Modifications of the RESPA method for MD simulations in the NPT statistical ensemble have also been formulated in an analogous manner. More details can be found in the original papers [15],[16].

V. Constraint Dynamics in Polymeric systems

One of the most important considerations in choosing the best algorithm for the solution of the classical equations of motion is, as we saw above, the time step of integration. This should be chosen appreciably shorter than the shortest relevant time scale in the simulation. For long-chain polymeric systems, in particular, where one explicitly simulates the intramolecular dynamics of polymers, this implies that the time step should be shorter than the period of the highest-frequency intramolecular motion. This renders the simulation of long polymers very expensive. One solution to this problem is provided by the MTS algorithm discussed above. Another

technique developed to tackle this problem is to treat bonds between atoms, characterized by the highest frequency intramolecular vibrations, as rigid. The MD equations of motion are then solved under the constraint that bond lengths are kept constant during the simulation. The motion associated with the remaining degrees of freedom is presumably slower, permitting the use of a longer time step in the simulation.

In general, system dynamics should satisfy many constraints (e.g., many bond lengths that should be constant) simultaneously. Let us denote the functions describing the constraints by σ_i=0 with $\sigma_k = \mathbf{r}^2{}_{ij} - d^2{}_{ij}$, meaning that atoms i and j are held at a fixed distance d_{ij}. A new system Lagrangian is introduced that contains all constraints

$$L^c = L - \sum_k \lambda_k \sigma_k(\mathbf{r}) \tag{59}$$

where k denotes the set of constraints and λ_k the corresponding set of Lagrange multipliers. The equations of motion corresponding to the new Lagrangian are

$$m_i \ddot{\mathbf{r}}_i = \mathbf{F}_i - \sum_k \lambda_k \frac{\partial \sigma_k}{\partial \mathbf{r}_i} = \mathbf{F}_i - \mathbf{g}_i \tag{60}$$

where the second term on the right-hand side of eq. (60) denotes the constraint forces. The question then is how to calculate the set of Lagrange multipliers λ_k. Two methods that have widely been used in our MD simulations are discussed here: The Edberg-Evans-Morriss method and the SHAKE method.

V.A. The Edberg-Evans-Morriss algorithm

This algorithm [17] starts by considering a set of a linear system of equations in $\{\lambda_{ij}\}$, which are formulated by taking the second derivatives in time of the constraint equations:

$$\mathbf{r}^2{}_{ij} - d^2{}_{ij} = 0 \Rightarrow 2\mathbf{r}_{ij} \bullet \dot{\mathbf{r}}_{ij} = 0 \Rightarrow \mathbf{r}_{ij} \bullet \ddot{\mathbf{r}}_{ij} + (\dot{\mathbf{r}}_{ij})^2 = 0 \tag{61}$$

One then solves the following set of algebraic and differential equations simultaneously:

$$m_i \ddot{\mathbf{r}}_i = \mathbf{F}_i + \mathbf{g}_i \tag{62}$$

$$\mathbf{g}_i = \sum_k \lambda_k \frac{\partial \sigma_k}{\partial \mathbf{r}_i} \tag{63}$$

$$\mathbf{r}_{ij} \bullet \ddot{\mathbf{r}}_{ij} + (\dot{\mathbf{r}}_{ij})^2 = 0 \,. \tag{64}$$

Note that site velocities enter this formulation explicitly. Upon substitution of the site accelerations from eq. (60) into eq. (64) one obtains a system of linear equations in $\{\lambda_{ij}\}$; thus, the determination of λ_{ij}'s reduces to the solution of a linear matrix equation which should be addressed in each time step.

V.B The SHAKE - RATTLE algorithm

The approach described before suffers from the problem that it is computationally expensive, since it requires a matrix inversion at every time step. The problem gets worse with increasing chain length, with the algorithm becoming practically inappropriate for chains of more than about 100 atoms long. Ryckaert *et al*. [18] developed a simpler scheme, named SHAKE, to satisfy the constraints in this case.

If one considers the classical form of the Verlet algorithm, then in the presence of constraints

$$\mathbf{r}^c{}_i(t+dt) = \mathbf{r}^u{}_i(t+dt)_i - \frac{dt^2}{m_i} \sum_k \lambda_k \frac{\partial \sigma_k(t)}{\partial \mathbf{r}_i} \tag{65}$$

where $\mathbf{r}^c{}_i$ are the constrained and $\mathbf{r}^u{}_i$ the unconstrained positions. If the constraints are satisfied at time $t+dt$, then $\sigma^c{}_k(t+dt) = 0$. But if the system moved along the unconstrained trajectory, the

constraints would not be satisfied at $t+dt$. In this case, by performing a Taylor expansion around the unconstrained positions, we get

$$\sigma^{c}{}_{k}(t+dt) = \sigma^{u}{}_{k}(t+dt) + \sum_{i=1}^{N}\left(\frac{\partial\sigma_k}{\partial\mathbf{r}_i}\right)_{\mathbf{r}^{u}{}_{i}(t+dt)} \bullet\left[\mathbf{r}^{c}{}_{i}(t+dt) - \mathbf{r}^{u}{}_{i}(t+dt)\right] + \mathrm{O}(dt^4) \quad (66)$$

and by using eq. (65)

$$\sigma^{u}{}_{k}(t+dt) = \sum_{i=1}^{N}\frac{dt^2}{m_i}\sum_{k'}\lambda_{k'}\left(\frac{\partial\sigma_k}{\partial\mathbf{r}_i}\right)\left(\frac{\partial\sigma_{k'}}{\partial\mathbf{r}_i}\right). \quad (67)$$

The above equation has the structure of a matrix equation

$$\sigma^{u}{}_{k}(t+dt) = dt^2\mathbf{M}\boldsymbol{\Lambda} \quad (68)$$

By inverting the matrix, one can solve for the vector $\boldsymbol{\Lambda}$. However, since the Taylor expansion in eq. (66) has been truncated, the σ's should be computed at the corrected positions, and the preceding equations should be iterated until convergence is reached.

This procedure is also computationally expensive, because it requires a matrix inversion at every iteration, as does the Edberg–Evans–Morriss algorithm. Ryckaert proposed a new method, SHAKE, where the iterative scheme is not applied to all constraints simultaneously but to each constraint in succession. Thus the need of inverting a large matrix is avoided. The key point is that $\mathbf{r}^{c}{}_{i} - \mathbf{r}^{u}{}_{i}$ is approximated as

$$\mathbf{r}^{c}{}_{i}(t+dt) - \mathbf{r}^{u}{}_{i}(t) \approx -\frac{dt^2\lambda_k}{m_i}\frac{\partial\sigma_k(t)}{\partial\mathbf{r}_i}. \quad (69)$$

By inserting the above equation into eq. (66), one gets

$$\sigma^{u}{}_{k}(t+dt) = dt^2\lambda_k\sum_{i=1}^{N}\frac{1}{m_i}\frac{\partial\sigma_k(t+dt)}{\partial\mathbf{r}_i}\frac{\partial\sigma_k(t)}{\partial\mathbf{r}_i} \quad (70)$$

from which

$$\lambda_k dt^2 = \frac{\sigma^{u}{}_k(t+dt)}{\sum_{i=1}^{N} \frac{1}{m_i} \frac{\partial \sigma_k(t+dt)}{\partial \mathbf{r}_i} \frac{\partial \sigma_k(t)}{\partial \mathbf{r}_i}} . \tag{71}$$

In an MD simulation, the constraints are treated in succession during one cycle of the iteration and the process is repeated until all constraints have converged to the desired accuracy. An improvement of the SHAKE algorithm is the RATTLE algorithm, which was proposed by Andersen [19]. In RATTLE, the velocity-Verlet algorithm is employed to integrate the dynamical equations.

As was also stated above, there are several applications of MD simulations in polymer science. An example taken from a recent study of polymer melt viscoelasticity is presented in the following section.

VI. MD Applications to Polymer Melt Viscoelasticity

Despite its simplicity and unquestionable utility, a brute-force application of the atomistic MD technique to polymeric systems is problematic, due to the enormously large computation time needed to track the evolution of such systems for times comparable to their longest relaxation times [1],[2]. This is the well-known *problem of long relaxation times*. To overcome this problem, a number of approaches have been proposed over the last years. The first is to develop new, more efficient, "clever" algorithms, such as the multiple time step algorithms for the integration of equations of motion described in section IV, which has allowed extending the simulation to times almost an order of magnitude longer than what is usually achieved with conventional algorithms. The second is to increase the range of length scales simulated with MD by using a number of processors (nodes) and special parallelization techniques; such techniques are described in detail in the next section.

Alternatively, a hierarchical approach can be adopted, which uses information from many different levels of abstraction, ultimately connecting to the atomistic level, and a combination of different molecular simulation methods and theoretical approaches. Such a methodology can be followed, for example, for the atomistic MD simulation of the viscoelastic properties of polymer melts. The methodology has been described in a number of publications [20]-[13] and includes two variants: In the first, equilibrium atomistic MD simulations are conducted on model polymer melt configurations pre-equilibrated with the powerful connectivity-altering end-bridging Monte Carlo algorithm; the latter algorithm is not subject to the limitations associated with long relaxation times faced with MD. Dynamic as well as viscoelastic properties are then extracted either directly from the MD simulations or indirectly through a mapping of the atomistic trajectories accumulated in the course of the MD simulation onto an analytical coarse-grained model, [20]-[22] such as the Rouse model for unentangled melts or the reptation model for entangled melts. In the second variant, nonequilibrium MD simulations are conducted on model polymer melt configurations which have been pre-oriented and thoroughly equilibrated with a field-on MC method [23] which generates configurations representative of a melt under conditions of steady-state uniaxial elongational flow. The MD tracks the relaxation of the preoriented chains back to equilibrium upon cessation of the flow. In this case, again, the linear viscoelastic properties of the melt are extracted either directly by the simulation or indirectly by utilizing a mapping onto an analytical model coarse-grained [13].

VI.A. Study of polymer viscoelasticity through equilibrium MD simulations

In detail, the methodology followed for the prediction of the viscoelastic properties of polymer melts under equilibrium conditions is a three-stage hierarchical approach, whereby the dynamic properties of polymer melts are calculated through the following procedure:

a) First exhaustive end-bridging Monte Carlo (EBMC) simulations [23] are conducted to equilibrate the melts at all length scales. The EBMC algorithm employs moves that modify the connectivity among polymer segments, while preserving a prescribed (narrow) molecular weight distribution. It can thus equilibrate the long-length scale features of a polymer melt orders of magnitude more efficiently than MD or other MC methods, its relative efficiency increasing rapidly with increasing chain length.

b) Relaxed configurations thus obtained are subjected to equilibrium MD simulations to monitor their evolution in time and extract dynamic properties. During the atomistic MD simulations, a large number of dynamical trajectories are accumulated.

c) Finally, the trajectories accumulated are mapped onto theoretical mesoscopic (coarse-grained) models to extract the values of the parameters invoked in the mesoscopic model description of the same systems.

With the above methodology, atomistic MD simulations were performed on united-atom model linear polyethylene melts with molecular length ranging from $N = 24$ up to $N = 250$ in the canonical NVT ensemble ($T = 450$K, $P = 1$atm). To speed-up the MD simulations the multiple time step rRESPA algorithm, presented in section IV, was used. The overall simulation time ranged from 100 ns to 300 ns, depending on the chain lengths of the systems studied. Many of the dynamic properties (such as the self-diffusion coefficient D) were calculated directly from the MD simulations. Others, however, such as the zero-shear rate viscosity η_0, required mapping

atomistic MD data upon a mesoscopic theoretical model. As such one can choose the Rouse model for relatively short PE melts and the reptation model for the longer-chain melts.

Figure 3 shows the mean-square displacement of the chain center-of-mass, $\left\langle \left(R_{cm}(t) - R_{cm}(0)\right)^2 \right\rangle$, in time for the longer-chain systems, C_{156}, C_{200} and C_{250}. From the linear part of these curves the self-diffusion coefficient D can be obtained using the Einstein relation,

$$D = \lim_{t\to\infty} \frac{\left\langle \left(\mathbf{R}_{cm}(t) - \mathbf{R}_{cm}(0)\right)^2 \right\rangle}{6t} \tag{72}$$

Figure 4 presents predictions for the self-diffusion coefficient D as a function of mean chain length N. For comparison, also shown in the figure are experimental data [26]. Three distinct regions appear in the figure:

a) A region of small-MW, alkane-like behavior (N<60), where D follows a power-law dependence $D \sim M^{-b}$, with b>1. In this regime chain end effects, which can be described through a free volume theory, dominate system dynamics [21].

b) An intermediate, Rouse-like regime (from N=60-70 up to N=156) where $b \approx 1$. System dynamics in this regime is found to obey the Rouse model, at least for the overall relaxation of the chain [20].

c) A long chain-length, reptation-like regime (N>156), where chain diffusivity exhibits a dramatic slow down, $b \approx 2.5$. According to the original formulation of reptation theory, the latter exponent should be 2. Phenomena such as contour length fluctuations (CLF) and constraint release (CR) typically accelerate the escape of the chain from the tube, causing an increase in D and a decrease in η_0 [24]. A recently proposed theory that incorporates CLF and CR phenomena predicts a stronger exponent, between –2.2 and –2.3 [25]. These values agree with recent

experimental results for concentrated polymer solutions and melts, which suggest an exponent between –2.2 and –2.4 for a variety of polymer systems [26].

In contrast to D, the prediction of other viscoelastic properties, such as the friction coefficient ζ or the zero-shear rate viscosity η_0, require that the atomistic MD data be mapped upon a mesoscopic theoretical model. For unentangled polymer melts, such a model is the Rouse model, wherein a chain is envisioned as a set of Brownian particles connected by harmonic springs [27]-[24]. For entangled polymer melts, a better model that describes more accurately their dynamics is the tube or reptation model [24]. According to this, the motion of an individual chain is restricted by the surrounding chains within a tube defined by the overall chain contour or primitive path. During the lifetime of this tube, any lateral motion of the chain is quenched.

The Rouse model is formulated in terms of three parameters: the number of beads N, the length of the Kuhn segment b, and the monomeric friction coefficient ζ. The friction coefficient ζ can be calculated directly from the diffusion coefficient D through

$$\zeta = \frac{k_B T}{ND} \tag{73}$$

while the zero-shear rate viscosity η_0 can be calculated from the density ρ, the end-to-end distance $\langle R^2 \rangle$ and the diffusion coefficient D through

$$\eta_0 = \frac{\rho RT \left\langle R^2 \right\rangle}{36MD}. \tag{74}$$

Reptation theory is formulated in terms of four parameters: N, b, ζ, and the entanglement spacing (or, alternatively, the tube diameter) α. If α were known, ζ and η_0 could be calculated through:

$$\zeta = \frac{k_B T a^2}{3N \left\langle R^2 \right\rangle D} \tag{75}$$

and

$$\eta_0 = \frac{\rho RT}{36M}\frac{\langle R^2\rangle}{D}\frac{\langle R^2\rangle}{a^2}. \tag{76}$$

The calculation of the tube diameter α is a formidable task and can be addressed either through a direct topological analysis of accumulated polymer melt configurations thoroughly equilibrated with an efficient MC algorithm [29] or by utilizing a geometric mapping of atomistically represented chain configurations onto primitive paths [22],[30]. The latter mapping is realized through a projection operation involving a single parameter ξ, which governs the stiffness of the chain in the coarse-grained (primitive path) representation. The parameter ξ is mathematically defined as the ratio of the constants of two types of Hookean springs: The first type connects adjacent beads within the projected primitive path, and the second type connects the projected beads of the primitive path with the corresponding atomistic units [30]. Different values of ξ lead to different parameterizations, i.e., to different primitive paths and, consequently, to different values of the contour length L. Once a value for ξ has been chosen, the primitive path is fully defined which allows calculating the tube diameter a through the following equation of reptation theory

$$La = \langle R^2\rangle \tag{77}$$

To find the proper value of the projection parameter ξ, one can follow a self-consistent scheme based on the mean square displacement of the primitive path points $\phi(s,s;t)$ [22]. $\phi(s,s;t)$ is defined as

$$\phi(s,s;t) \equiv \left\langle \left(\mathbf{R}(s,t) - \mathbf{R}(s,0)\right)^2 \right\rangle, \tag{78}$$

where **R**(*s,t*) is the position vector of the *primitive* segment at contour length *s* at time *t*. and **R**(*s,*0) is the position vector of the *primitive* segment at contour length *s* at time 0. According to reptation theory

$$\phi(s,s;t) = 6Dt + \sum_{p=1}^{\infty} \frac{4\langle R^2 \rangle}{p^2\pi^2} \cos\left(\frac{p\pi s}{L}\right)^2 \left[1 - \exp\left(-tp^2/\tau_d\right)\right] \qquad (79)$$

where the sum is over all normal modes *p* and τ_d denotes the longest relaxation or disengagement time. For small times ($t<\tau_d$), $\phi(s,s;t)$ is dominated by the terms with large *p* and the above equation becomes

$$\phi(s,s;t) = 6Dt + \int_0^{\infty} dp \frac{4La}{p^2\pi^2} \frac{1}{2}\left(1 - \exp\left(-tp^2/\tau_d\right)\right) = 6Dt + 2\left(\frac{3}{\pi}\langle R^2 \rangle D\right)^{1/2} t^{1/2} \qquad (80)$$

Eq. (80) offers a nice way of mapping atomistic MD trajectories uniquely onto the reptation model, through a self-consistent calculation of the parameter ξ. First, a value of ξ is chosen and the mapping from the atomistic chain onto its primitive path is carried out by following the procedure described by Kröger *et al* [30]. Then, eq. (78) is used to calculate $\phi(s,s;t)$ for the primitive path points, averaged over all *s* values. For times $t<\tau_d$, the resulting curve is compared to that obtained from eq. (80), using the values of $<R^2>$ and *D* (long-time diffusivity of the centers of mass) calculated directly from the atomistic MD simulations. The procedure is repeated until convergence is achieved, that is until a ξ value is found for which the two curves coincide. This mapping is performed self-consistently, without any additional adjustable parameters or any experimental input. It allows a reliable estimation of the tube diameter α, by utilizing atomistically collected MD data only for times shorter than τ_d. Thus, the total duration of the MD simulations required is governed solely by the time needed reliably to

calculate the center-of-mass diffusion coefficient D. With the values of $<R^2>$, D and α, ζ and η_0 can be calculated using eqs. (75) and (76).

With the above procedure the tube diameter α was calculated to be $\alpha \sim 60$ Å for the longer-chain systems C_{200} and C_{250}, whereas for the shorter systems, $N < 200$, no proper value of the parameter ξ could be identified [22].

Figure 5 shows results for the monomeric friction factor ζ as a function of mean chain length N, over the entire range of molecular lengths studied. Filled circles depict results obtained by mapping atomistic MD data onto the Rouse model, whereas open circles depict results obtained from mapping the atomistic MD data onto the reptation model. According to its definition, ζ should be independent of chain length, its value determined solely by the chemical constitution of the melt. The figure shows clearly that, at around C_{156}, a change in the mechanism of the dynamics takes place, which cannot be accommodated by the Rouse model unless a chain-length dependent ζ is assumed. On the contrast, in this regime (N>156), the reptation model provides a consistent description of the system dynamics characterized by a constant (0.4×10^{-9} dyn s/cm) chain-length independent ζ value per methylene or methyl segment.

Figure 6 presents the zero-shear rate viscosity η_0 as a function of molecular weight for all systems studied here. For systems of chain length less than C_{156} the Rouse model, eq. (74), was used, whereas for the longer systems the reptation equation, eq. (76), was used. Also reported in the figure are experimental η_0 values from measurements [22] conducted in a series of linear monodisperse PE melts. The η_0 predictions from the reptation model were obtained using the value of $\alpha = 60$ Å for the entanglement spacing. The agreement of the simulation results with the experimental ones is remarkable in all systems studied.

VI.B. Study of polymer viscoelasticity through non-equilibrium MD simulations - Simulation of the Stress Relaxation Experiment

An alternative way to learn about viscoelastic properties of polymer melts in the linear regime is to conduct MD simulations of pre-oriented polymer melt configurations generated by a field-on MC algorithm. It involves three stages:

(a) First, a coarse-grained description of the polymer melt is invoked through the definition of the conformation tensor, $\tilde{\mathbf{c}}$, which is a global descriptor of the long length-scale conformation of polymer chains. The conformation tensor $\tilde{\mathbf{c}}$ is defined as the second moment tensor of the end-to-end distance vector of a polymer chain reduced by one third the unperturbed end-to-end distance and averaged over all chains in the system:

$$\tilde{\mathbf{c}}(t) = 3\left\langle \frac{\mathbf{R}(t)\mathbf{R}(t)}{\langle R^2 \rangle_0} \right\rangle \tag{81}$$

In the above equation, **R** stands for the end-to-end vector of a macromolecule and $\langle R^2 \rangle_0$ is the mean-squared magnitude of that vector in the equilibrium, quiescent state, where chains are unperturbed to an excellent approximation. With the above definition, a series of detailed atomistic MC simulations can be initiated on model melt systems at various values of the orienting thermodynamic field α_{xx}, starting from the zero value (α_{xx}=0, equilibrium, quiescent, field-free state) [23]. Thus, configurations are obtained from the MC method where the chains are oriented through a thermodynamic field.

(b) In the second stage, the isothermal relaxation of these configurations to thermodynamic equilibrium is monitored, keeping their dimension along the *x*- direction constant and the average normal pressure in the *y*- and *z*- directions equal to the atmospheric pressure. The experiment

simulated is that of stress relaxation upon cessation of a steady-state uniaxial elongational flow. The MD simulation takes place in the $NTL_x\sigma_{yy}\sigma_{zz}$ statistical ensemble discussed in section III. The macroscopic variables kept constant in this ensemble are the typical macroscopic constraints encountered in the process of fiber spinning at the end of the spinning operation, when the fibers are under constant extension and the stress σ_{xx} in the direction of pulling is allowed to relax from its initial value to the equilibrium, field-free value, equal to $-P_{ext}$ (i.e., $\sigma_{xx}(t \rightarrow \infty) = -P_{ext}$). In addition to monitoring the temporal evolution of the stress component $\sigma_{xx}(t)$ during the $NTL_x\sigma_{yy}\sigma_{zz}$ MD simulation, also recorded is the evolution of certain ensemble-averaged descriptors of the chain long length-scale configuration. These descriptors include the diagonal components of the chain conformation tensor (c_{xx}, c_{yy} and c_{zz}) and the chain mean-square end-to-end distance $\langle R^2 \rangle$.

(c) The third stage includes the development of expressions describing analytically the time evolution of these quantities by solving the Rouse model under the initial and boundary conditions corresponding to our atomistic computer experiment.

Results are presented here from averaging over about 100 $NTL_x\sigma_{yy}\sigma_{zz}$ MD trajectories for each stress relaxation experiment, initiated at ensembles of strained configurations of two PE melt systems: a 32-chain C_{24} and a 40-chain C_{78} PE melt.

Figures 7a and 7b show the time evolution of the diagonal components, c_{xx}, c_{yy} and c_{zz} of the conformation tensor for the C_{24} and C_{78} melts, respectively. For both systems, the initial value of c_{xx} is significantly higher than 1, whereas those of c_{yy} and c_{zz} are a little smaller than 1, indicative of the oriented conformations induced by the imposed steady-state elongational flow field α_{xx}. As time evolves, c_{xx} drops whereas c_{yy} and c_{zz} increase continuously, approaching the

steady-state, field-free value of 1, indicative of fully equilibrated, isotropic structures in the absence of any deforming or orienting field.

Figure 8 shows the time evolution of the stress tensor component σ_{xx}, for the C_{24} PE melt systems studied. The stress tensor is calculated in two ways. The first (thin solid line) tracks the evolution of σ_{xx} as obtained from applying the molecular virial theorem on the relaxing configurations and averaging over all dynamical trajectories. The second one (thick dashed line) uses the Helmholtz energy function and an affine deformation assumption for chain ends [23]. According to the latter approach, the stress tensor at every time t is calculated from the ensemble-averaged values of mass density ρ, conformation tensor c_{xx}, and partial derivative of the Helmholtz energy function with respect to c_{xx} at time t, through

$$\sigma_{xx}(t) = -P_{ext} + 2\frac{R}{M}T\rho(t)c_{xx}\left[\left.\frac{\partial\left(A/N_{ch}\right)}{\partial c_{xx}}\right|_{T,\rho,c_{[xx]}}\right]_{c_{xx}=c_{xx}(t)} \tag{82}$$

where P_{ext} denotes the equilibrium (atmospheric) pressure and M the number average molecular weight of the system. This approach tracks the evolution of σ_{xx} as obtained from applying the thermodynamic stress equation, eq. (82), based on the current values of c_{xx} and $\partial(A/N_{ch})/\partial c_{xx}$, the latter taken from the melt elasticity simulations presented in ref. [23]. As expected, in both figures the estimates based on the virial theorem are subject to much higher statistical uncertainty, owing to the fluctuations in the instantaneous configurations. Clearly, averaging over many configurations is needed in order to improve the statistical quality of the virial theorem results. Apart from high-frequency noise, the virial theorem results display an oscillatory character. When the noise and oscillations are smoothed out, the ensemble averaged stress $\sigma_{xx}(t)$ from the virial theorem is in very good agreement with the thermodynamic estimate obtained from c_{xx} and $\partial\left(A/N_{ch}\right)/\partial c_{xx}$. This is an important result, as it opens up the possibility

of calculating stress with high precision directly from ensemble average conformational properties, based on a free energy function accumulated via efficient MC runs. The transverse components σ_{yy} and σ_{zz} are displayed in Figure 9. Both σ_{yy} and σ_{zz} fluctuate continuously around the constant value $-P_{atm}$, as required by the macroscopic restrictions placed on the $NTL_x\sigma_{yy}\sigma_{zz}$ ensemble. Similar are the plots for the C_{78} system. Relaxation times extracted from those stress relaxation computer experiments are identical to those determined from equilibrium MD simulations (section VI.A).

VII. Parallel MD Simulations of Polymer Systems

In the previous sections, we presented two different approaches for addressing the problem of long relaxation times, which plagues the conventional MD method: the multiple time step algorithm and a hierarchical methodology that leads to the prediction of the dynamic and rheological properties of polymer melts by mapping simulation data onto analytical theories. Enhanced MD simulation algorithms can further be developed by resorting to special parallelization techniques that allow sharing the total simulation load over a number of processors or nodes. The key idea in all these techniques is to compute forces on each atom or molecule and to perform corresponding velocity/position updates independently but simultaneously for all atoms. It is also desired that the force computations be evenly divided across the processors so as to achieve the maximum parallelism.

MD simulations of polymer systems, in particular, require computation of two kinds of interactions: *bonded* forces (bond length stretching, bond angle bending, torsional) and *non-bonded* van der Waals and Coulombic forces. Parallel techniques developed [31]-[33] include the *atom-decomposition* (or *replicated data*) method, the *force-decomposition* method and the

spatial (domain)-decomposition method. The three methods differ in how atom coordinates are distributed among the processors to perform the necessary computations. Although all of the methods scale optimally with respect to computation, their different data layouts incur different inter-processor communication costs which affect the overall performance of each method.

VII.A. Parallel MD algorithms

Here we will focus on a discussion of algorithms for parallelizing MD simulations with short-range interactions where the non-bonded forces are truncated, so that each atom interacts only with other atoms within a specified cutoff distance. More accurate, MD models with long-range forces are more expensive to deal with. Thus, special techniques are required for parallelizing MD simulations with long-range interactions.

a. Atom-Decomposition (Replicated-Data) Method

The most commonly used technique for parallelizing MD simulations of molecular systems is the *replicated-data* (RD) method [33]. In the literature there are numerous parallel algorithms and simulations that have been developed based on this approach [34]. The key idea of this method is that each processor is assigned a subset of atoms and updates their positions and velocities during the entire simulation, regardless of where they move in the physical domain.

Let us consider a polymeric system with a total number of atoms N, where both intramolecular and intermolecular interactions are present. The system is simulated in a parallel system of P processors. We first define $\boldsymbol{x}$ and $\boldsymbol{f}$ as vectors of length N, which store the position and total force on each atom, respectively. We also define the force matrix **F**, of dimensions NxN, with F_{ij} the force on atom i due to j. In the RD method each processor is assigned a group

of N/P atoms at the beginning of the simulation. Each processor is also assigned a sub-block of the force matrix $\boldsymbol{F}$ which consists of N/P rows of the matrix, as shown in Figure 10. If z indexes the processors from 0 to P-1, then processor z computes forces in the F_z sub-block of rows. It also is assigned the corresponding position and force sub-vectors of length N/P denoted as $\boldsymbol{x_z}$ and $\boldsymbol{f_z}$ .The computation of the non-bonded force F_{ij} requires only the two atom positions, x_i and x_j. But to compute all the forces in F_z, processor P_z will need the positions of many atoms owned by other processors. In Figure 10 this is represented by having the horizontal vector **x** at the top of the figure span all the columns of $\boldsymbol{F}$, implying that each processor must store a copy of the *entire* **x** vector, hence the name replicated-data. This also implies that at each timestep each processor must receive updated atom positions from all other processors and send the positions of its own atoms to all other processors, an operation called *all-to-all* communication.

A single timestep of an RD algorithm comprises the following steps:

(1) First every processor z *computes its own part of the non-bonded forces* F_z. In MD simulations this is typically done using neighbor lists to calculate non-bonded interactions only with the neighboring atoms. In an analogous manner with the serial algorithm, each processor would construct lists for its sub-block F_z once every few time steps. To take advantage of Newton's 3rd law (that $F_{ij} = - F_{ji}$), each processor also stores a copy of the entire force vector $\boldsymbol{f}$. As each pairwise non-bonded force between atoms i and j is calculated, the force component is summed both for atom i and negated for atom j. Next, the bonded forces are computed. Since each processor z knows the positions of all atoms, it can compute the non-bonded forces for its sub-vector x_z and sum the resulting forces into its local copy of $\boldsymbol{f}$.

Calculation of both bonded and non-bonded forces scale as N/P, i.e. with the number of non-bonded interactions computed by each processor.

(2) In this step, the local force vectors are summed across all processors in such a way that each processor ends up with the total force on each of its N/P atoms. This is the sub-vector f_z. This force summation is a parallel communication over all processors, an operation known as *fold* [35]. In the literature there are various algorithms that have been developed for optimizing this operation. The key characteristic is that each processor must receive N/P values from every other processor to sum the total force on its atoms. This requires total communication of N times N/P; i.e. the fold operation scales as N.

(3) Once each processor has the total force on its sub-vector x_z in step (3) it can update the positions and the velocities of each atom (*integration step*) with no communication at all. Thus, this operation scales as N/P.

(4) Finally the updated positions of each processor $\mathbf{x_x}$ should be shared among all P processors. Each processor must send N/P positions to every other processor. This operation is known as *expand* [35] and scales as N.

A crucial aspect in any parallel algorithm is the issue of load-balance. This concerns the amount of work performed by each processor during the entire simulation, which ideally should be the same for all processors. As we saw before, the RD algorithm divides the MD force computation (the most time consuming part in typical MD simulations) and integration evenly across all processors. This means that steps (1) and (3) scale optimally as N/P. Load-balance will be good so long as each processor's subset of atoms interacts with roughly the same number of neighbor atoms. This usually occurs naturally if the atom density is uniform across the

simulation domain (e.g., bulk simulations). In a different case (e.g., polymer chains at interfaces or adsorbed polymers) load-balance can be achieved by randomizing the order of the atoms at the start of the simulation or by adjusting the size of the subset of each processor dynamically during the simulation to tune the load-balance; these are called dynamic load balancing techniques [36].

In summary, the RD algorithm divides evenly the force computation across all processors. At the same time, its simplicity makes it easy to implement in existing codes. However, the algorithm requires global communication in steps (2) and (4), as each processor must acquire information from all other processors. This communication scales as N, independently of the number of processors P. Practically this limits the number of processors that can be used effectively.

b. Force-Decomposition Method

The next parallel MD algorithm discussed here is based on a block-decomposition of the force matrix rather than the row-wise decomposition used in the RD algorithm. The partitioning of the force matrix $\boldsymbol{F}$ is shown in Figure 11 and the algorithm is called *force-decomposition* (FD) algorithm [37]. The method has its origin in block-decomposition of matrices, which is commonly encountered in linear algebra algorithms for parallel machines.

The block-decomposition, shown in Figure 11, is actually done on a permuted force matrix $\boldsymbol{F'}$, which is formed by rearranging the columns of the original $\boldsymbol{F}$ in a particular way. The (ij) element of $\boldsymbol{F}$ is the force acting on atom i in vector $\boldsymbol{x}$ due to atom j in the permuted vector $\mathbf{x'}$. Now the F'_z sub-block owned by each processor z is of size $(N/P^{1/2})$x$(N/P^{1/2})$. As shown in the figure, to compute the non-bonded forces in F'_z, processor z must know one $N/P^{1/2}$-length piece

of each of the $\boldsymbol{x}$ and $\boldsymbol{x}'$ vectors, i.e. the sub-vectors x_a and x'_b. As these elements are computed they will be accumulated into the corresponding sub-blocks f_a and f'_b. The subscripts a and b each run from 0 to $P^{1/2}$ and reference the row and the column position occupied by processor z.

As in the RD algorithm, each processor has updated copies of the atom positions x_a and x'_b needed at the beginning of the timestep. A single timestep of the FD algorithm consists of the following steps:

(1) The first step is the same as that of the RD algorithm. First the non-bonded forces F'_z are computed. The result is summed into both f_a and f'_b. Next, each processor computes a fraction N/P of the bonded interactions. A critical point here is that in a pre-processing step of the run, we should guarantee that each processor knows all the atom positions needed for the bonded (intramolecular) interactions. This steps again scales as N/P.

(2) Step (2) is also the same as that of the RD algorithm. The key difference is that now the total force on atom i is the sum of the elements in row i of the force matrix minus the sum of elements in column i', where i' is the permuted position of column i. Thus this step performs a *fold of f_a (f'_b) within each row (column) of processors to sum these contributions*. The important point is that now the vector f_a (f'_b) being folded is only of length ($N/P^{1/2}$) and only the $P^{1/2}$ elements in one row (column) are participating in the fold. Thus, this operation scales as $N/P^{1/2}$ instead of N as in the RD algorithm. Finally, the two contributions are jointed to yield the total forces f_z (f'_z) on the atoms owned by processor P_z.

(3) The processor can now perform the integration for its own atoms, as in the RD case. This operation scales as N/P.

(4) Step (4) *shares* these updated positions with all the processors that will need them for the next timestep. As with the fold operation the processors in each row (column) expand their x_a (x'_b) sub-vectors within the row (column) so that each acquires the entire x_a (x'_b). This operation scales again as $N/P^{1/2}$ instead of N, as in the RD algorithm.

The FD algorithm, as the RD algorithm, divides the force computation evenly across all processors. The key point is that the communication and memory costs in steps (2) and (4) scale as $N/P^{1/2}$, rather than as N as in the RD algorithm. When large number of processors are used, this can be very important. At the same time, although more steps are needed, the FD algorithm retains the overall simplicity and structure of the RD method.

c. Domain-Decomposition Method

The third parallel method discussed here for MD simulations of systems with short-range interactions is the domain-decomposition (DD) method [31],[38]-[40]. In this method the physical simulation box is divided into small 3D boxes, one for each processor. The partitioning of a simulation box of length L in a DD algorithm is shown in Figure 12 (2D projection). Now each processor z owns a box labeled B_z with edge length L_z ($L_z = L/P$) and will update the positions of all atoms within its own box, x_z, at each timestep. Atoms are reassigned to new processors as they are moving through the physical domain. In order to compute the forces on its atoms, a processor must know the positions of atoms in nearby boxes (processors), y_z. Thus the communication required is global in contrast to total in the AD and FD algorithm. As it computes the force f_z on its atoms, the processor will also compute components of forces f''_z on the nearby atoms.

A single timestep of a DD algorithm consists of the following steps:

(1) The first step concerns, for each processor P_z, the *calculation of bonded and non-bonded forces for atoms within box* B_z. This step scales with the numbers of atoms N/P per processor.

(2) In step (2) the forces g_z *are shared with the processors owning neighboring boxes*. The received forces are summed with the previously computed f_z to create the total force on the atoms owned by the processor. The amount of data exchanged in this operation (and consequently the scaling of this step) is a function of the force cutoff distance and box length.

(3) After computing f_z, the atom positions x_z are updated. This operation also scales as N/P.

(4) Next the *updated positions are communicated to processors owning neighboring boxes* so that all processors can update their y_z list of nearby atoms

(5) Finally, periodically, atoms that have left box B_z are *moved* into the appropriate new processor.

The scaling of steps (1) and (3) in the DD algorithm is again the optimal N/P. The communication cost involved in steps (2), (5) and (6) is more complicated. It proves to be dependent on the relative value of the cut-off distance r_c in comparison with the sub-domain edge length L_z [31]. More specifically, if $L_z > r_c$ the scaling goes as the surface-to-volume ratio $(N/P)^{2/3}$. If $L_z \approx r_c$, then the communication scales as N/P. In practice, however, in the MD simulations of polymer systems there are several obstacles to minimizing communication costs in the DD algorithm.

- If Coulombic interactions are present then because of the $1/r$ dependence long cutoffs should be used. Thus $L_z < r_c$ and extra communication is needed in steps (3) and (5). Data from many neighboring boxes must be exchanged and the communication operation scales as r_c^3. Special techniques such as Ewald summation, particle-particle or particle-mesh methods can be implemented in parallel [41].
- As atoms move to new processors in step (6) the molecular connectivity information should be exchanged and updated between processors. This requires extra communication cost, depending on the type of the bonded interactions.
- If macromolecular systems are simulated uniformly in a simulation domain, then all boxes have a roughly equal number of atoms (and surrounding atoms); load-balance occurs. This will not be the case if the physical domain is non-uniform (e.g., for polymers in vacuum or with surrounding solvent). In this case it is not trivial to divide the simulation domain so that each processor has an equal number of atoms. Sophisticated load-balancing algorithms have been developed [36] to partition an irregular or non-uniformly dense physical domain, but the result is sub-domains which are irregular in shape, or connected in an irregular fashion to their neighboring boxes. In both cases the communication operation between processors becomes more costly. If the physical atom density changes with time, then the subject of load-balance becomes more problematic. Dynamic load-balancing schemes are again needed, which require additional computational time and data transfer.

In general, the DD algorithm is more difficult to integrate to existing serial codes than the RD and FD ones. This fact, coupled with the specific problems for macromolecular systems, has made DD implementations less common than RD in the simulations of polymer systems. For

simulations with very large systems, however, and in terms of optimal communication scaling, DD is more advantageous. It allows distributing the computational work over a large number of processors, and this explains why it is nowadays preferred in commercial MD packages such as LAMMPS [41],[42], NAMD [43] and AMBER.

VII.B. Efficiency - examples

Examples of applications of the above techniques can be found in various articles that describe parallel MD applications for systems consisting of many thousands (or even millions) of atoms performed on a large number of processors [31],[40]. Here we focus on the implementation of the parallel algorithm to the united-atom polyethylene melt MD simulations described in the previous section [44].

A critical point in any parallel implementation of an existing serial code is the percentage of the code that can be parallelized. To measure the performance of parallel implementations of existing sequential algorithms, the *speed-up* parameter S is used. This is defined as the ratio of the execution time of the serial (sequential) algorithm on a single processor to the execution time of the parallel algorithm running on P processors

$$S(P) = \frac{\tau_s}{\tau_p} \tag{83}$$

where τ_s and τ_p denote the execution time of the algorithms on one and P processors, respectively. For a fully parallel code running on P processors, $\tau_p = \tau_s / P$ and $S(P)=P$ (linear speedup).

A typical example where parallel implementation of atomistic MD algorithms can substantially speed up code execution is in $NTL_x\sigma_{yy}\sigma_{zz}$ simulations described in section VI.B (stress relaxation simulations). In this case the simulated systems are independent and

parallelism is straightforward to achieve by assigning each relaxing configuration to a different node (processor). Since no data communication between different systems is required in this case, excellent speed-up is achieved.

This has been verified with trial runs on a Cray T3E 900 machine at the Edinburgh Parallel Computing Center (EPCC) using the standard MPI approach [45]. Figure 14 presents a graph of the speed-up of the (parallel) code for a number of simulations, each one being executed with a model system containing 40 chains of C_{78} PE melt. As was expected, the speed-up is practically perfect (linear).

Figure 14 presents the corresponding speedup graph for a system containing 24 chains of a C_{1000} PE melt (a total of 24000 atoms). The runs were executed on a shared memory machine (Sun Enterprise 3000/3500 cluster at EPCC) using OpenMP Fortran [46], which is suitable for developing parallel MD applications in machines with shared memory architecture. Results are shown from parallel runs with different numbers of processors, ranging from 1 to 8 (dotted line) [44]. For comparison, the optimal (linear) speed-up is also shown (solid line). A speed-up of 5.5 is easily reached by using only 8 processors.

VIII.C Parallel Tempering

A trend in modern MD (and MC) simulations is to enhance system equilibration at low temperatures by the use of novel parallel techniques. One such a technique which has recently attracted considerable attention in different variations is “parallel tempering” (PT) [47],[48]. PT was introduced in the context of spin glass simulations, but the real efficiency of the method was demonstrated in a variety of cases, such as in the study of the conformational properties of complex biological molecules [49],[50]. Sugita and Okamoto [51] used a similar method called

"replica exchange molecular dynamics" to simulate protein folding and overcome the multiple-minima problem. They also used multiple-histogram reweighting techniques to calculate thermodynamic quantities in a wide temperature range. Recently, Yamamoto and Kob [52] used the same method to equilibrate a two-component Lennard-Jones mixture in its supercooled state. They found that the replica-exchange MC method is 10-100 times more efficient than the usual canonical molecular dynamics simulation.

PT has also been used successfully to perform ergodic simulations with Lennard-Jones clusters in the canonical and microcanonical ensembles [53]. Using simulated tempering as well as PT, Irbäck and Sandelin studied the phase behavior of single homopolymers in a simple hydrophobic/hydrophilic off-lattice model [54]. Q. Yan and de Pablo [55] used multidimensional PT in the context of an expanded grand canonical ensemble to simulate polymer solutions and blends on a cubic lattice. They indicated that the new algorithm, which results from the combination of a biased, open ensemble and PT performs more efficiently than previously available techniques. In the context of atomistic simulations PT has been employed in a recent study by Bedrov and Smith [56] who report parallel tempering molecular dynamics simulations of atomistic 1-4 polybutadiene polymer melts, in the 323K-473K temperature domain at atmospheric pressure. It has also been employed in the context of MC strategies in order to access the regime of small temperatures in the simulations of cis-1,4 polyisoprene melts [57]. In both of these works, it was shown that for a polymer melt well above the glass transition temperature, PT ensures a thorough sampling of configuration space, which is much more effective than afforded by conventional MD or MC simulation methods.

In PT, not a single system but an extended ensemble of n systems, labeled as i=1,...,n, is considered. Each such system is viewed as a copy of the original system, equilibrated, however,

at a different temperature T_i, i=1,...,n, with $T_1 < T_2 < ... < T_n$. The partition function of this extended ensemble is given by

$$Q = \prod_{i=1}^{n} Q_i \tag{84}$$

where Q_i denotes the individual partition function of the ith system in its relevant statistical ensemble. The strategy of simultaneously equilibrating not *a single* but *a number* of systems at different temperatures accelerates the equilibration of the lowest temperature systems by accepting configurations from the higher temperature systems, for which the rate of the system equilibration is higher. Thus, in the PT method, configurations are swapped between systems being equilibrated at adjacent temperatures. The acceptance probability of a swapping move between system configurations i and j=i+1 within a parallel tempering series of *NPT* simulations is given by:

$$\begin{aligned} acc[(i,j) \rightarrow (j,i)] &= \min[1, \exp\{-\beta_i(U_j + P_jV_j) - \beta_j(U_i + P_iV_i) + \\ &\quad + \beta_i(U_i + P_iV_i) + \beta_j(U_j + P_jV_j)\}] \\ &= \min[1, \exp\{\Delta\beta\Delta(U + PV)\}] \end{aligned} \tag{85}$$

with U, P, V symbolizing the potential energy function, pressure and volume, respectively [57].

For swapping to occur between two systems, it is important that their U+PV "instantaneous enthalpy" histograms overlap, since only then can there be a non-zero probability to exchange configurations. In this case, the success rate of configuration swapping can increase by performing the simulation with smaller size-systems characterized by larger (U+PV) fluctuations or with systems whose temperatures are not too wide apart.

PT is ideal to implement in parallel architectures by assigning each system to a single node, and by using standard MPI libraries [45] for inter-node communication.

BIBLIOGRAPHY

[1] K. Binder, Editor, *Monte Carlo and Molecular Dynamics Simulations in Polymer Science*; Oxford University Press: New York, 1995.

[2] M.P. Allen and D.J. Tildesley, *Computer Simulation of Liquids*; Oxford University Press: Oxford, 1987.

[3] D. Frenkel and B. Smith, *Understanding Molecular Simulations: From Algorithms to Applications*; Academic Press: New York, 1996.

[4] G. Ciccoti and W.G. Hoover, Editors, *Molecular Dynamics Simulations of Statistical Mechanics Systems*, Proceedings of the 97th Int. "Enrico Fermi" School of Physics, North Holland, Amsterdam, 1986.

[5] H. Goldstein, *Classical Mechanics*, Addison-Wesley, MA, 1980.

[6] W.C. Swope, H.C. Andersen, P.H. Berens, and K.R. Wilson, *J. Chem. Phys.*, **1982**, *76*, 637-649.

[7] S. Nosé, *Mol. Phys.* **1984**, *52*, 255-268.

[8] W.G. Hoover, *Phys. Rev. A* **1985**, *31*, 1695-1697.

[9] H.C. Andersen, *J. Chem. Phys.*, **1980**, *72*, 2384-2393.

[10] M. Parrinello and A. Rahman, *Phys. Rev. Lett.*, **1980**, *45*, 1196-1199.

[11] H.J.C. Berendsen, J.P.M. Postma, W.F. van Gunsteren, A. DiNola, and J.R. Haak, *J.Chem.Phys.* **1984**, *81*, 3684-3690.

[12] L. Yang, D.J. Srolovitz, and A.F. Yee, *J. Chem. Phys.* **1997,** *107*, 4396-4407.

[13] V.A. Harmandaris, V.G. Mavrantzas, and D.N. Theodorou, *Macromolecules* **2000**, *33*, 8062-8076.

[14] B.J. Palmer, *J. Comp. Phys.* **1993**, *104*, 470-472.

[15] M. Tuckerman, B. J. Berne, and G. J. Martyna, *J. Chem. Phys.* **1992**, *97*, 1990-2001.

[16] G. J. Martyna, M. E. Tuckerman, D. J. Tobias, and M. L. Klein, *Mol. Phys.* **1996**, *87*, 1117-1157.

[17] R. Edberg, D.J. Evans, and G.P. Morriss, *J. Chem. Phys.* **1986**, *84*, 6933-6939.

[18] J. P. Ryckaert, G. Ciccoti, and H. J.C. Berendsen, *J. Comp. Phys.* **1977**, *101*, 327.

[19] H. C. Andersen, *J. Comp. Phys.* **1983**, *52*, 24-34.

[20] V.A. Harmandaris, V.G. Mavrantzas, and D.N. Theodorou, *Macromolecules* **1998**, *31*, 7934-7943.

[21] V.A. Harmandaris, M. Doxastakis, V.G. Mavrantzas, and D.N. Theodorou, *J. Chem. Phys.* **2002**, *116*, 436-446.

[22] V.A. Harmandaris, V.G. Mavrantzas, D.N. Theodorou, M. Kröger, J. Ramírez, H.C. Öttinger, and D. Vlassopoulos, *Macromolecules*, **2003**, 36,1376-1387.

[23] V.G. Mavrantzas and D.N. Theodorou, *Macromolecules* **1998**, *31*, 6310-6332; V.G. Mavrantzas and H.C. Öttinger, *Macromolecules* **2002**, *35*, 960-975.

[24] M. Doi and S.F. Edwards, *The Theory of Polymer Dynamics*, Claredon, Oxford, 1986.

[25] S.T. Milner and T.C.B. McLeish, *Phys. Rev. Lett.* **1998**, *81*, 725-728.

[26] T.P. Lodge, *Phys. Rev. Lett*. **1999**, *83*, 3218-3221.

[27] J.D. Ferry, *Viscoelastic Properties of Polymers*, J. Wiley & sons, New York, 1980.

[28] J. Mark, Editor, *Physical Properties of Polymers*, American Chemical Society, Washington, 1984.

[29] A. Uhlherr, M. Doxastakis, V.G. Mavrantzas, D.N. Theodorou, S.J. Leak, N.E. Adam, P.E. Nyberg, "Atomic structure of a high polymer melt", *Europhys. Lett.*, **2002**, *57*, 506-511.

[30] M. Kröger, J. Ramírez, and H.C. Öttinger, *Polymer* **2002**, *43*, 477-487.

[31] S. Plimpton, *J. Comp. Phys.* **1995**, *117*, 1-19.

[32] S. Gupta, *Comp. Phys. Comm.* **1992**, *70*, 243-270.

[33] W. Smith, *Comp. Phys. Comm*. **1991**, *62*, 229-248.

[34] W. Smith and T.R. Forester, *Comp. Phys. Comm*. **1994**, *79*, 52-62.

[35] G.C. Fox, M.A. Johnson, G.A. Lyzenga, S.W. Otto, J.K. Salmon, and D.W. Walker, *Solving Problems on Concurrent Processors: Volume 1*. Prentice Hall, Englewood Cliffs, NJ, 1988.

[36] Y. Deng, R. Peiers, and C. Rivera, *J. Comp. Phys.*, **2000**, *161*, 250-263.

[37] B. Hendrickson and S. Plimpton, *J. Parr. Distr. Comp.* **1995**, *27*, 15-25.

[38] T. W. Clark, R. V. Hanxleden, J. A. McCammon, and L. R. Scott, in *Proc. Scalable High Performance Computing Conference-94*, pages 95-102, IEEE Computer Society Press, 1994.

[39] D. Brown, J.H.R. Clarke, M. Okuda, T. Yamazaki, *Comp. Phys. Commun.* **1994**, *83*, 1-13.

[40] M. Pütz and A. Kolb, *Comp. Phys. Commun.* **1998**, *113*, 145-167.

[41] S. J. Plimpton, R. Pollock, and M. Stevens, in *Proc. of the Eighth SIAM Conference on Parallel Processing for Scientific Computing*, Minneapolis, MN, March 1997.

[42] http://www.cs.sandia.gov/~sjplimp/lammps.html

[43] http://www.ks.uiuc.edu/Research/namd

[44] Unpublished data collected at EPCC under the TRACS programme during the period 10/01/-20/02/2000.

[45] http://www.mpi-forum.org/

[46] http://www.openmp.org/

[47] M. E. J. Newman and G. T. Barkema, *Monte Carlo in Statistical Physics* Clarendon, Oxford, 1999.

[48] C. J. Geyer, Computing Science and Statistics: Proceedings of the 23rd Symposium on the Interface, 156 1991.

[49] U.H.E. Hansmann, *Chem. Phys. Lett.* **1997**, *281*, 140-150.

[50] M.G. Wu and M.W. Deem, *Mol. Phys.* **1999**, *97*, 559-580.

[51] Y. Sugita and Y. Okamoto, *Chem. Phys. Lett.* **1999**, *314*, 141-151.

[52] R. Yamamoto and W. Kob, *Phys. Rev. E* **2000**, *61*, 5473-5476.

[53] J.P. Neirotti, F. Calvo, D.L. Freeman, and J.D. Doll, *J. Chem. Phys.* **2000**, *112***,** 10340-10349; F. Calvo, J.P. Neirotti, D.L. Freeman, and J.D. Doll, *J. Chem. Phys.* **2000**, *112***,** 10350-10357.

[54] A. Irbäck and E. Sandelin, *J. Chem. Phys.* **1999**, *110*, 12256-12262.

[55] Q. Yan and J. J. de Pablo, *J. Chem. Phys.* **2000**, *113*, 1276-1282.

[56] D. Bedrov and G.D. Smith, *J. Chem. Phys.*, **2001**, *115*, 1121-1124.

[57] M. Doxastakis, V.G. Mavrantzas, and D.N. Theodorou, *J. Chem. Phys.* **2001**, *115*, 11352-11361.

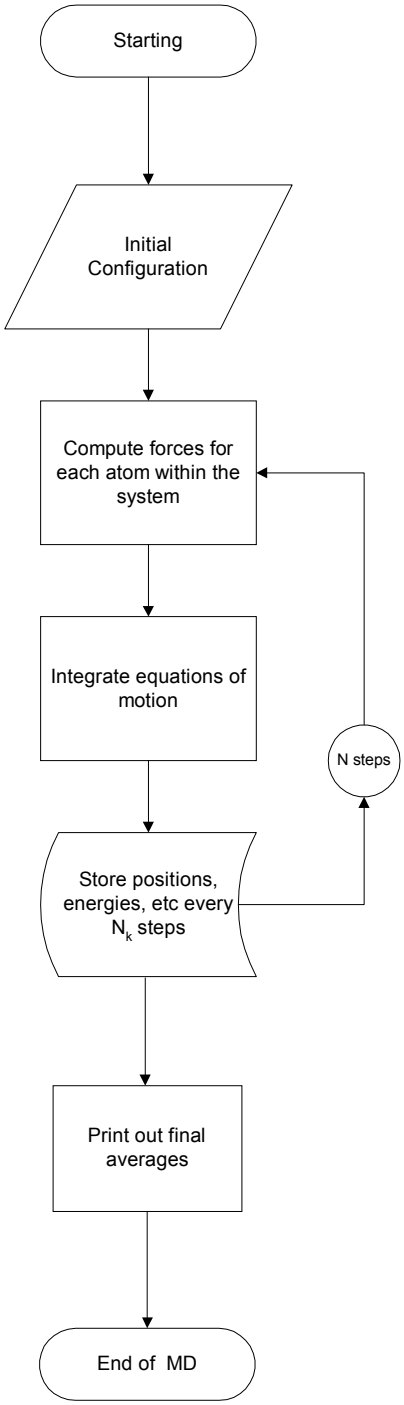

Figure 1 A simple flow diagram of a standard MD algorithm.

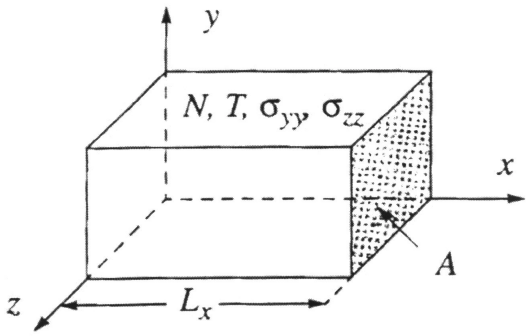

Figure 2 The $NTL_x\sigma_{yy}\sigma_{zz}$ statistical ensemble (after [12]).

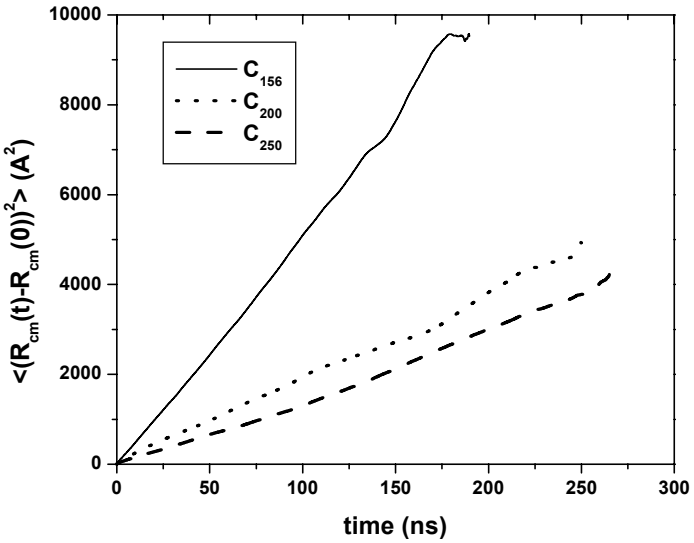

Figure 3 Mean square displacement of the center of mass for the C_{156} (solid), C_{200} (dot) and C_{250} (dashed) systems.

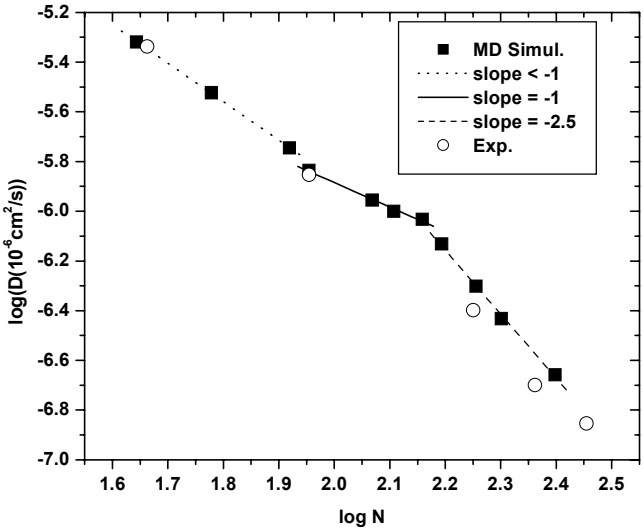

Figure 4 Predicted and experimental [26] self-diffusion coefficients D vs chain length N in a log-log plot (T=450K, P=1atm).

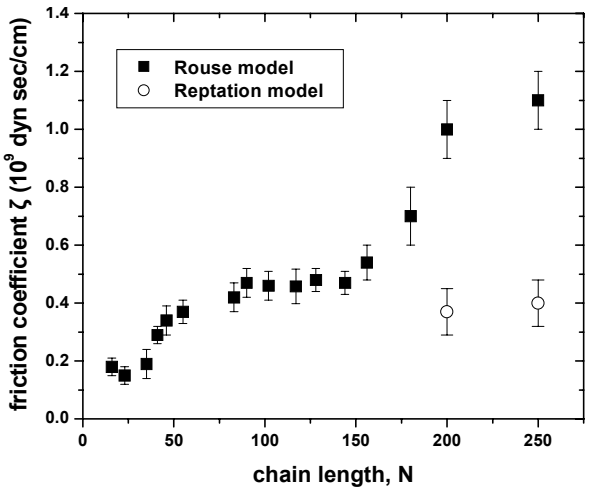

Figure 5 Monomer friction coefficient ζ vs chain length N, obtained from mapping the atomistic MD data onto the Rouse model (squares) or the reptation model (circles).

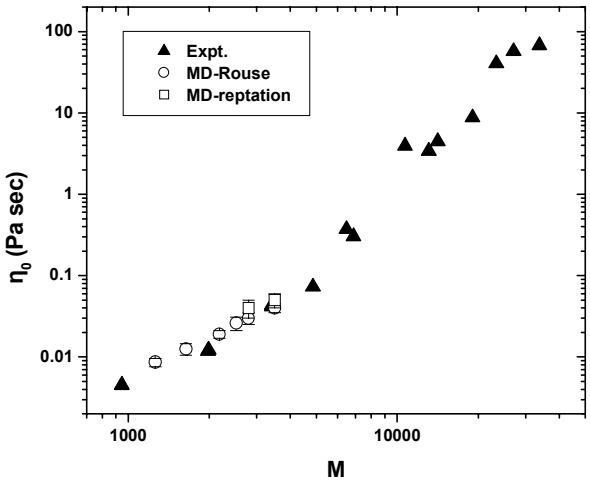

Figure 6 Zero-shear rate viscosity η_0 vs molecular weight M, obtained from the MD simulation and the Rouse model for small M (circles) or the reptation model for the high M (squares). Also shown in the figure are experimentally obtained η_0 values (triangles).

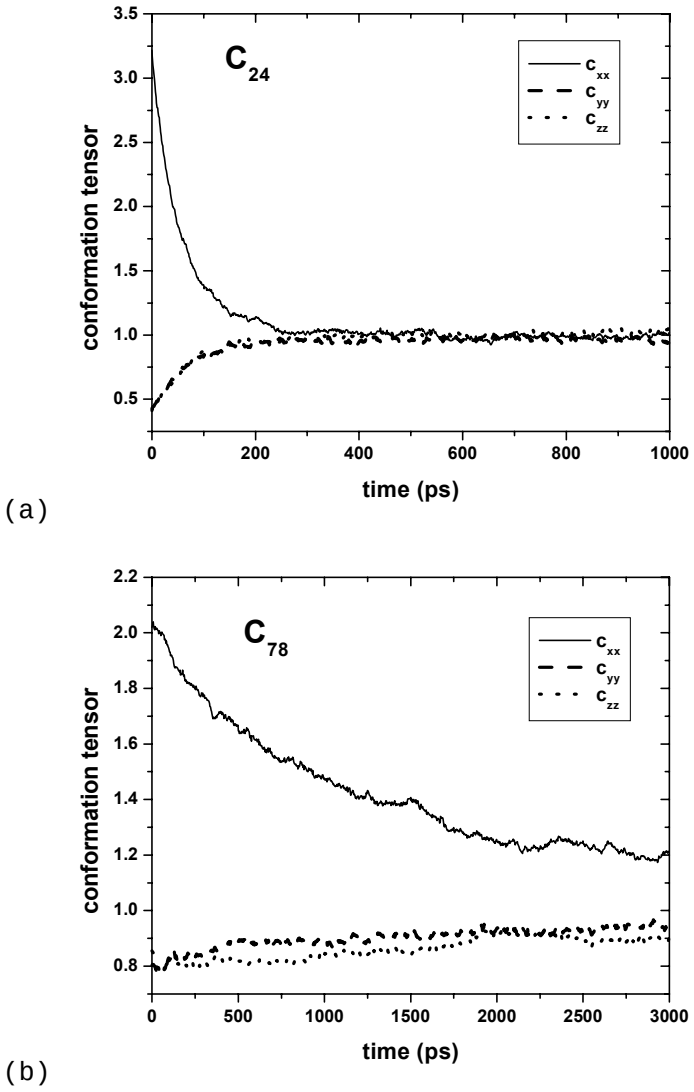

Figure 7 Evolution of the diagonal components c_{xx}, c_{yy} and c_{zz} of the conformation tensor c with time t for (a) the C_{24} and (b) the C_{78} PE melt systems. Results are averaged over all $NTL_x\sigma_{yy}\sigma_{zz}$ trajectories (T = 450 K, P_{ext} = 1 atm).

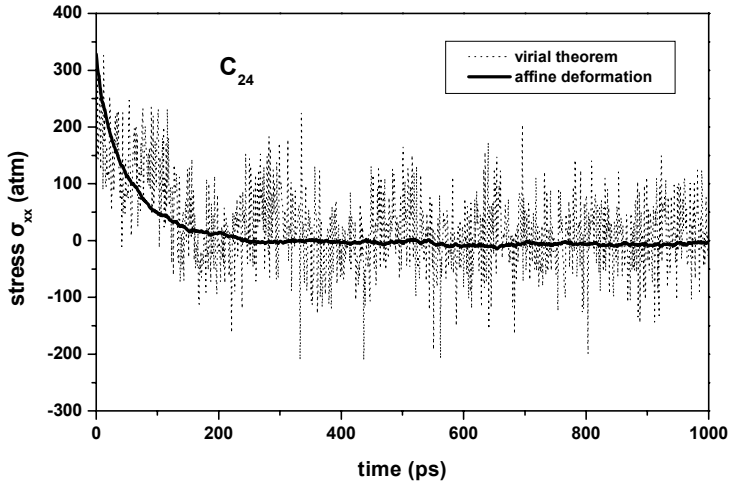

Figure 8 Evolution of the component σ_{xx} of the stress tensor with time t for the C_{24} system. The results at every time t have been obtained by applying either the virial theorem and averaging over all dynamical trajectories (dotted line) or by using a thermodynamic expression based for the free energy as a function of the conformation tensor (thick solid line) (T = 450 K, P_{ext} = 1 atm).

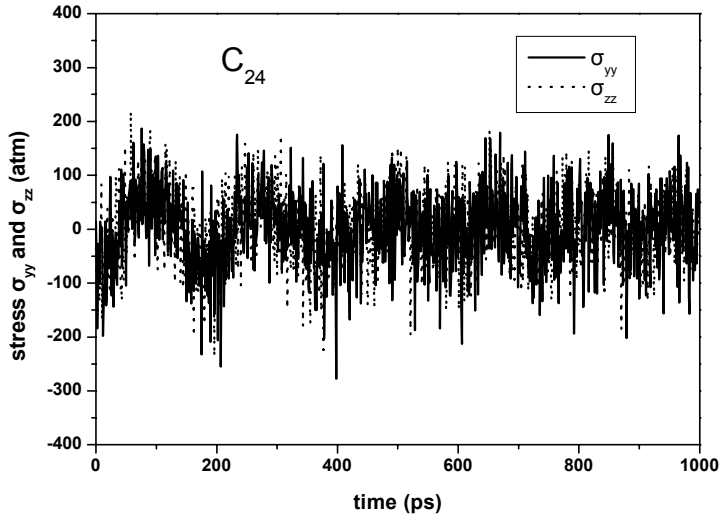

Figure 9 Evolution of the components σ_{yy} and σ_{zz} of the stress tensor with time t for the C_{24} system (T = 450 K, P_{ext} = 1 atm).

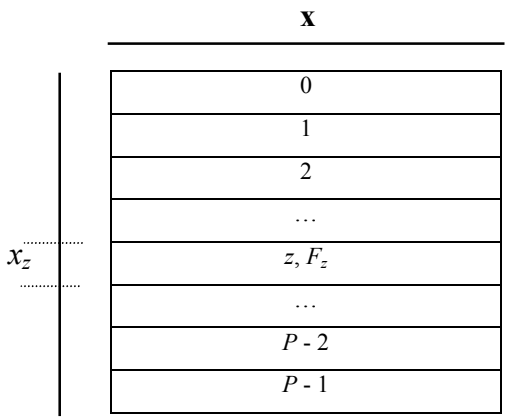

Figure 10 Division of the force matrix among P processors in the atom-decomposition method (after [31]).

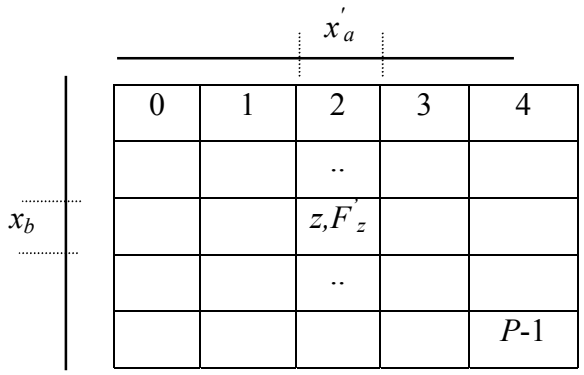

Figure 11 The division of the force matrix among *P* processors in the force-decomposition method (after [37]).

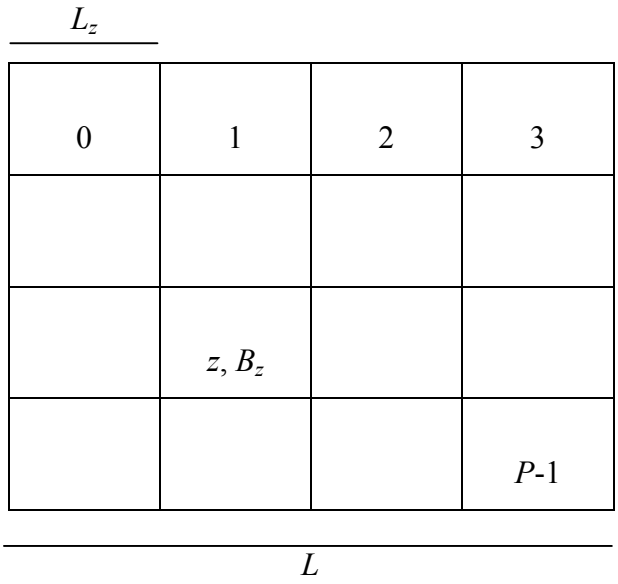

Figure 12 Partitioning of the simulation domain in a DD algorithm.

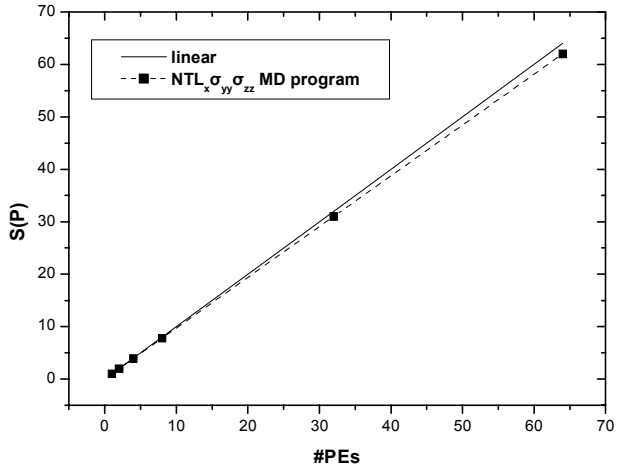

Figure 13 Speed-up graph of the parallelization of the $NTL_x\sigma_{yy}\sigma_{zz}$ MD simulation runs and the optimal (linear) speed-up.

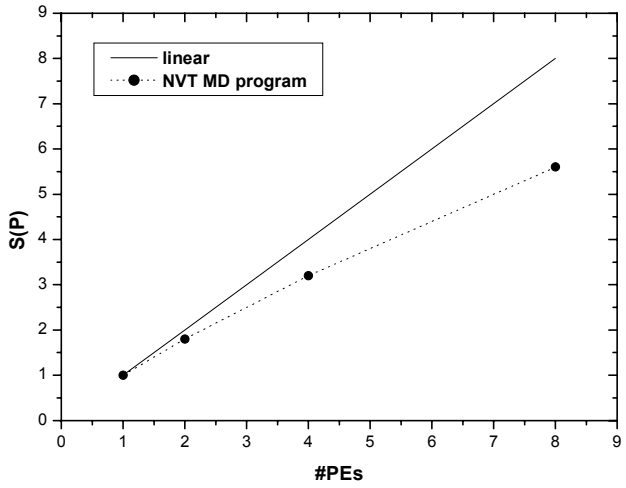

Figure 14 Speed-up graph of the parallelization of the *NVT* MD simulation runs (dotted line). Shown with the solid line is the optimal speed-up.