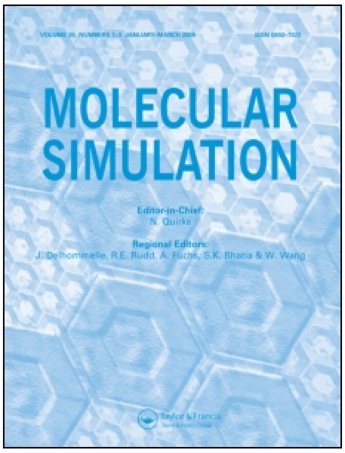


Molecular Simulation
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t713644482

Systolic Loop Methods for Molecular Dynamics Simulation, Generalised for Macromolecules

A. R. C. Raine[a]
[a] Department of Biochemistry, Cambridge Centre for Molecular Recognition, Cambridge, UK

To cite this Article Raine, A. R. C.(1991) 'Systolic Loop Methods for Molecular Dynamics Simulation, Generalised for Macromolecules', Molecular Simulation, 7: 1, 59 — 69

To link to this Article: DOI: 10.1080/08927029108022448

URL: http://dx.doi.org/10.1080/08927029108022448

Molecular Simulation, 1991, Vol. 7, pp. 59–69
Reprints available directly from the publisher
Photocopying permitted by license only

SYSTOLIC LOOP METHODS FOR MOLECULAR DYNAMICS SIMULATION, GENERALISED FOR MACROMOLECULES*

A.R.C. RAINE

Cambridge Centre for Molecular Recognition, Department of Biochemistry, Tennis Court Road, Cambridge CB2 1QW, UK

(Received September 1990, accepted November 1990)

Systolic loop programs have been shown to be very efficient for molecular dynamics simulations of liquid systems on distributed memory parallel computers. The original methods address the case where the number of molecules simulated exceeds the number of processors used. Simulations of large flexible molecules often do not meet this condition, requiring the three- and four-body terms used to model chemical bonds within a molecule to be distributed over several processors. This paper discusses how the systolic loop methods may be generalised to accommodate such systems, and describes the implementation of a computer program for simulation of protein dynamics. Performance figures are given for this program running typical simulations on a Meiko Computing Surface using different number of processors.

KEY WORDS: Proteins, polymers, molecular dynamics simulation, parallel computers.

1. INTRODUCTION

Simulation is an established and powerful tool for exploring the statistical thermodynamics of molecular systems [1]. The desire to simulate large flexible molecules arises in several situations: polymer scientists find it convenient to study the bulk and dynamical behaviour of novel systems by simulation [2]. Protein chemists use simulation to investigate and rationalize biochemical processes in terms of the structure and dynamics of protein molecules [3]. Furthermore, molecular dynamics and Monte Carlo simulations are used as powerful methods for exploring the conformational space of a complex molecule when attempting to determine its structure from 2-D NMR or X-ray diffraction data [4,5].

By their very nature, macro-molecular simulations pose a demanding computational problem: for reliable simulations of bulk polymers, many molecules must be simulated. For realistic simulations of even a single protein molecule, a large number of solvent molecules must be included in the simulation cell. The possibility of using the efficient systolic loop methods for molecular dynamics simulation on relatively cheap parallel computers is therefore very attractive.

*Presented at the CCP5 St. Andrews meeting on "Architecture and Algorithms in Condensed Phase Simulations", July 1990.

2. SYSTOLIC LOOP METHODS

The systolic loop methods described in [6, 7] were developed for simulating fluids composed of small rigid molecules, interacting via pair-wise forces only. Running on a transputer-based parallel computer, optimum efficiency (speed-up relative to a single processor) is achieved when the number of interaction sites in the system exceeds the number of processors by a factor of about fifteen. A description of one method, SLS (Systolic Loop, Single), is given:

2.1 The SLS Method

Consider a system of seven particles, interacting via pair-wise forces. The data representing the coordinates, and the forces acting upon each particle can be distributed over four processors in the manner shown in Figure 1.

The pair-wise forces 4-5, 3-6, and 2-7 can be evaluated in parallel. If the coordinate and force data are rotated around the processor ring, the interactions 3-4, 2-5 and 1-6 can be evaluated. If the data movements are repeated, then, after seven such systolic pulses, all the pair-wise interactions will have been evaluated, and the data will have returned to their original processors (Figure 2). The time taken for the calculation will be a third of the time that a single processor would take, but the total elapsed time will include the time taken to rotate all the data once round the loop.

2.2 The SLS-G Method

This scheme can be extended for the case where there are many more particles than processors. Consider a system of twenty one particles, distributed over the same four processors (Figure 3). If the data are moved as groups of particles rather than individually, then the *body* processors can evaluate forces between groups of particles, while the *head* processor can evaluate the forces between particles *within* each group. Furthermore, as the time taken for the calculation is proportional to the square of the size of the groups, while the communication time depends only on the total number of particles in the system, the calculations become more efficient as the size of the system increases. This method has been termed SLS-G (Systolic Loop, Single, Grouped)

2.3 Performance

The expected performance of the systolic loop methods is developed in [6], and can be summarised thus: if each group of atoms is of size n, and there are P such groups,

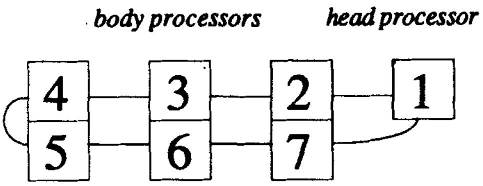


Figure 1 Distribution of particles over processors for the SLS method.

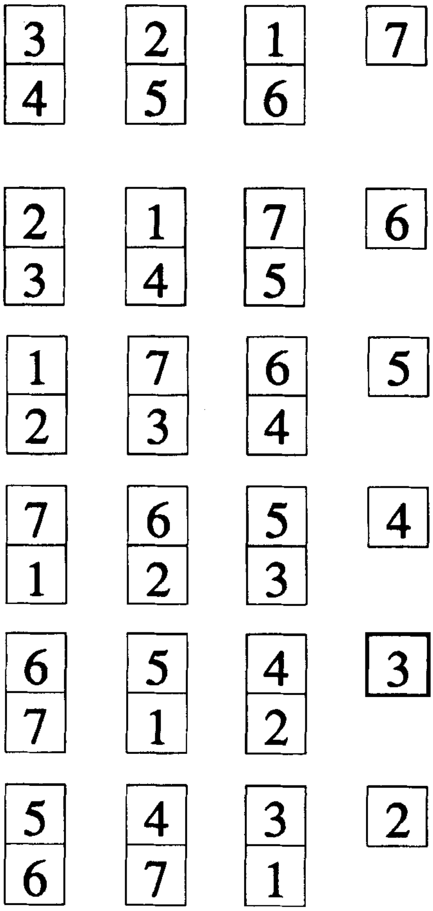
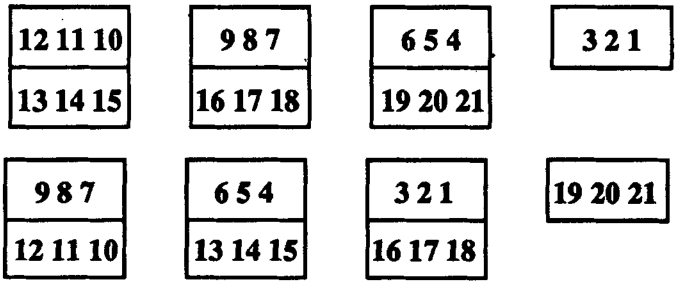


Figure 2 Data movements for the SLS method.



etc.

Figure 3 Distribution of data and data movements for the SLS-G method.

then the efficiency of the force evaluation stage of the calculation is given by:

$$E = \frac{Pn^2\, i}{Pn^2\, i + P(s + nc)} \qquad (1)$$

where i is the time taken for a pair interaction to be evaluated, s is the time taken to set up a communication between two processors and c is is the time taken, per atom, to communiate coordinate and force information.

It can be seen from equation (1) that efficiency depends on the size of the groups rather than directly on the number of processors (Figure 4). This means that, when the system to be simulated is of sufficient size, a large number of processors can be used without sacrificing efficiency.

3. FLEXIBLE MOLECULES

Simulation of fluids composed of flexible small molecules can readily be accommodated with the SLS-G method. The additional two-, three-, and four-body terms representing bond stretching, bond angle deformation, and bond torsion rotation (see Figure 5) can be arranged to be evaluated between atoms within individual groups. In fact, as these bonded interaction terms are few in comparison with the number of non-bonded terms (of order N rather than of order N^2), they can be evaluated by the *head* process in addition to its $1/2\ n(n\text{-}1)$ non-bonded terms of each systolic pulse, within the time taken by the *body* processes for their n^2 non-bonded terms.

For macromolecular simulations, it will often be the case that there are more processors available than there are molecules in the system under investigation. This means that, for the systolic loop methods to be used, any individual molecule must be split over two or more groups of atoms. The systolic loop methods described in [7] only guarantee that every *pair-wise* interaction can be taken into account, although obviously more complex terms may be evaluated if they involve atoms from within

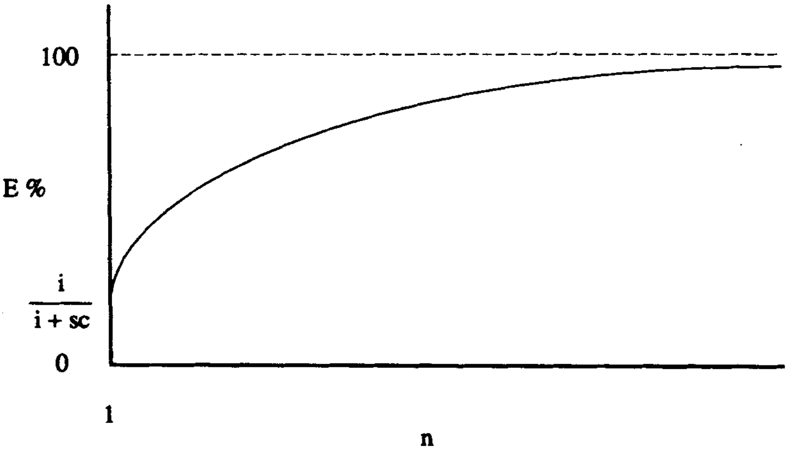


Figure 4 Calculated efficiency of the force evaluation stage of the SLS-G method as a function of group size.

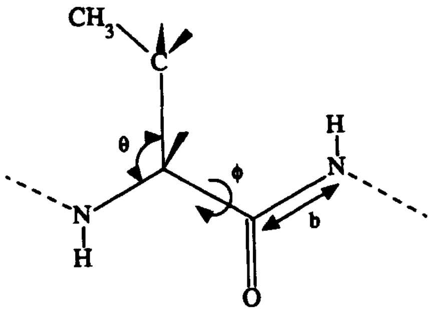


Figure 5 Bonded interactions in a flexible molecule.

one group, or between two groups. So, in order to use a systolic loop program for macromolecular simulations, some way of ensuring that all the triplets and quartets of atoms involved in bonded interactions within a molecule can be found in at most two groups.

3.1 Atom Division Rule

Inspection of successful and unsuccessful divisions of atoms into groups reveals a simple rule which, applied to group division, ensures that the bonded interactions can be accommodated.

Consider the division shown in Figure 6. The three-body interaction describing deformation of angle θ is defined by atoms (i), (ii), and (iii), which are from groups (2), (3) and (1) respectively. This interaction cannot be evaluated at any time during the systolic movement of coordinate data. If, however, the division of atoms is as shown in Figure 7, then the three- and four-body interactions between groups (1) and (3) do not involve atoms from group (2), and *vice-versa*. The generalisation of these observations is simple: no atom should be allowed to participate in more than one bonded term that crosses group boundaries.

3.2 Distributed Interaction Lists

Conventional simulation programs keep track of bonded interactions using linked-lists of atom indices [1], and a systolic loop program will have to do the same. In fact,

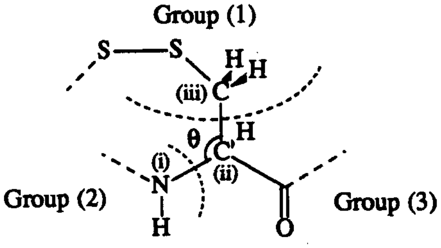


Figure 6 A division of atoms into groups which does not allow the bonded interaction to be accommodated within the systolic loop scheme.

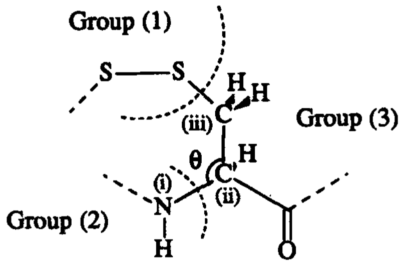


Figure 7 A division of atoms into groups which does allow the bonded interaction to be accommodated within the systolic loop scheme.

each processor will have to keep a set of P lists for each type of interaction. The first list of each type will apply before the first systolic pulse of data, the second before the second pulse and so on. These distributed lists can be calculated at the beginning of a simulation, once the initial assignment of atoms to processors has been made.

3.3 Consequences For Efficiency

As before, most of the bonded interactions can be evaluated by the *head* process, with little or no impact on the total time taken by the force evaluation. The additional inter-group interactions will add a small time, of order n. For some of the processors during some of the systolic pulses. Efficiency will only be affected if there is an uneven distribution of these additional interactions. This will depend on the topology of the molecules being studied, but is unlikely to have a large effect. A more significant source of inefficiency will be that, due to the application of the atom division rule, the groups will be of different sizes, and therefore at each systolic pulse each processor will complete its work in a different time. However, because it is unlikely that the same processor will take the longest time at any two subsequent pulses, and because of the loose synchronization between processors not direcly connected, even this load-imbalance is unlikely to degrade efficiency significantly: processors remote from one another can get temporarily ahead of, or lag behind, one another without causing bottle necks in the data-flow.

4. BOND LENGTH CONSTRAINTS: SHAKE

The time-step of a simulation, and therefore the rate at which it can explore phase space, is limited by the period of the fastest characteristic motions of the system being investigated. For flexible molecules, these will be the vibrations of bond lengths. Removing these degrees of freedom from the system, by fixing bond lengths, allows a longer time-step to be used.

For large flexible molecules, bond length constraints are customarily applied using the SHAKE method [8]. This takes an iterative approach at each time-step, applying a correction to each constrained bond in turn until all the constraints are satisfied, and allows the time-step of a simulation to be increased by a factor of two or three, at a relatively small cost.

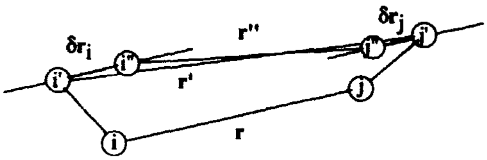


Figure 8 Vector relationships in the SHAKE method. r represents the bond ij before the unconstrained time-step, r′ the bond after the unconstrained step, and r″ the bond after the correction has been applied.

Can the SHAKE method be accommodated within the systolic loop scheme? In order to assess this, it is necessary to consider how SHAKE works. The SHAKE procedure applies its correction to the coordinates of atoms joined by constrained bonds *after* a normal unconstrained time-step has been made. Each such bond is corrected in turn and, because a correction to one bond may violate another constraint previously satisfied, the process is repeated until all the constrained bonds are within some small tolerance of their ideal lengths. The corrections are applied along the directions of the bond *before* the unconstrained time-step (Figure 8).

4.1 SHAKE in a Systolic Loop

(a) General case

In the general case, the atoms defining bonds which cross group (and therefore processor) boundaries will not be on adjacent processors. The cost of communicating the coordinates of such atoms between remote processors will be relatively high and, as SHAKE will iterate several times per time-step, will be prohibitive. The worst case would require a full systolic circlation of coordinates per iteration of SHAKE.

(b) Special case

Systems composed of un-branched chain molecules, with no cross-linking, form a special case which is more tractable. It is simple to arrange that the bonds crossing group boundaries only fall between adjacent processors. The communication required for each iteration of SHAKE will be small and local, and will therefore not offset the advantage gained by imposing constraints.

(c) Special case involving hydrogen atoms

In systems where hydrogen atoms are modelled explicity, the fastest motions will be vibration of H–X bonds (X being any other element). The time-step of simulations can be doubled even if just these bonds are constrained [8]. Since hydrogen atoms are monovalent, it is simple to arrange that no H–X bond crosses a group boundary. This ensures that the sets of constraints on different processors are completely independent, and can be applied in parallel with no communication penalty at all.

5. SIMULATION OF PROTEINS

The author's particular interest is in simulating protein systems, and the use of such simulations in the determination of protein structure from NMR data by simulated annealing. Therefore the implementation described here is designed for protein simulation. However, the methods used are generally applicable.

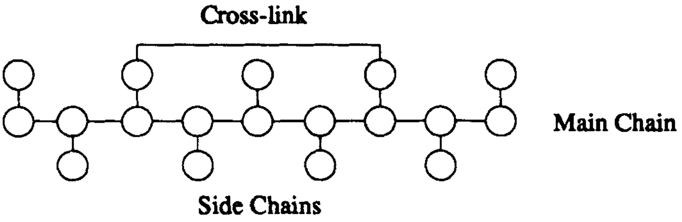


Figure 9 Cartoon of protein structure.

5.1 Structure Of Proteins

Proteins are large chain molecules composed of amino-acid residues. There are twenty common amino-acids, each having different chemical and physical properties. Some residues are capable of cross-linking the chain of a protein molecule either to itself, or to another molecule (Figure 9).

Because the active form of most proteins is a compact, globular and fairly rigid structure, the inclusion of hydrogen atoms explicitly is important: the detailed packing of residues in the interior of the molecule is strongly dependent on hydrogen-bonding and on van der Waals interactions between adjacent atoms.

5.2 Group Division For Protein Simulations

Because of the possbility of cross-linking of the protein chain, the strategy outlined in (b) above will not generally be appropriate. However, strategy (c) can be used to be sure that some advantage can be gained by using SHAKE. Consequentially a very simple division of atoms into groups is sufficient: group boundaries are made to fall only between amino-acid residues (Figure 10). This ensures that no H–X bonds cross processor boundaries, and therefore that SHAKE can be applied to the H–X bonds in each group independently and in parallel.

5.3 Implementation

A program (SLS-PRO) has been written, in occam 2, to run on Meiko and INMOS hardware, which will perform molecular dynamics simulations of protein systems using the SLS-G method. SLS-PRO implements the GROMOS [9] force-field for

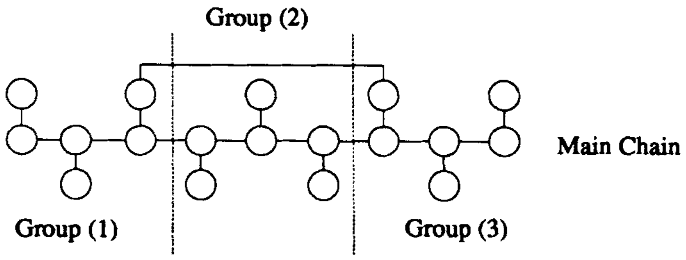


Figure 10 Cartoon of protein structure, showing group division.

Table 1 Features of protein simulation program SLS-PRO

Constant energy dynamics	– Simulation in the micro-canonical ensemble
Constant temperature dynamics	– Simulation in a pseudo-canonical ensemble (using the Berendsen thermostat)
Bond length constraints	– SHAKE may be applied to bonds involving hydrogen atoms
Distance restraints	– Atom-atom distances may be restrained (using a harmonic/linear potential)

biomolecular simulations, and reads GROMOS format files. This allows other programs from the GROMOS suite to be used to prepare input data on the host computer. Temperature control can be applied using the Berendsen thermostat [10], and the equations of motion are integrated using the Verlet leap-frog method [11]. A summary of the features of SLS-PRO is given in Table 1.

5.4 *Performance*

The performance of SLS-PRO was assessed by comparing simulations of crambin (a small protein of ~ 400 atoms per molecule) on a Meiko Computing Surface of three, five and twelve processors, with equivalent simulations run using GROMOS on a micro VAX 3600. The results are shown in Table 2.

As can be seen, for the simulations taking all pair-wise interactions into account, the speed-up is more or less linear. When a 1.0 nm cutoff is applied to the interactions (reducing the number of terms evaluated by a factor of three), the time taken for inter-processor communication is more significant, and the efficiency drops. For crambin, using twelve processors requires that each group of atoms contains those from only two amino-acid residues, and therefore the inefficiency due to imperfect load-balancing is observable. Simulations of larger molecular systems would allow more processors to be used effectively, but in the present version of the program, memory limitations prevent much bigger simulations being attempted. However, overall, each T800 used contributes 1–1.5 micro VAX 3600 processor equivalents for these calculations.

5.5 *Problems and Future Improvements*

As mentioned in section 1, the object of extending systolic loop methods to macromolecules, and especially poteins, is to allow simulation of large systems to be

Table 2 Performance of SLS-PRO for a simulation of crambin (all times are in seconds)

	Time per 100 steps (1.0 nm cutoff)	*Speedup relative to VAX*	*Time per 100 steps (all pairs)*	*Speedup relative to VAX*	*Average group size (atoms)*
microVAX 3600	705	1.0	2045	1.0	400
3 T800s	215	3.3	450	4.5	80
5 T800s	165	4.4	260	7.9	44
12 T800s	60	11.8	125	16.4	17

performed rapidly. Although SLS-PRO demonstrates that the generalisation is straightforward, the code as it stands does not meet this objective for two reasons:

(1) The naive manner in which the distributed lists or bonded interactions is derived from the original, complete, list for a system is extremely demanding of memory. This means that the size of the system which an be simulated is limited by the memory available to the *head* processor, where this division is carried out. In practice, using 4Mbyte processors, the upper limit is currently 800 atoms on 12 processors. Modifications are in hand to remove this limitation, and a capacity of approximately 500 atoms per *body* processor will be achieved for the same 4MByte nodes.

(2) SLS-PRO cannot cope with solvent molecules in addition to the protein molecule(s). There is no reason why water should not be included, and the required additions to the code are also in hand.

Further modifications are intended, including the implementation of a more sophisticated thermostat such as that of Nosé [12, 13], and the ability to read and write X-PLOR [14] format files.

6. APPLICABILITY OF SYSTOLIC LOOP METHODS TO LARGE SYSTEMS

It should be noted that the systolic loop methods are only optimum for all-pairs simulations. However, simulations applying a spherical cutoff to the non-bonded interactions of the same order of magnitude as the size of the system do not depart significantly from optimum efficiency [6]. Simulations where the range of the non-bonded interactions is much smaller than the size of the system would be performed better using a linked-cells method [1], for which an efficient geometric parallelisation has been described by Petersen and Perram [15] among others.

Conversely, simulations where the non-bonded interaction range is larger than the size of the system are often performed using the Ewald summation method [16], or a grid method such as P^3M [17] to handle the long range interactions. These methods would require a geometric decomposition of the problem similar to that described in [15], and would not be accommodated efficiently within the systolic loop schemes used in this work.

Currently available software packages for molcular dynamics simulation of protein systems rely on applying a smoothed spherical cutoff, of 0.8–1.5 nm, to the non-bonded and electrostatic interactions using an atomic neighbour list which is updated periodically by searching over all pairs of atoms. This approach is well suited to the systolic loop methods and seems to be reasonably efficient, even for fairly large systems (a simulation cell whose dimensions were ten times the cutoff radius would contain over 10^5 atoms – most 'large' systems would contain fewer atoms than this). However, applying such a cutoff, even with smoothing, cannot be said to be treating long range electrostatic interactions correctly. Ultimately, for more accurate simulations and for very large protein and water systems, a parallel linked-cell method using P^3M to deal with long range interactions would seem to be the most appropriate.

7. CONCLUSION

The work described in this paper sets out to find a generalisation of the systolic loop methods to allow molecular dynamics simulation of macro molecules. It was found that simple rules governing the data decomposition of the problem allowed the many-body terms of the potential function, and the SHAKE method of applying bond-length constraints, to be accommodated without changing the original scheme.

The computer program (SLS-PRO), which was written applying these rules, performs effectively for typical protein simulation problems. Running on a twelve processor machine, SLS-PRO gives performance equivalent to a mini super computer such as an Alliant or Convex. The generalised systolic loop method is not limited to protein structure, however, but is appropriate for simulations of *any* large flexible molecular system.

Acknowledgements

This work was performed while the author was a graduate student of the Department of Chemistry at the University of York, UK. The author is grateful for the support of Dr. R. Hubbard at York, Dr. D. Fincham of the University of Keele, and Dr. W. Smith of SERC Daresbury laboratory for their help and advice. The author also thanks Polygen Corporation for the provision of equipment, and the Edinburgh Concurrent Super computer project for computer time.

References

[1] Allen M.P. and Tildesley D.J. *Computer Simulations of Liquids* Pub. O.U.P. (1987).
[2] Ryckaert J.P. and Bellemans A. *Chem. Soc. Farad. Discuss.*, **66**, 95 (1978).
[3] Karplus M. and Petsko G.A. *Nature*, **347**, 631 (1990).
[4] Brünger A.T., Karplus M. and Petsko G.A. *Acta Cryst.*, **A45**, 50 (1989).
[5] Clore G.M. and Gronenborn A.M. *Crit. Rev. Biochem.*, **24**, 479 (1989).
[6] Fincham D. *Mol. Sim.* **1**, 1 (1987).
[7] Raine A.R.C., Fincham D. and Smith W. *Comput. Phys. Comm.*, **55**, 13 (1989).
[8] Ryckaert J.P., Ciccotti G. and Berendsen H.J.C. *J. Comp. Phys.*, **23**, 1311 (1977).
[9] Van Günsteren W.F., Berendsen H.J.C., Hermans J., Hol W.G.J. and Postma J.P.M. *Proc. Natnl. Assoc. Sci. USA*, **80**, 4315 (1983).
[10] Berendsen J.J.C., Postma J.P.M., van Günsteren W.F., di Nola A. and Haak J.R. *J. Chem. Phys.*, **81**, 368 (1984).
[11] Verlet L. *Phys. Rev.*, **159**, 98 (1967).
[12] Nosé S. *Molec. Phys.*, **52**, 255 (1984).
[13] Hoover W.G. *Phys. Rev. Sect. A*, **31**, 1695 (1985).
[14] Brünger A.T., Kuryan J. and Karplus M. *Science*, **235**, 458 (1987).
[15] Petersen H.G. and Perram J.W. *Molec. Phys.*, **67**, 849 (1989).
[16] Sangster M.J.L. and Dixon M. *Adv. Phys.*, **25**, 247 (1976).
[17] Eastwood J.W., Hockney R.W. and Lawrence D.N. *Comput. Phys. Commun.*, **19**, 215 (1980).