ELSEVIER

Journal of Molecular Graphics and Modelling 21 (2002) 89–99

Journal of Molecular Graphics and Modelling

www.elsevier.com/locate/JMGM

A large scale molecular dynamics simulation code using the fast multipole algorithm (FMD): performance and application

James A. Lupo [a,*,1], Zhiqiang Wang [a], Alan M. McKenney [a], Ruth Pachter [a], William Mattson [b]

[a] *Materials Directorate, Air Force Research Laboratory, Wright-Patterson, AFB, OH 45433-7702, USA*
[b] *Department of Physics, University of Illinois, Urbana-Champaign, 1110 W. Green Street, Urbana, IL 61801, USA*

Received 10 February 2002; accepted 30 March 2002

Abstract

We present the performance of the fast classical molecular dynamics (MD) code, fast molecular dynamics (FMD), designed for efficient, object-oriented, and scalable large scale simulations, and summarize its application to a liquid crystalline cluster. FMD uses an implementation of the three-dimensional fast multipole method, developed in our group. The fast multipole method offers an efficient way (order O(N)) to handle long range electrostatic interactions, thus, enabling more realistic simulations of large molecular systems. Performance testing was carried out on IBM SP2, SGI Origin 2000, and CRAY T3E massively parallel systems using the MPI massage passing library. The electrostatic forces were tested on models of up to 100,000 randomly placed charges, and on protein and liquid crystalline molecular systems of over 99,000 atoms. Tests on the stability of the method are presented, along with comparisons with direct calculations, the NAMD2 code, and the physical multipole-based cell-multipole method.
© 2002 Published by Elsevier Science Inc.

Keywords: Fast multipole algorithm; Liquid crystalline cluster; Molecular dynamics

1. Introduction

Atomic level simulations play an important role in providing structure-to-property relationships of materials. The properties of interest often require very large scale molecular dynamics (MD) simulations, both in terms of large molecular structures and long duration simulations. The demands of such large models, coupled with the implementation of new algorithms and methods, requires a code which is efficient, modular, portable, and readily extensible. In addition, the efficient use of massively parallel processors demands good scalability. The general techniques used in MD have advanced over the years [28]. With the advent of parallel processing, emphasis has been placed on developing new new algorithms to deal efficiently with such problems as the spatial discretization of the problem domains [34,32], and the rapid computation of the short range molecular forces [25]. For very large systems, the efficient computation of the long range electrostatic non-bonded forces can make the difference between a tractable and an impossible model. Since the electrostatic force is a pairwise interaction, a direct calculation method requires O(N^2) operations per time step. In many cases, the simple cutoff method is highly inappropriate [26,27]. The Ewald summation method improves the scaling to O($N^{(3/2)}$), but only for those models which use periodic boundary conditions. While a variety of new techniques have been developed, such as the local reaction field [20], particle–particle–particle-mesh (P3M) [17], and particle-mesh Ewald (PME) [8], significant work has gone into methods based on electrostatic multipoles in general [15,16,6] and specifically in parallel [14,18,9,3,33]. The fast multipole method[15] uses a multiscale hierarchy of partitions of the volume and a divide-and-conquer strategy to compute the power series, allowing the electrostatic forces to be computed to specified accuracy in O(N) operations. Still, the cost of computing all electrostatic interactions remains high, and a multiple time step method is often used to reduce the overhead.

A number of MD programs have appeared which use fast multipole methods [11,12,35]. We chose the NAMD version 1.4 [24] as the software base for our fast molecular dynamics (FMD) code. The code was modified to be fully supported on a variety of high end computing platforms running MPI, extended with several new features, and a three-dimensional fast multipole method was implemented for rapid calculation of electrostatic forces.

* Corresponding author.
[1] 9667 Independence Dr., Westminster CO 80021-6845, USA.

We have implemented the fast multipole method in a library named fast multipole method in three-dimensions (FMM3D) [22,23]. The library includes a variety of schemes for computing multipole translations, which were introduced to control both computational time and accuracy, and it has been incorporated into the FMD program. In this paper, we first briefly describe the FMD code and the fast multipole method used in FMM3D. We then present the results of a variety of performance testing, using simple models of up to 100,000 random point charges, and more realistic models such as liquid crystal clusters of 4-n-pentyl-4′-cyanobiophenyl (5CB) with up to 99,522 atoms. The random charge models test the performance of the FMM3D library, while the molecular models test the overall performance of FMD. The performance is characterized by the parallel scalability and the order-N nature of the FMM algorithm.

2. Program design and algorithms

2.1. Fast molecular dynamics (FMD)

FMD is derived, in part, from NAMD version 1.4 [24], developed by the Theoretical Biophysics Group at the University of Illinois and the Beckman Institute. It uses an X-PLOR [5] compatible force-field, based on the CHARMM force field [4], to compute the forces between intra- and inter-molecular atoms. The interaction potential takes the form $\Phi = \Phi_{\text{bonded}} + \Phi_{\text{non-bonded}}$, where

$$\Phi_{\text{bonded}} = \Phi_{\text{bond}} + \Phi_{\text{angle}} + \Phi_{\text{dihedral}} + \Phi_{\text{improper}} \quad (1)$$

represents the physically connected near-neighbor interactions, and

$$\Phi_{\text{non-bonded}} = \Phi_{\text{electrostatic}} + \Phi_{\text{van der Waals}} \quad (2)$$

represents the long-range pair-wise interactions.

NAMD was written in a C++ object-oriented style to ease support and maintenance, and facilitate implementation and testing of novel computational methods. This was ideal for our purposes, and the core of FMD retains the C++ object-oriented nature of NAMD . The following lists the new features in FMD.

1. Replacement of distributed parallel multipole tree algorithm (DPMTA) [3] with FMM3D. The FMM3D support is written in C and FORTRAN-77. It utilizes the DFFTpack library from NETLIB [29], which is written in FORTRAN-77. Unlike the MIMD-style incorporation of DPMTA in NAMD , FMM3D was implemented SPMD-style in FMD. This, coupled with the MPI modification below, made execution of FMD in a vendor supported job batch environment, such as NQS or PBS, more convenient.
2. Porting of the parallel communication system from PVM to MPI, since MPI is fully supported on DoD high performance computing platforms. Several versions of the MPI library are also available for use on workstation clusters. FMD will successfully operate with vendor provided versions of the MPI library, as well as the two most popular portable implementations: LAM-MPI and MPI-CH.
3. Addition of support for NCSA hierarchical data format (HDF) [13] formatted input/output files. Since the HDF file format is architecture independent, the I/O files can be generated and read on any machine which supports the HDF library. This allows free movement of simulations at any stage between machines, as well as facilitation of data exchange.
4. Adoption of network distributed global memory (NDGM) and the distributed interactive computing environment (DICE) packages [7] developed by the Army Research Laboratory as the visualization and control interface.
5. The source files were restructured to use the GNU AUTOCONF [21] tools, which facilitates configuration and compilation of FMD on a wide variety of computer architectures, ranging from stand-alone Intel CPU-based Linux workstations, to heterogeneous workstation clusters, to massively parallel processors such as the Cray T3E, IBM SP2, and SGI Origin 2000.
6. Addition of support for periodic boundary conditions with arbitrary triclinic unit cells and periodicity in any combination of dimensions. We note, however, that FMM3D has not yet been modified to support periodic boundaries, though that work is in progress.
7. Introduced optional, non-standard, formats for the PDB and PSF files to allow handling structures of nearly 16 million residues, and 2.1 billion atoms.

2.2. Parallelization

The parallelization of MD steps is based on a spatial decomposition scheme using cubical volumes, called *patches* [24]. Long-range electrostatic forces are computed by collecting the atom positions and charges on each processor and passing them to a separate force calculation package, FMM3D. The van der Waals forces are treated with one of several cut-off techniques. FMM3D uses an oct-tree spatial discretization scheme. This separate scheme allows for run time optimizations of FMM3D independent of the rest of the code, though at a cost of increased overhead and memory requirements.

There is a large overhead involved in calculating all of the multipole terms in the long range interactions. This cost is decreased by a multiple time step technique which assumes that the multipole terms vary much slower than the local terms, achieved by specifying a certain number of time steps for a fast multipole cycle. At the start of each cycle, the full long range interaction is calculated, including multipole and local contributions. For the remainder of the cycle, the multipole terms are reused, and only the local interactions are recalculated. A cycle size of 10 has been found to give an acceptable trade off between accuracy and time savings, though stability concerns limit the cycle size to 4 or less for nanosecond scale simulations.

2.3. Fast multipole method in 3D (FMM3D)

The FMM3D fast multipole method, as described by Greengard and Rokhlin [15,16], contains two principal elements:

- Approximation of the field due to sufficiently distant, i.e. *well-separated*, charges by spherical harmonic series;
- Calculation of all contributions to the series via a multiscale technique using a hierarchical cubic oct-tree structure of computational cells.

The FMM3D scheme for combining the field approximations in a hierarchical fashion has been described by McKenney and co-workers [22,23]. The FMM3D method breaks down into four main phases to estimate its cost (execution time) scalings:

1. Computation of finest-level multipole expansions. Cost: $O(Np^2)$.
2. Computation of finest-level local expansions. (This includes the computation of all coarser-level expansions.) Cost: $O(p^4 8^L)$.
3. Evaluation of finest-level local expansions on each atom. Cost: $O(Np^2)$.
4. Direct calculation of near interactions. Cost: $O(N^2/8^{-L})$.

Thus, if the finest level width w is fixed, the cost of the direct calculation is $8^L \propto N$. The total cost is $O(Np^4)$. Note that these scalings only apply for sufficiently large N, p, and L. On the other hand, if L is fixed, then the direct calculation will still cost $O(N^2)$.

The accuracy of the method is determined by ϵ^p, where ϵ is the ratio of the size of the level-L cell to the size of its neighborhood, and p is the order of the truncation terms in the multipole expansion. Note that this is not explicitly dependent on the level L or the cell width w. The level L is adjusted as the system size increases only to optimize the cost to order-N.

2.4. Periodic boundary conditions

Periodic boundary conditions were implemented by modifying how neighboring patches were determined. In NAMD 1.4, patches are always cubic, and sized slightly greater than the cut-off distance so at most only 26 nearest neighbor patches are considered. In a triclinic system, the number of neighbors in any direction may be different, since the unit cell width varies in each dimension.

Finding neighbors was reimplemented as a two-step process. First, a boundary map is created, which identifies all of the patches. For simplicity a two-dimensional map is shown in Fig. 1, showing a system which is periodic in the X-dimension only.

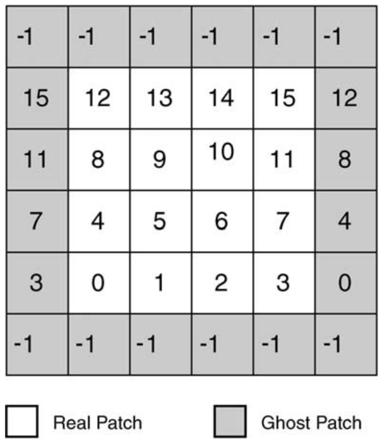

Fig. 1. Example of a two-dimensional boundary map for a system periodic in the X-dimension only.

Each patch is marked with its identification number, and a margin of *ghost* patches are added around the boundary. The number of margin cells is controlled by the width of the patch in that direction. At a boundary, these *ghost* patches are either marked with a negative one, to indicate that no patches exist beyond that edge, or with the corresponding image patch, if the system is periodic at that boundary. This example shows cubic cells, but they may be rhombic, in which case more than one margin cell may be added.

The second step involves determining the actual neighbors of each patch. For this a neighbor mask is created, as shown in Fig. 2.

The mask covers all possible neighbors of the center cell. This mask is laid over each patch in turn, and the patch identifiers which appear in the neighbor patch cells of the mask are recorded. The offset is usually chosen to match the number of margin cells added to the boundary map. It is done in a fashion which assures that all cells within the cutoff specified for the quasi-long-range forces are included.

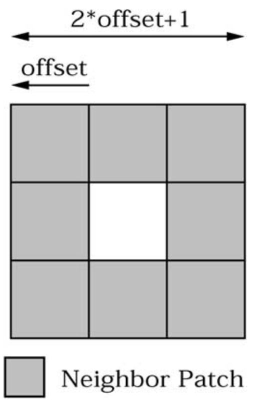

Fig. 2. Example of a neighbor mask in two-dimensions for a system with an offset (or margin) of 1 in each dimension.

Table 1
List of molecular models

Name	Molecules	Atoms
BPTI	1	892
BR	1	3502
5CB × 1	118	4484
5CB × 8	944	35872
5CB × 18	2214	80712
5CBinferior2619	2619	99522

2.5. Performance

2.5.1. FMD benchmarks

Our benchmarking has focused on the FMD performance on massively parallel platforms. The benchmark suite consists of five molecular models ranging in size from 3508 to 99,522 atoms, with 1 to 2619 molecules. These are described in Table 1. The first model is based on the photo-reactive protein bacteriorhodopsin (bR), with an attached retinal chromophore. The other models are variously size clusters composed of molecules of the liquid crystal 4-*n*-pentyl-4′-cyanobiophenyl (5CB). All models were run with a time step of 1 fs. Velocities were initialized to a random distribution corresponding to 300 K, and the FMM3D cycle was set to 10 time steps.

Performance for the larger models was found to scale with the number of processors, P, as $P^{0.8}$ on up to 64 nodes for the largest models. Fig. 3 shows the typical behavior, in this case on a SGI Origin 2000 (O2K) running the portable batch system (PBS) job queuing system under IRIX 6.5.

2.5.2. FMM3D

In order to isolate the performance of FMM3D, a model of randomly placed charges was used. The particle charges are randomly assigned in the range of −0.5 to 0.5 electronic charge. They are positioned within a cube chosen to keep the density approximately 1 Å^{-3}. FMM3D is called by FMD, however, there are no calculations of bonded energy terms. The van der Waals energy terms are still calculated, but with parameters set to zero and a cutoff distance so adjusted that a total of about 1000 patches are used to subdivide the cube. All calculations were performed on an O2K and an IBM SP2 system running the PBS job queuing system. In all cases, the normal batch environment was used. While nodes were dedicated to the jobs, some system resources were shared with other concurrent users, thus, the results should be taken as *typical*, and not as optimal.

The advantage of the FMM is its order-N scaling with the total number of the particles. Fig. 4 shows the CPU time spent per MD step for various system sizes with numbers of particles ranging from 800 to 60,000 (the power of 2.5, 2.75, ..., 4.75 of 10) using FMM3D and full direct calculation of Coulomb interactions. The FMM cell levels were set in the range from 2 to 4 according to the system size. The multipole truncation term, p, was set to 5. Due to the overhead of the fast multipole method, FMM3D requires a longer run time than the direct method for small systems. The crossover point at which FMM3D becomes advantageous is shown to be near 1800 particles. Reference lines are added in the figure for N- and N^2-scaling, which pass the crossover point. The near ideal N-scaling trend of FMM3D is shown for system sizes above 10,000 particles.

In comparison with the direct full calculations, the accuracy of FMM in these calculations is shown to have a standard deviation of 0.01% and maximum deviation of 0.02%, as shown in Table 2. Note that if we were to use the 20 Å, cutoff method, the error is so large for a cube of size 26 or larger that it gives a wrong answer.

To test scaling with the number of processors, a model containing 99,999 randomly charged particles was used. The

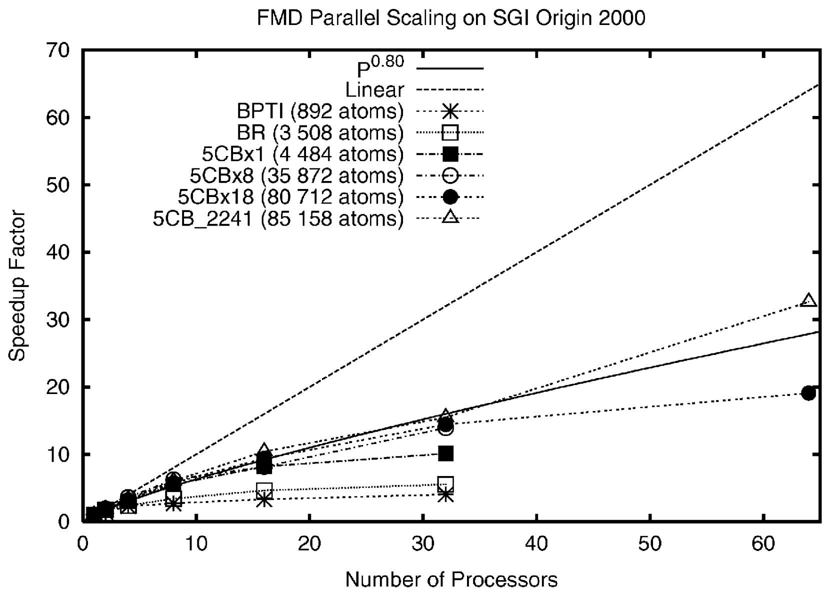

Fig. 3. FMD Parallel Performance on a SGI Origin 2000 running PBS under IRIX 6.5 performance.

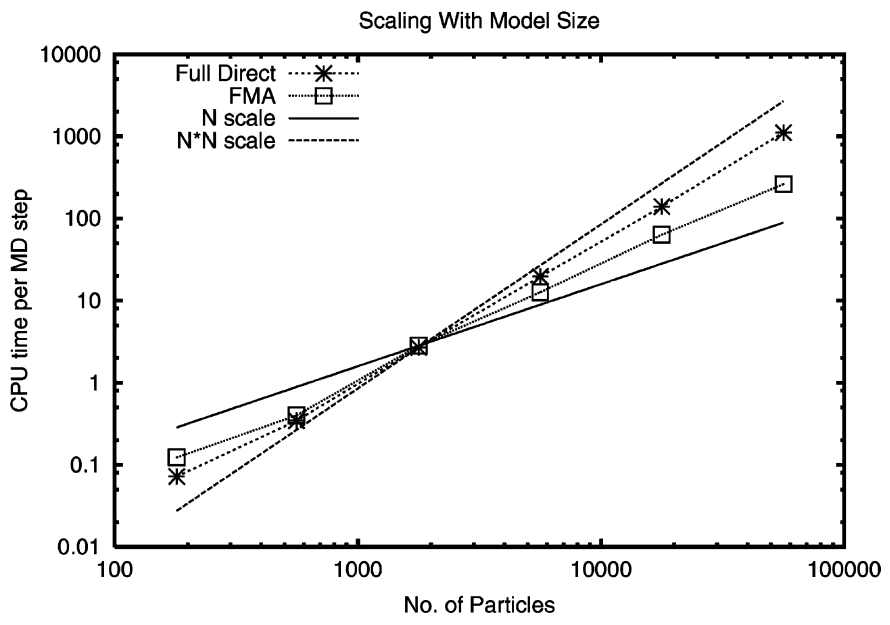

Fig. 4. CPU time vs. the number of particles.

Table 2
Electrostatic energy (kcal/mol) comparisons by model size

$\mathrm{Log}_{10}N$	Size (Å)	Full dir	FMA	Cutoff (20 Å)
2.25	5.6	−1778.8	−1778.8	
2.75	8.2	−2942.1	−2942.1	
3.25	12	−4670.5	−4670.3	−4670.5
3.75	18	−7551.8	−7552.1	−7672.3
4.25	26	−17745.8	−17750.3	38978.6
4.75	38	50829.1	50829.5	−293162.1

number of processors were chosen in powers of 2, from 1 up to 128. The finest FMM3D cell level was 4 and the multipole truncation term p was set to 5.

As shown in Figs. 5 and 6, significant speedup over the full direct method is obvious. The scalability performance is very good on up to 32 nodes, but deteriorates on 64 nodes and slowdown is obvious on 128 nodes. Both CPU and wall-clock times are shown in the figure for FMA and full direct calculations. For FMA, the difference between the real

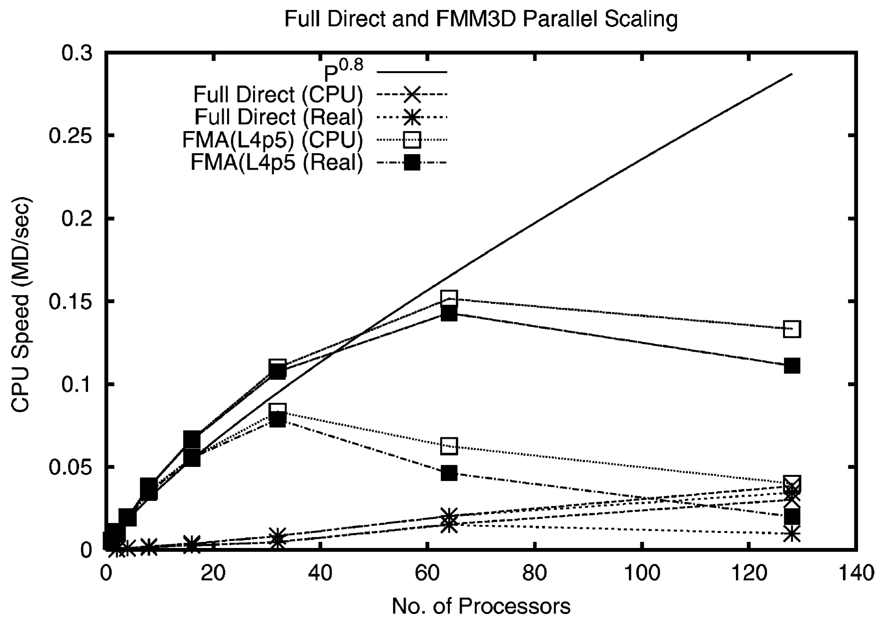

Fig. 5. CPU and wallclock(real) time speed vs. the number of processors on a SGI Origin 2000 for FMD and a 100,000 atom model.

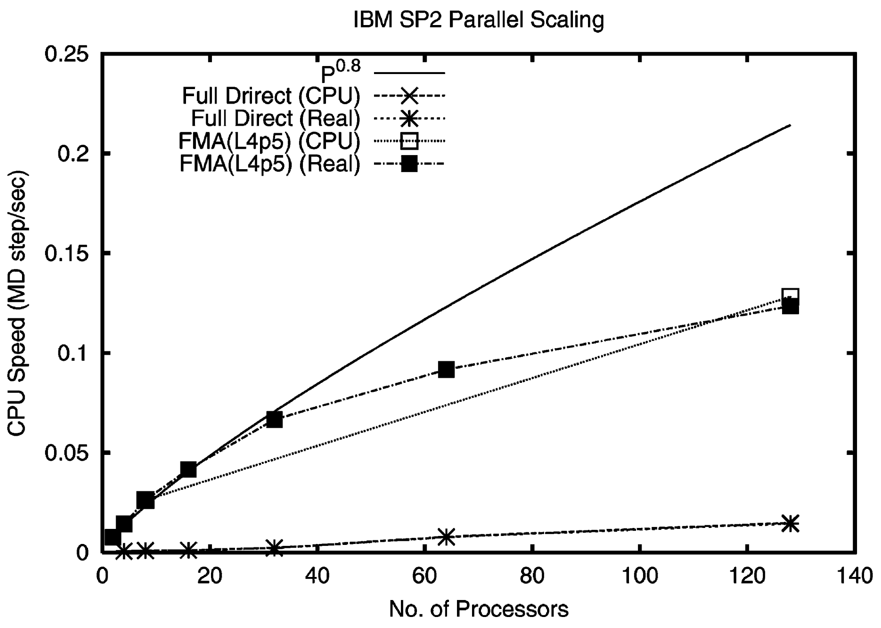

Fig. 6. CPU and wallclock(real) time speed vs. the number of processes on an IBM SP2 for FMD and a 100,000 atom model.

and CPU time becomes significant at 64 nodes, while its not apparent till 128 nodes for the direct method. Speedup on the SP2 shows better scalability than that on the O2K for large numbers of nodes.

Additional details of the scaling performance of FMM3D, which include all phases of the method, are shown in Fig. 7. These phases are:

- *Multipole*: compute and translate multipole expansions;
- *Transfer*: transfer results of multipole-to-local translations;
- *Local*: compute local expansion;
- *Eval*: evaluate local expansion for each atoms;
- *Direct*: direct calculation of near interactions;
- *Results*: broadcast results among all processors;
- *Total*: total time in FMM3D.

It is noted that the *Transfer* and *Results* steps are the main hindrance to the scalability of the FMM3D library. These are the phases which require the most communications. Therefore, the slowdown shown in Figs. 5 and 6 is likely due to a communication bottleneck.

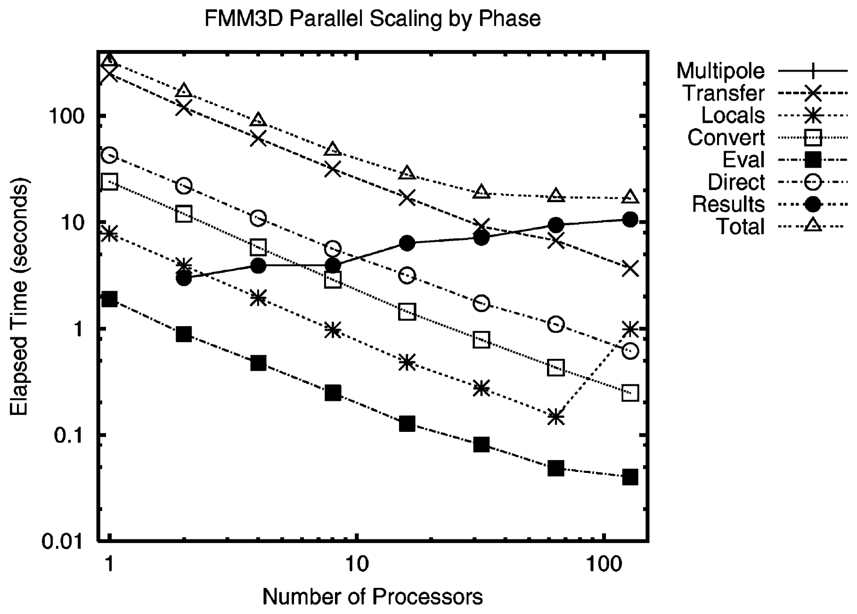

Fig. 7. IBM SP2 elapsed times of different phases in FMM3D vs. the number of processors. The model consists of 100 000 particles. FMM3D was set to retain 6 expansion terms ($p = 6$) and a 5 level oct-tree ($l = 5$) was used.

3. Application

3.1. Comparisons and stability

A comparison to the MPI version of NAMD 2.0, the newer NAMD release [19] was also performed. NAMD 2.0 has a number of additional features, such as the capability of using periodic boundary conditions. One remarkable change in program design is the introduction of an interoperable run time system, Converse, which allows the coexistence of different paradigms, such as PVM and CHARM++. It can also have MPI built on top of Converse. As a result, we were able to run the MPI version of NAMD2 on an O2K as well.

A 100,000 random charge model was used on the O2K. Two fast multipole method modules (FMM3D for FMD and DPMTA for NAMD2) and full direct calculations were used. The FMM3D run used a cell level of 4and retained five terms in its multipole expansions, while DPMTA used a cell level 5 and retained eight terms in its multipole expansion. Since DPMTA and FMM3D use different spatial discretizations and different forms for the multipole expansions, these values differ because they where chosen so both codes would give comparable accuracies. Table 3 shows the resulting electrostatic energy values, which indicates similar accuracy for both packages.

For FMD runs using FMM3D, good scalability of up to 64 nodes was seen, with some slowdown occurring on 128 nodes. For NAMD2 runs using DPMTA, similar speed and good scalable performances are obtained from 2 to 16 nodes, but NAMD2 with DPMTA is slower than FMD with FMM3D on >16 nodes. For 64 and 128 node jobs, both FMD/FMM3D and NAMD2/DPMTA show degraded speedup. The possible causes for this behavior include: differences between the FMM3D and DPMTA FMM implementations; differences related to system decomposition strategies; and possibly the overhead of MPI on top of Converse in NAMD2. Since NAMD2 only outputs the wallclock time and the runs were made on non-dedicated nodes, efforts were taken to check on repeatability and reduce the effects of system load.

A series of tests compared FMM3D with the cell-multipole method (CMM) by Ding et al. [9,10] and the direct method. These were intended to check for any charge distribution shape sensitivity in the multipole methods. The usual algorithm specifies a spatial discretization fine enough to leave but a few atoms in each cell. In reality, the shape of the molecular structure, combined with trade offs between memory consumption, CPU utilization, and communication overhead often lead to discretizations leaving cells with 30–40 atoms each.

Table 3
Electrostatic energy (kcal/mol) comparisons for 100,000 particle model

	FMD (full dir)	NAMD2 (full dir)	FMM3D	DPMTA
Energy	23230.9	23230.9	23231.1	23231.8

The CMM method is based on physical multipoles, that is, explicit expressions for monopoles, dipoles, quadrapoles, octapoles, hexadecapoles and higher terms, rather than on multipoles expressed as expansions in spherical harmonics. This approach has the benefits of being more intuitive and simpler to implement. However, the higher order multipoles are very difficult to derive beyond the octapole level, making it impossible to approach the accuracy readily set in FMM3D.

The tests consisted of randomly placing 1–250 point charges, with randomly assigned charges ranging from −1 to 1, inside a cubic and a rectangular cell. Test points were uniformly distributed over the surface of spheres with radii of 3, 4 and 5 times the bounding radius of the cell. These correspond to fast multipole method *well-separated* criteria of 2, 3 and 4, respectively. At each test point, the electrostatic potential was calculated using FMM3D, CMM and the direct method. FMM3D was tested with p =5, 10, 15 and 20, with p =5 corresponding roughly to the hexadecapole level in CMM. CMM was tested including terms up through the quadrapole and octapole levels, using the equations presented by Ding et al. in [9]. For each charge distribution, the results at each test point were used to compute the average and rms deviations from the full direct result.

The results were consistent across the board. The spread in rms values tended to increase as more particles were added. No unusual dependence on the distribution of charges was observed. The results for the rectangular distributions tended to be more accurate than the cubic distributions. This was clearly due to the shape constraint reducing the number of higher level multipoles required to approximate the potential. The FMM3D results showed behavior similar to CMM, but tended to be more accurate by factors of 30–50. This was largely due to the higher multipoles approximated by the $p = 5$ truncation term. With $p = 20$, the FMM3D rms values were down in the range of numerical noise.

Specifically, the FMM3D potentials showed rms deviations from the direct potentials that were always $<6 \times 10^{-5}$ for 250 particles at the lowest accuracy setting of $p = 5$ and WS = 2. For fewer than 50 particles, the deviations were at or $<2 \times 10^{-5}$. The deviations were well $<1 \times 10^{-5}$ for 10 or fewer particles per cell. It is clear that FMM3D potentials can be made to approach the direct potential to within the limits of numerical noise as long as the basic assumptions of the FMM algorithm are met: cell sizes small enough to guarantee only a few particles per cell, p suitably large, and the WS criteria well maintained. Given these conditions, it should not be surprising that the FMM can be as accurate as the direct method.

The multipole terms are assumed to change slowly enough that they can be reused for several time steps before they are recomputed. However, the *steps-per-cycle* setting has been observed to have an impact on energy conservation. A total of 35,000 atom models run for several nanoseconds with a setting of 10 steps-per-cycle exhibit a downward drift in the total energy of the system at a rate of approximately 2.4% per nanoseconds. Changing the number of steps-per-cycle to 4 reduced the energy drift to -3.6×10^{-1}% per nanoseconds, while a setting of 2 reduced it to -2×10^{-4}% per nanoseconds. For a properly equilibrated system, the exact values of the energy will change slightly during the course of a run, and the changes average out to zero when energy is conserved.

To gauge the magnitude of these drifts, a simple model composed of four alanine residues was created and run for 150 ps with both FMD in full direct mode, and Harvard CHARMM, release 23f5. The total energy drift was measured over the final 75 ps of each run. FMD yielded a drift of −0.091% per nanoseconds, while CHARMM yielded 0.119% per nanoseconds. Thus, FMD with FMM3D active and set to two steps-per-cycle conserves energy as least as well as the full direct method. Even at four steps-per-cycle, the energy conservation is reasonably good. Adjusting the number of steps-per-cycle in the range of 4–10 during the initialization and heating phase of a MD run seems reasonable, while a setting in the 2–4 range should be used during the equilibration and simulation phases. The exact values to be used are very model dependent.

While reducing the number of steps-per-cycle increases the total computational load, it does serve to somewhat improve the parallel scaling of the program. Benchmarks were run on an O2K using the 5CB × 8 and 5CB × 18 models. Both models were run with steps-per-cycle set at 2, 3 and 10 on up to 64 nodes, as shown in Fig. 8. The scaling on 1–16 nodes is slightly better in all cases for 2 and 3 steps-per-cycle, but becomes worse at 32 nodes. This is consistent with the scaling performance of FMM3D, which tends to peak on 16 nodes. The smaller cycle size allows the scaling behavior of FMM3D to become more predominate.

The setting of p was doubled to increase the accuracy of the multipole method and assess the increase of the number of steps-per-cycle. The energy stability did not improve, and fewer steps-per-cycle was always better. The FMM3D overhead also increased the overall runtime at higher p, offsetting any savings in the larger number of steps-per-cycle. Thus, neither a speed nor stability benefit is gained by the combination of higher accuracy and large steps-per-cycle. The settings of steps-per-cycle and p should be chosen-based strictly on the required accuracy and energy stability concerns.

The same sort of energy conservation problems were observed with DPMTA for smaller problems run over picosecond time scales [2]. It appears one can not assume that small local atom motions contribute insignificantly to the far field, given the observed energy conservation behavior. We should note that this behavior brings into question one of the claims made of the CMM method [9]. The authors indicate they can use up to 100 steps-per-cycle, arguing that the accuracy is still acceptable with such a long time between updates of the multipole terms. Even with the much higher accuracy available with FMM3D, such a long cycle time does not seem feasible in light of its impact on energy conservation.

3.2. Liquid crystalline cluster

The electro-optics of a polymer dispersed liquid crystal (PDLC) can be varied by changing its chemical constituents. The understanding of the surface anchoring transition from homeotropic (normal) to homogeneous (in plane) of liquid crystals on different polymer walls in the nematic temperature region may enable the design of such materials [1]. The simulation of a 944 5CB molecule cluster is so far the largest

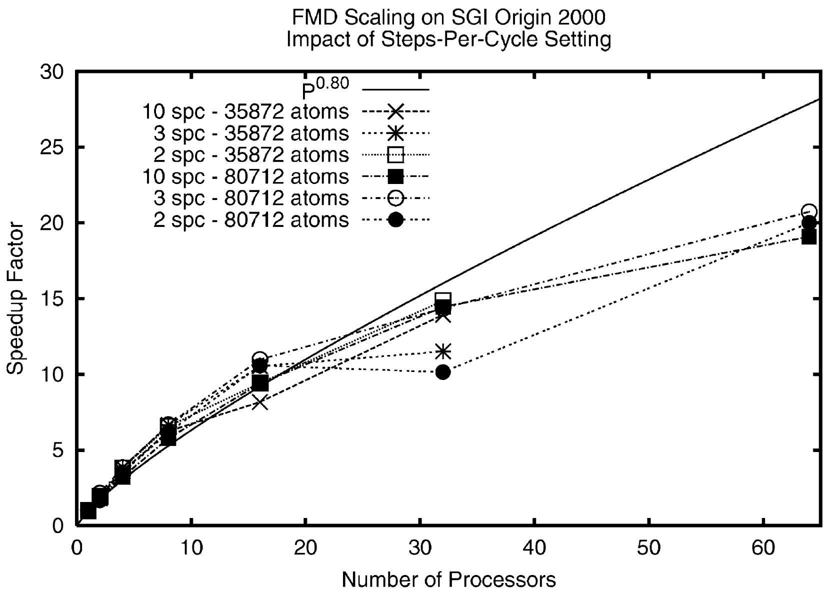

Fig. 8. Comparison of the FMD parallel scaling performance on a SGI Origin 2000 with two different models run with three different steps-per-cycle settings.

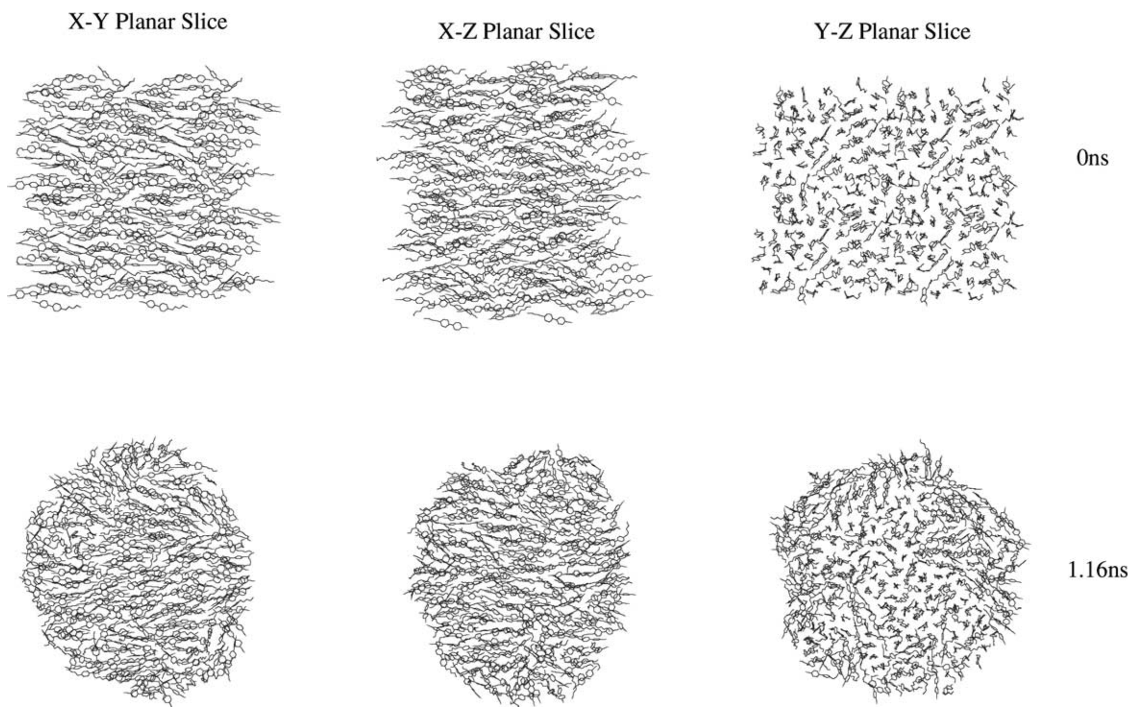

Fig. 9. Snapshots of the 944 molecule 5CB model at 0.0 and 1.16 ns. Each view is a 20 Å thick slab through the center of the model.

in size for full atomic scale molecular dynamics simulation of 5CB clusters. The molecules are initially oriented with a nematic order parameter of ca. 0.6. Velocity is rescaled to stabilize the temperature to 300 K. The MD time step is taken as 1 fs. The long-range interactions are updated in a cycle of 10 time steps. The number of multipole expansion terms is typically 6 and FMM finest cell level, L, is 4.

When evaluating the validity of FMM3D as applied to this problem, a measure of its accuracy is important. A comparison between FMD with FMM3D, explicit electrostatics, and a conventional cutoff of 14 Åwas set-up. Preliminary tests with the explicit method showed that FMD ran the 5CB model nearly eight times slower than it did with FMM3D. Resources were not available to exactly repeat the entire 5CB sequence of runs. A compromise was to select a set of velocity and coordinate restart files close to 1.6 ns into the FMM3D-based run. FMD was restarted with these files, specifying use of explicit electrostatics or a simple cutoff. Both models were allowed to run for an additional 100,000 time steps (100 ps), and their trajectories compared with the FMM3D results over the same period.

Since this is a dynamic problem, small changes will tend to accumulate as the atoms move under slightly different forces. However, FMM3D follows the explicit results for nearly 20,000 time steps before divergence becomes significant, at least in electrostatic energy. Also, the electrostatic energy using a cutoff for 14 Å is not accurate. The features in the energy trajectories, such as the energy in bonds and bond angles were found to remain remarkably similar. The FMM3D model trajectories, tended to track those of the explicit model much better than those in the cutoff model. The close agreement of the FMM3D and the explicit method makes it easy to justify the technique as applied here, especially given the performance improvement.

The application of the FMD code to the 5CB droplet of 944 molecules (35,872 atoms) shows a clear correlation between the droplet shape and the nematic director: the droplet becomes an oblate spheroid, flatter along the director by about 10%. It also shows more than a 20 Å thick surface effect on the nematic orientational order at temperature 300 K. Surface molecules tend to orient tangentially to the sphere. Our simulation of a 118 molecule droplet shows little to no nematic order, which is contrary to a previous study by Tsige et al. [30]. However, the nematic order still persists in our simulations of bulk systems at temperatures higher than the measured nematic–isotropic transition temperature. Fig. 9 shows slice views of the 5CB system at the initial time and at 1.16 ns. The change of the system shape and molecular orientations on the surface are obvious. Indeed, the cluster transformed from a cube to a sphere in about 500 ps. However, the reconstruction and orientation order changes continue throughout the simulations. Details of the MD simulation on 5CB droplet are described elsewhere [31].

4. Conclusion

The MD code FMD with the FMM3D implementation of the fast multipole algorithm scales as $P^{0.8}$ as the model size increases to 99,522 atoms, with full atomic interactions. Use of FMM3D improves the efficiency significantly over the direct calculation of long-range interactions with accuracy much better than the cutoff method. The scalability is better than $P^{0.8}$ on up to 32 nodes but lags the overall FMD performance as the number of nodes increases. Improvements aimed at reducing the communication bottleneck, or modifying FMM3D's partitioning scheme to reduce overhead may help improve performance.

The scalability and efficiency of FMD with FMM3D are similar to NAMD2 with DPMTA for nodes up to 16. FMD with FMM3D performs better than NAMD2 with DPMTA on high nodes above 16 in speed. Both packages show no additional speedup for 64 or 128 node jobs.

While the fast multipole method does yield extremely high accuracies, one must be careful not to interpret this to mean acceptable energy stability. Multiple time stepping based on the assumption of slow variation of the long range terms does show stability sensitivity based on the cycle size. This is especially true for the long duration simulations required by the large structures used in material properties studies.

Copies of the FMD source code are available by request. Please address your interest to Ruth Pachter at the address above, or by e-mail to Ruth. Pachter@afrl.af.mil.

Acknowledgements

This work was supported in part by the DoD Common High Performance Software Support Initiative (CHSSI). Grants of high performance computer time were provided by the Army High Performance Computing Research Center, MN; the Maui High Performance Computing Center, HI; and the Aerospace Systems Center Major Shared Resource Center, OH.

References

[1] K.R. Amundson, M. Srinivasarao, Liquid-crystal-anchoring transitions at surfaces created by polymerization-induced phase separation, Phys. Rev. E 58 (1998) R1211–R1214.
[2] T.C. Bishop, R.D. Skeel, K. Schulten, Difficulties with multiple time stepping and fast multipole algorithm in molecular dynamics, J. Comput. Chem. 18 (14) (1997) 1785–1791.
[3] J. Board, Z. Hakura, W. Elliot, W. Rankin, Scalable Variants of Multipole-Accelerated Algorithms for Molecular Dynamics Applications, Technical Report TR94-006, Duke University, Durham, NC, 1994.
[4] B.R. Brooks, R.E. Bruccoleri, B.D. Olafson, D.J. States, S. Swaminathan, M. Karplus, Charmm: a program for macromolecular energy, minimization, and dynamics calculations, J. Comput. Chem. 4 (2) (1983) 187–217.
[5] A.T. Brünger, X-PLOR, Version 3.1: A System for X-Ray Crystallography and NMR, 1992, The Howard Hughes Medical Institute and Department of Molecular Biophysics and Biochemistry, Yale University, New Haven, CT.
[6] H. Cheng, L. Greengard, V. Rokhlin, A fast adaptive multipole algorithm in three-dimensions, J. Comput. Phys. 155 (1999) 468–498.
[7] J.A. Clarke, Emulating shared memory to simplify distributed-memory programming, IEEE Comput. Sci. Eng. (1997) 55–62.
[8] T. Darden, L. Pedersen, A. Toukmaji, M. Crowley, T. Cheatham, Particle-mesh-based methods for fast ewald summation in molecular dynamics simulations, in: Proceedings of the Eighth SIAM Conference on Parallel Processing for Scientific Computing, Society for Industrial and Applied Mathematics, Minneapolis, MN, March 1997.
[9] H.-Q. Ding, N. Karasawa, W.A. Goddard III, Atomic level simulations on a million particles: the cell multipole method for coulomb and london nonbond interactions, J. Chem. Phys. 97 (6) (1992) 4309–4315.
[10] H.-Q. Ding, N. Karasawa, W.A. Goddard III, The reduced cell multipole method for coulomb interactions in periodic systems with million-atom unit cells, Chem. Phys. Ltrs. 196 (1/2) (1992) 6–10.
[11] M.O. Fenley, W.K. Olson, K. Chua, A.H. Boschitsch, Fast adaptive multipole method for computation of electrostatic energy in simulations of polyelectrolyte DNA, J. Comput. Chem. 17 (8) (1996) 976–991.
[12] F. Figueririido, R.M. Levy, R. Zhou, B.J. Berne, Large scale simulation of macromolecules in solution: combining the periodic fast multipole method with multiple time step integrators, J. Chem. Phys. 106 (23) (1997) 9835–9849.
[13] M. Folk, L. Kalman, W. Whitehouse, HDF User's Guide, 4.1r1 Edition, National Center for Supercomputing Applications, Champaign, IL, May 1997.
[14] L. Greengard, W. Gropp, A parallel version of the fast multipole method, in: Parallel Processing for Scientific Computing, Society for Industrial and Applied Mathematics, 1989, pp. 213–222.
[15] L. Greengard, V. Rokhlin, A fast algorithm for particle simulations, J. Comp. Phys. 73 (1987) 325–348.
[16] L. Greengard, V. Rokhlin, On the Efficient Implementation of the Fast Multipole Algorithm, Research Report YALEU/DCS/RR-602, Yale University Department of Computer Science, February 1988.
[17] R. Hockney, J. Eastwood, Computer Simulation Using Particles, Adam Hilger, New York, NY, 1988.
[18] Y.C. Hu, On the accuracy of Anderson's fast n-body algorithm, in: Proceedings of the 8th SIAM Conference on Parallel Processing for Scientific Computing, Society for Industrial and Applied Mathematics, Minneapolis, MN, 1997.
[19] L. Kalé, R. Skeel, M. Bhandarkar, R. Brunner, A. Gursoy, N. Krawetz, J. Phillips, A. Shinozaki, K. Varadarajan, K. Schulten, NAMD2: greater scalability for parallel molecular dynamics, J. Comp. Phys. 151 (1) (1999) 283–312.
[20] F.S. Lee, A. Warshel, A local reaction field method for fast evaluation of long-range electrostatic interactions in molecular simulations, J. Chem. Phys. 97 (5) (1992) 3100–3107.
[21] D. MacKenzie, Autoconf, 2.12 Edition, 1996, Free Software Foundation.
[22] A. McKenney, R. Pachter, Implementation issues for fast multipole implementations for molecular dynamics simulations, in: Proceedings of the SIAM Annual Meeting on Society for Industrial and Applied Mathematics, Kansas City, MO, 1996.
[23] A. McKenney, R. Pachter, S. Patnaik, W. Adams, Molecular dynamics simulations of a siloxane-based liquid crystal using an improved fast multipole algorithm implementation, in: Proceedings of the Materials Research Society Symposium on Materials Research Society, Vol. 408, Boston, MA, 1996, pp. 99–105.
[24] M. Nelson, W. Humphrey, A. Gursoy, A. Dalke, L. Kalé, R. Skeel, K. Schulten, R. Kufrin, NAMD : a parallel, object-oriented molecular dynamics program, Int. J. Supercomput. Appl. High Performance Comput. 10 (4) (1996) 251–268.
[25] S. Plimpton, Fast parallel algorithms for short-range molecular dynamics, J. Comput. Phys. 117 (1995) 1–19.
[26] P. Procacci, T. Darden, M. Marchi, A very fast molecular dynamics method to simulate biomolecular systems with realistic electrostatic interactions, J. Phys. Chem. 100 (24) (1996) 10464–10468.
[27] P. Procacci, M. Marchi, G.J. Martyna, Electrostatic calculations and multiple time scales in molecular dynamics simulation of flexible molecular systems, J. Chem. Phys. 108 (21) (1998) 8799–8803.

[28] T. Schlick, R.D. Skeel, A.T. Brunger, L.V. Kalé, J.A. Board Jr., J. Hermanns, K. Schulten, Algorithmic challenges in computational molecular biophysics, J. Comput. Phys. 151 (1999) 9–48.

[29] P.N. Swarztrauber, FFTPack, Version 4, National Center for Atmospheric Research, Boulder, CO.

[30] M. Tsige, M.P. Mahajan, C. Rosenblatt, P.L. Taylor, Nematic order in nanoscopic liquid crystal droplets, Phys. Rev. E 60 (1) (1999) 638–647.

[31] Z. Wang, S. Patnaik, J. Lupo, R. Pachter, Large-scale molecular dynamics simulations of 4-*n*-pentyl-4′-cyanobiphenyl (5CB) liquid crystalline model in the bulk and as a droplet, Comput. Theor. Poly. Sci. 11 (2001) 375–387.

[32] M.S. Warren, J.K. Salmon, A Portable Parallel Particle Program, Los Alamos National Laboratory, Los Alamos, NM, 1994.

[33] C.A. White, M. Head-Gordon, Derivation and efficient implementation of the fast multipole method, J. Chem. Phys. 101 (8) (1994) 6593–6605.

[34] M.R. Wilson, Replicated data and domain decomposition molecular dynamics techniques for simulation of anisotropic potentials, J. Comput. Chem. 18 (4) (1997) 478–488.

[35] R. Zhou, B.J. Berne, A new molecular dynamics method combining the reference system propagator algorithm with a fast multipole method for simulating proteins and other complex systems, J. Chem. Phys. 103 (21) (1995) 9444–9459.