DATA-PARALLEL MOLECULAR DYNAMICS OF COMPLEX MOLECULES BY A FLEXIBLE, MULTIPLE TIME SCALES APPROACH *

CARLO NARDONE † PIETRO ROSSI MARIA VALENTINI ‡

CRS4 Parallel Computing Group,
Center for Advanced Studies, Research and Development in Sardinia
via N.Sauro 10, I-09123 Cagliari, Italy, email: cmn@crs4.it

Abstract

Classical Molecular Dynamics of polyatomic liquids involves a special treatment of the intramolecular bonds, which is not very well suited for massively parallel computers programmed in data-parallel style. An alternative approach, based on a 'Multiple Time Scales' algorithm, is presented, which is fully data-parallel and energy-conserving with the same order in Δt as standard Verlet integration scheme. Results are given for codes written in CMFortran and tailored for the CM-200 simulating SPC water either with the Ewald sum method (all-pairs interactions) or with reaction field treatment of the long-range potential, allowing the usage of a coarse-grained cell method for the parallelization of the force calculations.

1 Introduction

Molecular Dynamics (MD) is a well-established simulation technique to infer the statistical and dynamical properties of an ensemble of molecules, given an appropriate form of the interaction potential [1]. In the framework of classical mechanics, a polyatomic molecule can be modeled by a set of point particles (atoms and massless 'virtual sites'). Molecules interact via a site-site pairwise potential, including for example electrostatic and Van der Waals terms ('non-bonded' terms).

The intramolecular potential is chosen so as to constrain covalent bond lengths and angles to oscillate quickly, although for a proper treatment a quantum mechanical approach should be followed ('bonded' terms). Usually the time scale of intramolecular motion is much smaller than intermolecular motion. Apparently, this would entail using too small a time step compared to the phenomena normally of interest in this type of simulation. This fact motivated the freezing of the 'hard' degrees of freedom by considering the molecule to be rigid.

*Work carried out with the financial contribution of Sardinia Regional Authorities

†author to whom correspondence should be addressed

‡present address: LPMI, École Polytechnique, 91120 Palaiseau, France

2 Standard Bond Treatment

The two methods commonly used to simulate rigid polyatomic molecules are the rigid body equations of motion and the so-called method of constraints [2].

The first method is based on writing Euler equation for a rigid body. Aside from the need of special treatment in order to avoid spurious singularities, this method rapidly becomes very cumbersome to manage if one wants to retain some internal degrees of freedom (d.o.f.) in a complex molecule. This is generally needed at least for some of the 'softer' internal d.o.f., typically torsional angles or even bond angles.

The second method consists in adding Lagrange multipliers to the forces written in the atomic Cartesian coordinates. Taking for example a bond length constraint between atoms k and l, written as $\sigma_{kl} = (\mathbf{r}_k(t) - \mathbf{r}_j(t))^2 - d_{kl}^2 = 0$, the constraint force to add is $\mathbf{G}_{ik} = -\sum_l \lambda_{kl} \nabla_{ik} \sigma_{kl}$. This formulation allows naturally the introduction of intramolecular terms as wanted, therefore it is the standard method of choice. The constraints σ_{kl} are fulfilled *exactly* at each time step (to obtain better numerical stability), which results in a set of nonlinear equations for the Lagrange multipliers λ_{kl}. They can be solved either by Newton-Raphston iterative method, which needs a matrix-inversion for each molecule for each time step and therefore too computationally expensive for large molecules, or by a more convenient iterative procedure exploiting the sparseness of the constraints among atoms. This procedure is called SHAKE if the Verlet integration scheme is used [3], similar procedures exist for other integration schemes.

SHAKE solves each constraint in succession, looping iteratively over all constraints until convergence to a desired tolerance is reached. Therefore it is clearly a sequential algorithm on each molecule. Also, it is known that SHAKE may have problems of convergence in some cases, requiring specific recasting of the constraints [4].

3 The MTS Algorithm

An alternative approach is based on reconsidering the argument about the time step needed to model the fast intramolecular motion, using a Multiple Time Scales (MTS) algorithm. With MTS technique the evolution is split into two components: only one 'slow' force calculation is required every n 'fast' force calculations. The fast dynamics comprises the intramolecular oscillations and the slow dynamics comprises the intermolecular interactions. The advantage of adopting this approach stems from the fact that while there are $O(N^2)$ long-range binary intermolecular interactions, there are only $O(MN)$ intramolecular potential terms (where N is the number of molecules and M is the number of constraints to be fulfilled per molecule). Therefore one can fix a 'fast' time step to model correctly the intramolecular vibrations while retaining a 'slow' time step as typical in the standard approach described in the previous paragraph.

In the past there have been doubts about the quality of the energy conservation in this kind of time integration schemes. This is not a difficulty if one adopts an explicitly time reversible and area preserving scheme. Such an algorithm has been described independently in [5] and [6]. It is based on an approximation of the evolution operator $exp[t\mathcal{L}(\mathcal{H})]$, where $\mathcal{H}(p,q)$ is the Hamiltonian, q and p are the sets of coordinates and momenta and $\mathcal{L}(f)g = \sum_i [(\partial f/\partial p_i)(\partial g/\partial q_i) - (\partial f/\partial q_i)(\partial g/\partial p_i)]$ is the Liouville operator. For any observable $\Phi(t) = \Phi[p(t), q(t)]$ the evolution operator solves formally the time evolution governed by the Hamiltonian $d\Phi(t)/dt = \mathcal{L}(\mathcal{H})\Phi$, by means of the expression

$\Phi(t) = exp[t\mathcal{L}(\mathcal{H})]\Phi(0)$.

The approximation is the following:

$$exp[\Delta t\mathcal{L}(\mathcal{H})] \cong exp[\Delta t\mathcal{L}(h_n)] \times \cdots$$

$$\times exp[\Delta t\mathcal{L}(h_1)]exp[\Delta t\mathcal{L}(h_0)]exp[\Delta t\mathcal{L}(h_1)] \times \cdots \times exp[\Delta t\mathcal{L}(h_n)], \quad (1)$$

valid for a decomposition of the form $\mathcal{H} = h_0 + 2h_1 + \ldots + 2h_n$, and within an accuracy $o(\Delta t^2)$ [5].

When one chooses $h_0 = 1/2\sum_i p_i^2/m_i$ and $h_1 = 1/2V(q)$ the well-known leapfrog update in the 'velocity Verlet' [1] form is recovered. If we split the Hamiltonian in the 'fast' and 'slow' contributions, i.e. $h_1 = 1/2V_f(q)$ and $h_2 = \cdots = h_n = 1/2V_s(q)$, we get the approximation

$$exp[\Delta t\mathcal{L}(\mathcal{H})] \cong exp[\frac{\Delta t}{2}\mathcal{L}(V_s(q))]\times$$

$$\times \left[exp[\frac{\Delta t}{2n}\mathcal{L}(V_f(q))]exp[\frac{\Delta t}{n}\mathcal{L}(\frac{1}{2}\sum_i \frac{p_i^2}{m_i})]exp[\frac{\Delta t}{2n}\mathcal{L}(V_f(q))]\right]^n \times exp[\frac{\Delta t}{2}\mathcal{L}(V_s(q))] \quad (2)$$

which is the core of the proposed MTS algorithm.

This formulation of MTS has the advantage of being inherently time reversible (by the symmetry of the expansion 2) and area-preserving (in the sense of Liouville's theorem), therefore the energy drift is negligible. Furthermore the energy is conserved with an accuracy $o(\Delta t^2)$, i.e. equivalent to the standard Verlet or leap-frog integration scheme. Higher-order schemes can easily be formulated using the same formalism.

The algorithm in pseudo-code is the following:

```
initialize ( q(i), p(i) )
f_s(i) <- forces_slow( q(i) )
f_f(i) <- forces_fast( q(i) )
# main loop
do iter = 1,max_iter
   p(i) <- p(i) + dt / 2 * f_s(i)
   do count = 1,n
      p(i) <- p(i) + dt / (2*n) * f_f(i)
      q(i) <- q(i) + dt /    n  * p  (i)
      f_f(i) <- forces_fast( q(i) )
      p(i) <- p(i) + dt / (2*n) * f_f(i)
   end do
   f_s(i) <- forces_slow( q(i) )
   p(i) <- p(i) + dt / 2 * f_s(i)
end do
```

where `f_s` is equal to $-\partial V_s(q)/\partial q_i$ and `f_f` is equal to $-\partial V_f(q)/\partial q_i$. Note that one 'slow' forces calculation and n 'fast' are required per time step.

4 Parallel Implementation

4.1 Data-parallel strategies

Molecular Dynamics is a typical example of N-body problem. When the interactions are long-range (which is the case in charged systems such as water), the calculation of the forces acting on each particle requires considering all the $N(N-1)/2$ pairs of particles (when multiple interactions are negligible, which is generally true for non-bonded interactions). Leaving aside known methods to handle the $O(N^2)$ interactions indirectly, such as multipole expansion and particle-mesh, we are left with a particle-particle treatment. This class of problems displays an intrinsic data-parallelism [7] that can be exploited in different ways on distributed-memory, massively parallel processors (MPP) machines, according to the ratio between the number of particles N and the number of (virtual) processors p available:

1. $p \sim O(N^2) \Rightarrow$ one processor per binary interaction;
2. $p \sim O(N) \Rightarrow$ one processor per particle;
3. $p \ll N \Rightarrow$ one processor per spatial cell.

In the Fortran 90 environment the solution for case 1 is to use a `spread` of the status array of the N particles in order to calculate all forces concurrently, but for the number of particles of interest this approach is unfeasible. Case 2 requires ordering of the particle status in a ring which is `cshifted` $N-1$ times in order to pass the status to every other particle, accumulating the forces acting onto each particle (the last `cshift` would correspond to self-interaction, therefore it is skipped); this technique is also called 'digital orrery' in the Thinking Machine parlance [8]. Case 3 corresponds to a coarse-grained parallelism with fewer processors, exploiting a spatial decomposition of the simulation box. It is feasible only for short-range forces, otherwise the efficiency is severely limited by the communications. The scheme have been described elsewhere, a detailed explanation of its implementation on the CM-200 is in [9]. In both cases 2 and 3 it is possible to exploit Newton's third law cutting the force calculations by half, keeping track of the accumulated forces on an auxiliary array.

The remaining tasks of a MD code, i.e. the time integration scheme for updating the particle status and the global sums necessary to get macroscopic quantities are trivially parallelized by the CMFortran syntax and intrinsics.

4.2 Spatial decomposition strategies

We decided to implement both strategies 2 and 3, trying to keep the different codes as much compatible as possible. Therefore we have two basic data structures for the molecules: an array with a single parallel axis (with `:NEWS` layout) which is used everywhere apart from the force module for strategy 3, and an array with three parallel axes (again with `:NEWS` layout) spanning a coarse-grained 3D grid. Mapping to the latter structure and back to the original one is done using library calls particulary efficient on CM-200 (`CMF_SCAN_ADD`, `CMF_DEPOSIT_GRID_COORDINATE`, `CMF_SEND_OVERWRITE`, `CMRT_get`), therefore it is possible to perform this procedure for each time step. Typically, the mapping takes a mere 1% of the CPU time.

In the coarse-grained algorithm, usually only the nearest neigbor cells are considered (3^3), visited by means of a succession of `cshift` along a raster. But if the computation vs communication ratio is large (as in the case of molecules), next-nearest neigbor cells can be considered as well (5^3). In fact, in the hypothesis of perfect load-balancing among cells and neglecting communication cost, the efficiency of any such scheme is given by η = (actual number of interactions within cutoff r_{cut}) / (effective number of interactions considered for a cell length L). In the 3^3 case, $L = r_{cut}$, therefore $\eta_3 = (4/3\pi L^3)/[(3L)^3] = 0.155$ (quite low!), while in the 5^3 case $2L = r_{cut}$, then $\eta_5 = [4/3\pi(2L)^3]/[(5L)^3] = 0.268 = 1.73\eta_3$. Clearly this could be carried over for more and more neigbors, eventually hitting the load-balancing or the communications limit. It is interesting to note here that the 'particle-coloring' algorithm devised by [9] to save further interactions, have efficiencies $\eta'_3 = 0.226 = 1.46\eta_3$ and $\eta_5 = 0.398 = 2.57\eta_3$.

4.3 Bonded terms strategies

Some more considerations are necessary to apply case 3 to polyatomic molecules. If the molecule is small ($R \ll r_{cut}$) it is feasible to map the entire molecule in the coarse-grained grid as if it were a point particle, provided that $L = r_{cut} + 2R$ (R being the size of the molecule). In this case there are no interprocessor communications either for SHAKE or for the 'fast' time steps in MTS (they both do not change the molecule center of mass). Also, both SHAKE and MTS computation time scale as $O(N)$ (with a much smaller factor than the non-bonded terms), but since the number of iterations necessary to get a given accuracy is dominated by the worst case, there is the potential danger of poor load balancing if one distributes different molecules onto different processors.

The situation is more complicated when the molecule is large. A scheme must be devised in order to distribute the sites of the molecule among different processors and minimizing the interprocessor dependencies caused by the bonded terms [10, 11]. This is true for both SHAKE and MTS, but the real advantage of the latter is again its intrinsic parallelism on the bonded terms, while the former is bounded to consider one constraint after the other.

4.4 Timing results

As an example of actual CPU time scaling, we show in Fig.1 this time (measured by CM timer library calls) per time step per molecule, referring to an all-pairs code for the water model described in the next paragraph. While the serial code CPU time per molecule scales as N, the parallel code reaches an optimum dependent on the number of physical processors, then starts to scale as N, with a much smaller factor compared to the workstation. The minimum corresponds to N equal to the number of floating-point processors per pipeline length in the specific CM-200 used.

In the case of the coarse-grained code, the CPU time scales as $O(N)$, the factor is dominated by the non-bonded terms and depends on r_{cut} as well as on the maximum allowed number of molecules per cell `NW` (which must be tailored on the local fluctuations, which in turn depend on the physical parameters of any specific run). In the case of the water model, for N=16384, $r_{cut} = 4\sigma$ (see next paragraph) and `NW` = 15 we have roughly 40 s per time step on the CM-200/8k, corresponding to about 20μs per effective molecule-molecule interaction. The communication time due to the raster account for only 5% of the CPU time. The fact that the coulombic interactions account for about

75% of calculations could justify the effort to write a specialized kernel in microcode; encouraging experiments have been performed with DPEAC on the CM-5 (overall gain of a factor of two).

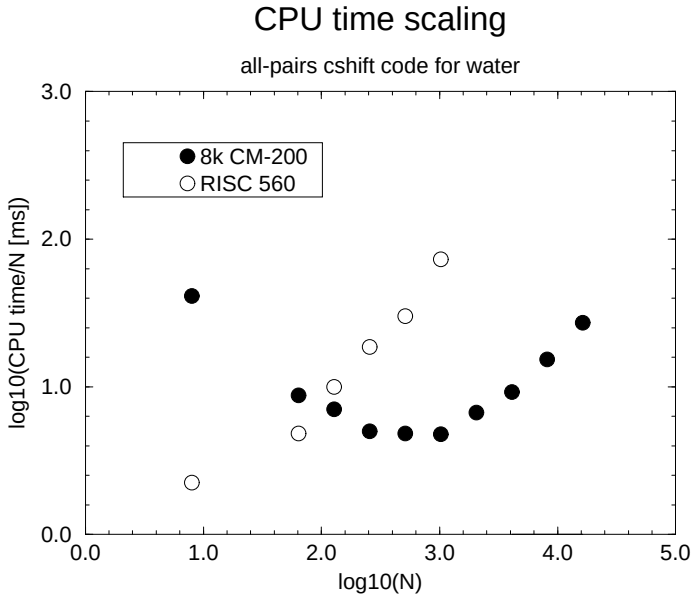

Figure 1: *Log-log plot of CPU time per step per molecule (ms) vs number of molecules N for SPC-flexible model of water (all-pairs, Ewald sum code) running on a workstation IBM RISC/6000 560 (open circles) and on a CM-200/8k (full circles).*

5 Application to water

We choose to consider water as a test of the MTS algorithm because it is both a simple, non-trivial polyatomic molecule and because of its outstanding scientific interest. Also, flexible water model results using different MTS schemes have been published [12]. The SPC water model [13] is a 3-sites model where the sites coincide with the O and H nuclei and are assigned respectively -0.82e and 0.41e electric charge. Furthermore, the Oxygen atoms interact each other through a Lennard-Jones potential $V(r) = 4\epsilon[(\sigma/r)^{12} - (\sigma/r)^6]$. with interaction length $\sigma = 3.166$ Å and potential well $\epsilon = 0.6502$ kJ/mole. We tested as intramolecular potential an harmonic one, with the parameters used in reference [12]. The intramolecular potential is written

$$V(q) = \frac{1}{2}k_r \sum_i \sum_{j=1}^{2} (r_i^{(j)} - r^{(0)})^2 + \frac{1}{2}k_\theta \sum_i (\theta_i - \theta^{(0)})^2 \qquad (3)$$

where $\mathbf{r^{(j)}} = |\overline{OH_{(j)}}|$ and $\theta = \widehat{HOH}$, with $r^{(0)} = 1.0$ Å and $\theta^{(0)} = 109.47^o$ are the SPC equilibrium values. The constants k_r and k_θ have been chosen equal respectively to $4.637 \cdot 10^5\,\mathrm{kJ/mole\,nm^2}$ and $383\,\mathrm{kJ/mole\,rad^2}$.

The coulombic, long-range terms imply that an all-pair algorithm is to be used (augmented by the so-called Ewald sum to account for the infinite replica of the simulation box). Nevertheless, for many purposes it is possible to introduce a cutoff, and therefore a coarse-grained algorithm, provided that a correction for the average polarization of the cutoff sphere is included (reaction field method).

Clearly the accuracy of the energy conservation is crucial for the present MTS algorithm. We have calculated a relative drift $(1/E_{tot})(dE_{tot}/dt)$ of $(2.3 \pm 0.1) \cdot 10^{-6}\mathrm{ps}^{-1}$ (which means 0.2 K only over 100 ps) over a period of 70000 time steps, for a simulation of 512 molecules including 178 Ewald terms, at 297 ± 5 K.

This is to be compared with previous MTS schemes which can have drifts many hundred times higher [12]. The time steps used were $\Delta t = 1$ fs for the 'slow' and $\Delta t_f = 0.1$ fs for the 'fast' forces, which gives a fluctuation of the total energy of approximately 1% of the fluctuation in the total potential energy, as commonly required. Fig.2 shows the total energy fluctuation observed in our system of 512 molecules at 300 K as a function of Δt in log-log plot. The data is fitted reasonably well by the predicted $o(\Delta t^2)$ accuracy.

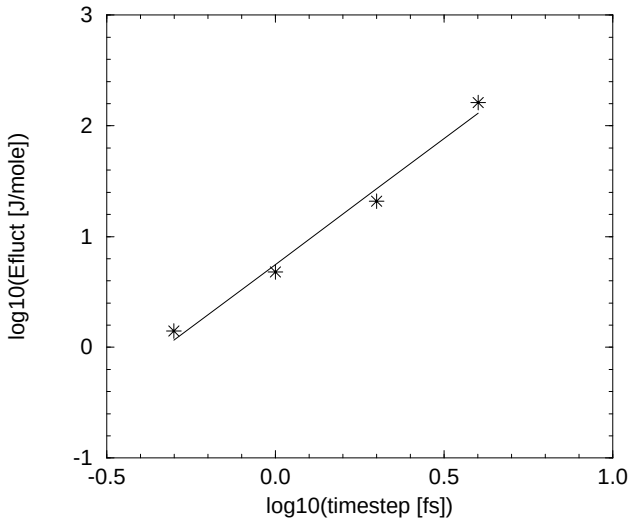

Figure 2: *MTS energy conservation. Log-log plot of the total energy fluctuations in J/mole vs 'slow' time step Δt in fs (stars). The line shows the best fit which has slope* 2.3 ± 0.2

An apparent difficulty inherent in flexible models of polyatomic molecules is that 'hard' d.o.f. are difficult to thermalise [2]. In particular, the vibrational kinetic energy is transfered relatively quickly to the translational d.o.f. (few tenths of ps for a periodic box of 512 water molecules), whereas the exchange in the opposite direction is much slower (several tens of ps). We have found that the Maxwellian shuffle technique thermalise each degree of freedom at room temperature within 2 ps only. This technique consists in updating the velocities of all atoms with a Maxwell-Boltzmann distribution at the desired temperature every few time steps [14] (in our case, every 100 time steps of 1 fs). This corresponds to put each particle instantaneously in contact with a heat bath, forcing ergodicity of the N-body system.

6 Conclusions

In this paper, we have shown that one can exploit the presence of separate time-scales to simulate flexible polyatomic molecules in a straightforward and efficient way, by using a Multiple Time Scale algorithm. We have demonstrated good energy conservation for a real, important system, i.e. water, spending the same CPU time as for conventional rigid

models. MTS allows a reduction of the computation time over STS by exploiting the different numerical complexity of the intra- and inter- molecular interactions.

In conclusion, we have shown that a proper MTS approach is an effective method in the Molecular Dynamics simulation of polyatomic liquids, allowing a natural move to parallel, efficient code for running on MPP machines.

Acknowledgments

We are pleased to acknowledge the work of Paolo Marenzoni for help in porting the Flexible Water code to CMFortran, and for the DPEAC kernel development, alongside with Ralph Santos. We also thank Ernesto Bonomi and Francesco Sciortino for valuable discussions and suggestions.

References

[1] M.P. Allen, and D.J. Tildesley, *Computer Simulation of Liquids* (Clarendon Press, 1987).

[2] G. Ciccotti and J.-P. Ryckaert, *Computer Phys. Rep.*, **4**, 345 (1986).

[3] W.F. van Gunsteren and H.J.C. Berendsen, *Molec. Phys.*, **34**, 1311 (1977).

[4] K.D. Hammonds and J.-P. Ryckaert, *Computer Phys. Comm.*, **62**, 336 (1991).

[5] J.C. Sexton and D.H. Weingarten, Technical Report RC 17668, IBM Research Division, (1992).

[6] M.E. Tuckerman, B.J. Berne and G.J. Martyna, *J. Chemical Phys.*, **97**, 1990 (1992).

[7] E. Bonomi and M. Tomassini, *Int. J. Modern Phys. C*, **3**, 709 (1992).

[8] B.G.J.P.T. Murray, P.A. Bash, and M. Karplus, Technical Report TMC-31 CB88-3, Thinking Machines Corporation, (1988).

[9] F. Hedman and A. Laaksonen, *Int. J. Modern Phys. C*, **4**, 41 (1993) and *Int. J. Quantum Chem.*, **46**, 27 (1993).

[10] S.L. Lin, J. Mellor-Crummey, B.M. Pettitt and G.N. Phillips Jr. *J. Comp. Chem.*, **13**, 1022 (1992).

[11] S.E. DeBolt and P.A.Kollman, *J. Comp. Chem.*, **14**, 312 (1993).

[12] A. Wallqvist and O. Teleman, *Molec. Phys.*, **74**, 515 (1991).

[13] H.J.C. Berendsen, J.P.M. Postma, W.F. van Gunsteren, and J.Hermans, in B. Pullman (ed.), *Intermolecular Forces*, Reidel, 331 (1981).

[14] E. Bonomi, *J. Stat. Phys.*, **39**, 167 (1985).