10.1098/rsta.2004.1419

THE ROYAL SOCIETY

DL_POLY_3: the CCP5 national UK code for molecular-dynamics simulations

By I. T. Todorov and W. Smith

Computational Science and Engineering Department, CCLRC Daresbury Laboratory, Daresbury, Warrington WA4 4AD, UK
(i.t.todorov@dl.ac.uk; w.smith@dl.ac.uk)

Published online 16 July 2004

DL_POLY_3 is a general-purpose molecular-dynamics simulation package embedding a highly efficient domain decomposition (DD) parallelization strategy. It was developed at Daresbury Laboratory under the auspices of the Engineering and Physical Sciences Research Council. Written to support academic research, it has a wide range of applications and will run on a wide range of computers; from single-processor workstations to multi-processor computers, with accent on the efficient use of multi-processor power. A new DD adaptation of the smoothed particle mesh Ewald method for calculating long-range forces in molecular simulations, incorporating a novel three-dimensional fast Fourier transform (the Daresbury Advanced Fourier Transform), makes it possible to simulate systems of the order of one million particles and beyond. DL_POLY_3 structure, functionality, performance and availability are described in this feature paper.

Keywords: DL_POLY; molecular-dynamics simulation; software

1. Introduction to molecular dynamics

Molecular dynamics (MD) is an important and widely used theoretical tool that allows researchers in chemistry, physics and biology to model the detailed microscopic behaviour of many different types of systems, including gases, liquids, solids, surfaces and clusters. In an MD simulation, the classical equations of motion governing the microscopic time evolution of a many-body system are solved numerically, subject to the boundary conditions appropriate for the geometry or symmetry of the system. Thus, MD methodology, based on the principles of classical mechanics, can provide information of the microscopic dynamical behaviour of the individual atoms constituting a given system. This information can be used to monitor the microscopic mechanisms of energy and mass transfer in chemical processes, and dynamical properties such as absorption spectra, rate constants and transport properties can be calculated. In addition to providing a microscopic dynamical picture, MD can also be employed as a means of sampling from a statistical mechanical ensemble and determining equilibrium properties. These properties include average thermodynamic

One contribution of 12 to a Theme 'Discrete element modelling: methods and applications in the environmental sciences'.

quantities (pressure, volume, temperature, etc.), structure, and free energies along reaction paths.

The basic input needed for the classical equations of motion for a model system comprises an iteration time-step (Δt, compulsory for the numerical integration), particle positions (x_i, identifying structure), particle velocities and masses (v_i, m_i,), and a description of interparticle interactions including fields external to the system (resulting in particle forces, f_i). The better this description, the higher the quality of the results of an MD simulation. More input may be needed (temperature, T, pressure, P, volume, V, etc.) depending on what type of thermodynamic ensemble the equations of motion generate.

There are various methods for numerical integration of Newton's equations of motion, of which there are two that excel in their simplicity of implementation, numerical stability, low memory requirement and satisfactory accuracy: (i) leapfrog Verlet (LFV) and (ii) velocity Verlet (VV). Further reading on such methods can be found in Allen & Tildesley (2002), Gear (1971) and Swope *et al.* (1982).

Leapfrog Verlet

(0) $x_i(t)$, $v_i(t - \frac{1}{2}\Delta t)$.

(1) $f_i(t)$, calculated afresh.

(2) $v_i(t + \frac{1}{2}\Delta t) = v_i(t - \frac{1}{2}\Delta t) + \Delta t \dfrac{f_i(t)}{m_i}$.

(3) $x_i(t + \Delta t) = x_i(t) + \Delta t v_i(t + \frac{1}{2}\Delta t)$.

Velocity Verlet

(0) $x_i(t)$, $v_i(t)$, $f_i(t)$.

(1) $v_i(t + \frac{1}{2}\Delta t) = v_i(t) + \frac{1}{2}\Delta t \dfrac{f_i(t)}{m_i}$.

(2) $x_i(t + \Delta t) = x_i(t) + \frac{1}{2}\Delta t v_i(t + \frac{1}{2}\Delta t)$.

(3) $f_i(t + \Delta t)$, calculated afresh.

(4) $v_i(t + \Delta t) = v_i(t + \frac{1}{2}\Delta t) + \frac{1}{2}\Delta t \dfrac{f_i(t + \Delta t)}{m_i}$.

Without doubt, there are more-accurate and sophisticated algorithms, such as the classic Runge–Kutta and gear predictor–corrector (Gear 1971). However, such algorithms are more time-consuming and demanding of memory, and generally have less long-term stability and thus are less suitable for MD simulation of systems comprising many particles.

2. Background of DL_POLY

The DL_POLY project originated in the United Kingdom's Collaborative Computational Project, widely known as CCP5 (Smith 1987). The objective was to develop a new molecular simulation package to meet the needs of the UK academic community for a general-purpose molecular-dynamics (MD) code which would exploit

the emerging parallel computers to the fullest advantage as well as satisfy a number of other important criteria, particularly the demand for 'open' software permitting verification and extension by the user (meaning that the package should be available in the form of source code). In addition, the package was required to support a wide range of applications; for example, macromolecules (both biological and synthetic), complex fluids and ionic materials of high complexity. The DL_POLY_3 package is the outcome of these requirements.†

With support from the Engineering and Physical Sciences Research Council (EPSRC), the DL_POLY package has been under continuous development in the UK since 1994. It is available free of cost to academic scientists pursuing research of non-commercial nature and has been applied in a broad range of scientific studies since its first official release in 1996 (Smith & Forester 1996) as DL_POLY_2. Over 1000 user licences have been taken out worldwide since 1994. The original code, DL_POLY_2, was based on a replicated data (RD) (Smith 1991) parallelization strategy, but recent developments have introduced a domain decomposition (DD) (Pinches *et al.* 1991) version, DL_POLY_3, to permit simulation of systems of the order of one million atoms and beyond.

This article describes the structure and capabilities of DL_POLY_3 and where applicable close comparison with DL_POLY_2 is drawn. (We hereafter use DL_POLY when we describe features general to either version package.) The following section describes the DL_POLY structure and overall design, with special attention being paid to the difference between the parallelization strategies in DL_POLY_2 and DL_POLY_3 and the consequent benefits and drawbacks of either. The subsequent sections describe DL_POLY functionality and its performance with three model systems on an IBM SP4 cluster (HPCx). The final section outlines the availability of the package.

3. Structure

(*a*) *Overall design*

As supplied, DL_POLY consists of a single UNIX directory with several subdirectories as follows:

(1) *source*, FORTRAN source code;

(2) *build*, makefiles for building DL_POLY for different platforms;

(3) *execute*, program execution subdirectory;

(4) *data*, input and output examples;

(5) *java*, the DL_POLY Java graphical user interface (GUI);

(6) *utilities*, analysis and data-preparation programs;

(7) *manuals*, documentation.

† There is also the DL_POLY_2 version and a number of spin-offs, such as DL_MULTI, DL_PROTEIN and DL_MESO.

The *source* subdirectory consists of around 150 subroutines written in a uniform style in FORTRAN 77.† The adopted style has been chosen to improve the legibility of the code and simplify the encoded algorithms, in keeping with the best coding practice. The basic design philosophy assumes that the users assemble these into tailor-made programs to meet their own requirements. However, to assist the inexperienced, facilities to build a 'standard' version are supplied (in the *build* subdirectory, see below). To simplify data transfer between subroutines, the use of COMMON blocks is limited to those demanded by certain communication routines in the parallel implementation, and to facilitate array allocation and definition. An important feature of the subroutines is that they incorporate parallelism automatically (see below); data distribution between processors and the necessary communications are localized in a few dedicated subroutines.

The *build* subdirectory stores the UNIX makefiles that assemble the executable versions of the code. A number of makefiles for various combinations of multi-processor platforms with specific operating systems and message passing interface (MPI) technologies are available. To use, the required makefile is copied into the source subdirectory where the code is compiled and linked. The final executable from this procedure is then placed in the execute subdirectory, which is supplied with example UNIX scripts showing how it is used on various machines. The input data for a given job are located in this subdirectory at run time. The output is also written here.

The *data* subdirectory contains examples of input and output from DL_POLY. These constitute the benchmarks and test cases relevant to the specific version of DL_POLY.

The *java* subdirectory provides a simple GUI for all DL_POLY codes. It allows viewing of the molecular system, simple (background) job submission and construction and analysis of DL_POLY input and output files. Based on the Java language, the GUI is highly portable and easily extended by the user.

The *utilities* subdirectory stores a collection of analysis and data-preparation tools intended to help the users further.

The final subdirectory is *manuals*, which houses the DL_POLY documentation. The documentation takes the form of two manuals: a user manual and a reference manual. The former is a guide to using the code in scientific work, the latter is a much more detailed description of the DL_POLY code and is suitable for software developers.

(*b*) *Parallelization*

Nowadays, parallelism is an obligatory design feature for every sort of simulation software. In both DL_POLY_2 and DL_POLY_3 parallelism is inherently embedded, although two very different strategies are employed. It is also important to note that the codes can also be run on *serial* computers without modification (for DL_POLY_3 the MPI is compulsory).

The communication between nodes is handled by standard MPI subroutines. The communications are embedded within dedicated subroutines and are easily adapted for different parallel computers.

† A few new subroutines such as SPME subroutines are written in FORTRAN 90. Our aim is that in the future the whole package will be FORTRAN 90 compliant.

The parallel strategy underlying DL_POLY_2 is the *replicated data* (RD) scheme (Smith & Forester 1994*a*,*b*). The essential ingredients of the RD algorithm are as follows.

(1) Each processor has a complete copy of the coordinates and velocities of all the atoms in the system.

(2) Each node of the parallel computer is assigned the task of calculating a specific subset of all the possible pair forces. Subsets are aimed to be nearly equivalent in length.

(3) The total force on an individual atom is obtained as a global sum, over all nodes, of all the pair forces associated with the atom. (This step results in every node having a complete replica of the computed force arrays.)

(4) Independent subsets of the equations of motions are integrated on each node.

RD is a popular parallelization strategy for medium-range parallel machines, though there are three fair criticisms that can be made. First, in this algorithm, the fact that all the nodes have a copy of the configuration data implies that this strategy is expensive in terms of memory. This proves to be very expensive for large systems (more than 10^4 particles). Second, the global sum required to complete the force calculations imposes a strict limitation on the efficiency of the algorithm when the number of nodes is relatively large. This gives rise to poor scaling when the number of nodes used is relatively large (greater than 10^2). Third, the integration of the equations of motion does not exploit the inherent parallelism efficiently as additional communications are necessary to maintain data replication.

DL_POLY_3 is based on a very progressive parallelization strategy—the *domain decomposition* (DD) scheme. Domain decomposition is a universal strategy adopted in many areas of mathematical modelling. In MD it is generally adapted from the *link-cell* method (Hockney & Eastwood 1981; Pinches *et al.* 1991; Rapaport 1991). It is appropriate for systems in which the potential cut-off is very short in relation to the size of the system being simulated, and hence has applications in areas as diverse as polymers, microscale hydrodynamics, phase transitions, surfaces, micelles, etc. Parallel adaptations of the link-cell method are particularly powerful as they enable extremely large systems to be simulated very cost effectively.

The serial form of the algorithm is well documented (Essmann *et al.* 1995). It suffices here to outline only the basics. According to the basic strategy, the MD cell is divided into sub-cells, with width slightly greater than the radius of the cut-off. A simple N-dependent (where N is the number of particles) algorithm assigns each atom to its appropriate sub-cell and a *linked list* is used to construct a logical chain identifying common cell members. A subsidiary *header* list identifies the first member of the chain. This allows all the atoms in a cell to be located quickly. The calculation of the forces is treated as a sum of interactions between sub-cells, in the course of which all pair forces are calculated. Allowance for periodic boundary conditions is easily made. The algorithm performs well on serial machines because it greatly reduces the time spent in locating interacting particles.

Parallel versions of this algorithm are easily constructed (Pinches *et al.* 1991; Rapaport 1991). The MD cell is divided into geometrically identical regions (domains) and each region is allocated to a node. The mapping of the regions on the array of nodes is

a non-trivial problem in general, although specific solutions for machines like hypercubes are much easier to obtain. The region on each node is further subdivided into sub-cells as in the serial algorithm. The coordinates of the atoms in sub-cells adjacent to the boundaries of each region are passed on to neighbouring nodes sharing the same boundaries (*exchange of boundary data*). After which each node may proceed to calculate all pair forces in its region *independently*. No further communication between the nodes is necessary until after the equations of motion have been integrated: particles which have moved out of their node region must be reallocated to a new node.

Regarding the exchange of boundary data, it is crucial that data can only be passed in complementary directions (north–south, east–west, up–down in three dimensions) at any given instant and in between exchanges, the exchanged data must be re-sorted before the next exchange. This is necessary to ensure the corner and edge sub-cell data are correctly exchanged between regions sharing edges and corners, rather than faces.

Although the DD algorithm is specifically designed for systems with short-range forces, it can also be used for systems with Coulombic forces. Work has been done (Bush *et al.* 2004) to produce a new DD adaptation of the smoothed particle mesh Ewald (SPME) method as devised by Essmann *et al.* (1995) for calculating long-range forces in molecular simulations. In this adaptation two strategies are employed to optimize the traditional Ewald sum. The first is to calculate the (short-ranged) *real space* contributions to the sum using the DD method as outlined above. The second is to use a fine-grained mesh in *reciprocal space* and replace the Gaussian charges by finite charges on mesh points. The mesh permits the use of three-dimensional fast Fourier transforms (3D FFTs) (Brigham 1988). In DL_POLY_3 we have used the Daresbury Advanced Fourier Transform (DAFT) (Bush 1999), which is a novel, fully distributed, parallel implementation of the 3D FFT that fits well with the DD concept. DL_POLY_3 only allows the use of the SPME method for calculation of Coulombic forces since the standard Ewald method does not benefit from the DD concept and so it would slow down the performance of DL_POLY_3 to that of DL_POLY_2.

Overall the algorithm is simple, powerful and flexible. However, unlike the RD concept, the DD concept only works well if the simulated system is reasonably isotropic. This is essential because the uniformity of the particle distribution guarantees good load balancing of the parallel force calculations. Thus special considerations must be given to systems which are inherently non-uniform, such as slabs, isolated clusters, etc.—systems with any kind of vacuum gap.† Some awareness of the consequences of such structures is required by the user if the code is to be used efficiently.

4. Functionality

(*a*) *Molecular systems*

DL_POLY incorporates a wide range of the functions required from a modern molecular simulation package. The potential range of applications is vast (Smith *et al.* 2002):

† When there is a vacuum gap in a system, the very idea of parallelization (quasi-equal workload on each node) breaks down since nodes mapping the vacuum space would be seriously underloaded (if not idle) and hence parallelization is inefficient.

from biological systems and polymeric materials, through to inorganic materials and solutions. A brief idea of the range of DL_POLY applications can be obtained from the following list.

(1) Liquids and solutions.

   (a) Structure and dynamics: Ar, SF_6, H_2O, $C_nH_{2n-1}(OH)$, electrolyte and polyelectrolyte solutions (PEO, complexes), etc., H–D isotopic substitution, H-bond network, hydrophobicity studies, cluster analysis, etc.

   (b) Spectroscopy: coumarin, fluoroprobe, tetracene, naphthalene, solvation energy, RDF analysis, polarizability studies (polar/non-polar solutions), conformation, luminescence in silicate glasses doped with rare-earth ions, etc.

(2) Structure and materials: pressure and temperature studies, phase transitions, order–disorder transitions, melts, solid solutions, doping effects, cluster analysis, etc.

   (a) Ionic solids: MgO, MnO, KNO_3, TiO_2, SiO_2, GeO_2, $AlPO_4$, ZrO_2, etc.

   (b) Simple metals and alloys: Al, Ag, Ni, Cu, etc.

   (c) Molecular crystals: carbonates, valinomycine, p-tephenyl, etc.

(3) Amorphous systems: glasses and synthetic polymers, transition temperature, dynamic property analysis, thermal properties, surface properties, doping effects, cluster analysis, etc.

(4) Biological systems: membranes, proteins, biopolymers, surfactants, etc., dynamical properties (diffusion), conformational properties, solutions, etc.

(5) Surfaces and interfaces, etc.: solid–solid (crystal growth), solid–liquid (condensation, adsorption, wetting), solid–gas (gas in porous zeolites: adsorption, diffusion), etc.

(6) Catalysis: homogeneous and heterogeneous catalysis, free energy pathways, diffusion, etc.

(7) Complex materials: liquid crystals, chiral aluminium phosphates, clathrate hydrates, novel simulations (abstract phase-transitions and colloids).

Within these broad categories, a wide selection of molecular models is possible. DL_POLY_3, as yet, offers only a choice of flexible molecules with rigid bonds, whereas DL_POLY_2 also offers rigid and partly rigid molecules (rigid bodies connected by rigid bonds).† Flexible molecules are described in terms of extensible bonds and variable bond angles (see the force field description below). Rigid molecular units are described by fixed geometry assemblies of atoms. Partly rigid molecules can be constructed from standard bond constraints (Ryckaert *et al.* 1977) or by rigid units

† Our aim is to include rigid bodies in DL_POLY_3. However, their maximum size will be limited to that of a link cell (*ca.* 8 Å) in order to make sure that no rigid body is partly shared between neighbouring domains in this way to keep the DD concept intact.

linked by flexible or constrained bonds. The dynamical treatment of these different models requires markedly different techniques (see below).

In the case of systems with Coulombic interactions, the charged entities are atoms, modelled either as point charges or as polarizable cores and shells (Fincham & Mitchell 1993). Finer details of the models may be gleaned from the following section describing the force field.

(*b*) *Force fields*

A force field is a set of empirical functions and associated parameters which collectively describe the interactions between atoms and molecules.

DL_POLY is supplied with a wide selection of parametrized force fields. Advanced users may find that the source code is sufficiently flexible for implementation of force fields of their own design (with more complex functional forms). This flexibility is ensured by the generality of its subroutine design and the open access to the source code.

In general, DL_POLY allows two types of force fields: *external* and *molecular*. Examples of the external fields available are shown in Appendix A.

Molecular interactions in DL_POLY are divided conceptually into those that are *intra*-molecular (e.g. *bonded*) and those that are *inter*-molecular (e.g. *non-bonded*). The former type is defined for sets of specific atoms forming molecules or complexes, whereas the latter is defined for sets of specific atomic species. By default, for intramolecular interactions, the electrostatic and (non-bonding) van der Waals interactions are not evaluated. However, optionally this can be overridden in DL_POLY. For intermolecular interactions, the electrostatic interactions are evaluated by default. However, this can be overwritten too.

DL_POLY offers a broad range of empirical potentials in various common forms. The following types of potentials are featured:

(1) *bond* (*intra*) potentials (Appendix B);

(2) *two-body* (*inter*) potentials (Appendix C);

(3) *metal* (*density-dependent*, *intra*) potentials (Appendix C);

(4) *valence angle* (*intra*) potentials (Appendix D);

(5) *three-body* (*inter*) potentials (Appendix D);

(6) *dihedral* and *improper dihedral angle* (*intra*) potentials (Appendix E);

(7) *inversion angle* (*intra*) potentials (Appendix E);

(8) *four-body* (*inter*) potentials (Appendix E).

DL_POLY incorporates a wide choice of techniques for calculating the *long-range electrostatic* interactions (*inter*). The default technique is the SPME method, as commented on in the previous section. However, in other circumstances direct sum methods are suitable. The reaction field method (Allen & Tildesley 2002; Tironi *et al.* 1995) is an inexpensive alternative. Also available in DL_POLY are the direct Coulombic sum method with a distance-dependent dielectric (common in simulations

of biological systems) and the less rigorous truncated and shifted Coulomb sum methods.

DL_POLY also incorporates two position restraining options. It allows atomic sites to be

(i) completely immobilized (i.e. 'frozen' at a fixed point in the MD cell) as this is achieved by setting all forces and velocities associated with that atom to zero during each MD time-step, and

(ii) tethered to a fixed point in space taken as their position at the beginning of the simulation.

The specification, which comes as part of the molecular description, requires a tether potential type similar to the forms (B 1), (B 4), (B 5) in Appendix B.

(*c*) *Boundary conditions*

A choice of boundary conditions is an important feature of any simulation package. Native to the DD concept and offered in DL_POLY are the following boundary conditions: cubic periodic boundaries, orthorhombic periodic boundaries and parallelepiped periodic boundaries (Allen & Tildesley 2002). A further set—truncated octahedral periodic boundaries, rhombic dodecahedral periodic boundaries (Smith & Fincham 1993), slab (x, y, periodic; z, non-periodic) and none (e.g. isolated biopolymer in space)—is only native to the RD concept and therefore offered only in DL_POLY_2, although the last two conditions are also permitted in DL_POLY_3 with a caveat on efficiency (as mentioned in the parallelization subsection).

(*d*) *Solvents*

As with the force field, DL_POLY has no default model solvent. The user defines the required solvent model as another molecular species in the system of interest. In the case of aqueous solutions, most of the commonly used water models are acceptable, provided the electrostatic model does not depend on a point-multipole representation.† While DL_POLY_3 permits only flexible solvent molecules (though rigid water molecules are possible through use of constraint bonds), DL_POLY_2 permits both rigid and flexible solvents. Thus long-established water models such as SPC (Berendsen *et al.* 1981), ST2 (Stillinger & Rahman 1974) and MCY (Matsuoka *et al.* 1976) can be handled. The availability of polarizability and three-body forces within DL_POLY offers a facility to extend beyond these models.

(*e*) *Molecular-dynamics algorithms*

DL_POLY offers a variety of MD simulation algorithms, which differ in terms of their dynamical and thermodynamic properties. All the algorithms in DL_POLY are offered based on two integration schemes, the velocity Verlet (VV) and the leapfrog Verlet (LV) (Allen & Tildesley 2002; Gear 1971; Swope *et al.* 1982), as discussed in § 1.

† However, DL_MULTI, a spin-off of DL_POLY, allows the use of point-multipoles (Leslie 2004).

Different algorithms are required to simulate different molecular models. Flexible molecules and unbonded atoms are treated with the VV algorithm. Molecules defined with bond constraints alone are handled by parallelized versions of the SHAKE (Ryckaert *et al.* 1977) or RATTLE (Anderson 1983) algorithms.† Rigid molecules and rigid units connected by extensible bonds are handled with Fincham's implicit quaternion algorithm (FIQA) (Fincham 1992) (available only in DL_POLY_2). Rigid units connected by rigid bonds are handled by an algorithm (QSHAKE) devised specially for DL_POLY_2 (Forester & Smith 1998).

Algorithms also differ in regard to the thermodynamic *ensemble* they represent. Strictly speaking, not all algorithms represent valid ensembles, but are often labelled by the most closely related ensemble. The following ensembles are available in DL_POLY:

(1) NVE, fixed particle number, volume and energy (the microcanonical ensemble);

(2) NVT, fixed particle number, volume and temperature (the canonical ensemble);

(3) NPT, fixed particle number, pressure and temperature;

(4) N$\underline{\underline{\sigma}}$T, fixed particle number, stress, $\underline{\underline{\sigma}}$, and temperature.

In practice, the ensembles are distinguishable by the use of a thermostat (conserving temperature) and a barostat (conserving pressure). Choosing one or both is what determines the thermodynamic identity of the ensemble. The following integration algorithms are incorporated in DL_POLY:

(1) NVE;

(2) NVT, Berendsen *et al.* (1984); NVT(E_{kin}), Evans & Morriss (1984); NVT, Hoover (1985);

(3) NPT, Berendsen *et al.* (1984); NPT, Hoover (1985); NPT MTK,‡ Martyna *et al.* (1996);

(4) N$\underline{\underline{\sigma}}$T, Berendsen *et al.* (1984); N$\underline{\underline{\sigma}}$T, Hoover (1985); N$\underline{\underline{\sigma}}$T MTK, Martyna *et al.* (1996).

5. Performance

To evaluate the DL_POLY_3 performance, a set of test MD simulations was carried out on the HPC*x* (IBM SP4, http://www.hpcx.ac.uk) supercluster at Daresbury Laboratory. Three model systems were considered, at conditions as shown in table 1: (i) zircon ($Zi_{13\,500}$ $Si_{13\,500}$ $O_{54\,000}$); (ii) radiation damage: high-energy U^{2+} in perovskite (U_1 $Ca_{60\,551}$ $Ti_{60\,552}$ $O_{181\,656}$); and (iii) solid argon ($Ar_{2\,985\,984}$). Simulations were carried out for 10 time-steps on different number of processors (2^N). Results from these simulations are shown in table 2 as values of the real time per simulation time-step given as a function of number of nodes in parallel use.

† Parallelization is either a DD or an RD type. Although SHAKE is part of RATTLE, the full RATTLE is implemented only in the DL_POLY versions based on the VV scheme, whereas SHAKE is used in the DL_POLY versions based on the LV scheme.

‡ The Martyna–Tuckerman–Klein approach to symplectic (time-reversible) algorithms is only available within the DL_POLY versions based on the VV scheme.

Table 1. *The model systems simulated using DL_POLY*

system	size (particles)	ensemble	time-step (ns)	temperature (K)
zircon	81 000	NPT Hoover 0.5 5.0	1×10^{-3}	300
perovskite	302 760	N$\underline{\sigma}$T Hoover 0.5 5.0	5×10^{-5}	300
argon	2 985 984	NPT Berendsen 0.5 5.0	1×10^{-3}	300

Table 2. *DL_POLY_3 scaling performance from simulations of the systems as shown in table 1 on HPCx*

(The time-per-time-step (in seconds, averaged over 10 time-steps) is listed as a function of number of nodes (2^N) used in parallel. Also listed are values of the *link-cell* algorithm (in *italic*) employed in these simulations.)

	nodes					
system	2^0	2^1	2^2	2^3	2^4	2^5
zircon	11.69 s	6.35 s	3.97 s	2.90 s	1.70 s	0.99 s
	12·12·11	*6·12·11*	*6·6·11*	*6·6·5*	*3·6·5*	*3·3·5*
perovskite	N/A	N/A	18.07 s	10.50 s	5.90 s	3.33 s
			7·7·13	*7·7·6*	*3·7·6*	*3·3·6*
argon	N/A	N/A	7.72 s	3.90 s	2.06 s	1.01 s
			43·43·86	*43·43·43*	*21·43·43*	*21·21·43*

	nodes				
	2^6	2^7	2^8	2^9	2^{10}
zircon	0.60 s	0.43 s	0.26 s	0.19 s	N/A
	3·3·2	*1·3·2*	*1·1·2*	*1·1·1*	
perovskite	1.77 s	1.25 s	0.95 s	0.71 s	N/A
	3·3·3	*1·3·3*	*1·1·3*	*1·1·1*	
argon	0.53 s	0.30 s	0.20 s	0.11 s	0.09 s
	21·21·21	*10·21·21*	*10·10·21*	*10·10·10*	*5·10·10*

Also shown in table 2 are the link-cell algorithms ($M_x \cdot M_y \cdot M_z$) employed in the simulations. M_x (analogously for M_y and M_z) is the integer number of the ratio of the width of the system domains in x-direction to the maximal short-range cut-off specified for the system:

$$M_x = \mathrm{nint}\left(\frac{\max(x_{ij})/\#(\mathrm{nodes})_x}{\max(\text{cut-off})}\right). \tag{5.1}$$

Every domain (node) of the MD cell is loaded with

$$(M_x + 1)(M_y + 1)(M_z + 1)$$

linked cells of which $M_x M_y M_z$ belong to that domain and the rest are a halo image of linked cells from neighbouring domains. In this respect, the more linked cells per domain, the less halo data to keep, the more efficient the load distribution per

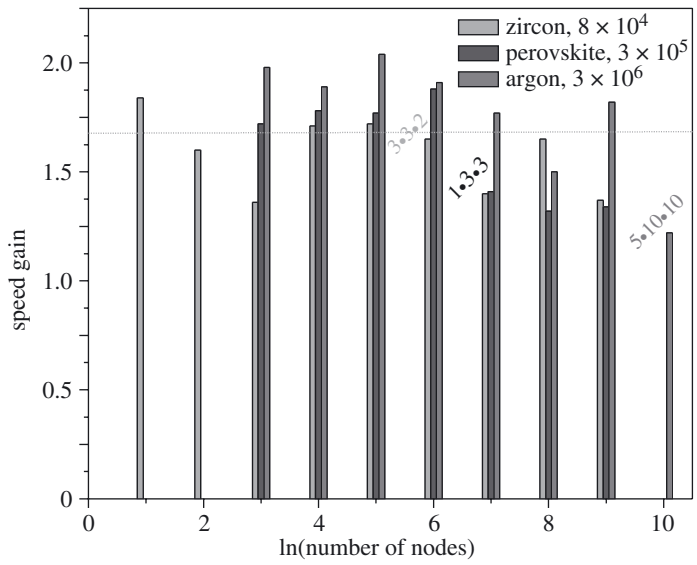

Figure 1. DL_POLY_3 scaling performance from simulations of the systems as shown in table 1 on HPCx. The speed gain function, γ, is plotted as a function of the logarithm of the number of nodes used in a simulation.

node and the less the communications between nodes and therefore, the better the parallelization. This is also clear from figure 1, where the speed gain function, γ†, is plotted as a function of the logarithm of the number of nodes used in a simulation. Figure 1 shows very good parallelization for system (i) up to 32 nodes, system (ii) up to 64 nodes, and system (iii) up to 512 nodes. Drops in the performance ($\gamma < 1.7$ permanently) are clearly exhibited for systems (i) and (ii) after drop of the link-cell algorithm to below $(3 \cdot 3 \cdot 3)$. It is crucial to note that increased parallelization efficiency ($\gamma > 1.2$) remains even when the *link-cell* algorithm is used inefficiently (i.e. there still is a sufficient speed gain in simulations when the number of nodes used in parallel is increased).

Performance is also affected by the fluctuations in the inter-node communication, due to unavoidable communication traffic when a simulation job does not have exclusive use of all machine resources. Such effects strongly worsen the performance when the real time-per-time-step is of the same magnitude as the average communication time (i.e. nodes spend more time communicating rather than calculating), which is specific for the specific simulation control parameters. This explains the sudden drop in performance for system (iii) on 2^{10} nodes.

For purposes of comparison, systems (i) and (ii) were also simulated using DL_POLY_2 on 64 nodes (see table 3). It is clear from table 3 that not only does DL_POLY_3 exhibit better parallelism than DL_POLY_2 but also better optimization of the MD system loading and redistribution onto many nodes and of the inter-node communications. This results in the exceptional over-performance of DL_POLY_3

† The speed gain function is defined (at 2^N) as the ratio of the time-per-time-step when 2^{N-1} nodes are used in parallel to the time-per-time-step when 2^N nodes are used in parallel. $\gamma \geqslant 1.7$ is considered to be very good parallelization. When $\gamma = 2$ parallelization is called 'embarrassing' or perfect.

Table 3. *Comparison of performance between DL_POLY_3 and DL_POLY_2 on 64 nodes*

	performance			
	DL_POLY_3		DL_POLY_2	
system	start-up time (s)	time-per-time-step (s)	start-up time (s)	time-per-time-step (s)
zircon	6	0.60	209	1.40
perovskite	17	1.77	585	5.50
argon	70	0.53	N/A	N/A

over DL_POLY_2 by approximately 30 times shorter start-up times and approximately 3 times shorter time-per-time-step for the considered systems.

It is worth stressing that two effects contribute to that over-performance both due the difference between RD and DD parallelization strategies. The first one (clearly expressed in the start-up times) is that in DL_POLY_3 less time is lost referencing array values since arrays are DD distributed and effectively a number of nodes used in parallel shorter than those referenced in DL_POLY_2. Hence, less time is spent on communicating since in the RD strategy global force array summation is done each time-step. The second one is that the DD implementation of the SPME method incorporating DAFT performs an order of three and more (depending on the system size and number of nodes used in parallel) times better than the RD implementation of the SPME method (Bush *et al.* 2004). Standard Ewald summation was not used in any of the considered systems since it scales more poorly than the SPME method.

6. Availability

DL_POLY is supplied to individuals under a licence and is free of cost to academic scientists pursuing scientific research of a non-commercial nature. A group licence is also available for academic research groups. All recipients of the code must first agree to the terms of the licence. All commercial rights of the package are owned by the Council for the Central Laboratories of the Research Councils (UK) and commercial organizations wishing to obtain and use the code must first obtain an appropriate licensing agreement with CCLRC. Commercial organizations interested in acquiring the package should approach Dr W. Smith (w.smith@dl.ac.uk) at Daresbury Laboratory in the first instance. Daresbury Laboratory is the sole centre for distribution of the package.

A copy of the available academic licences and more information about DL_POLY packages can be found at

http://www.cse.clrc.ac.uk/msi/software/DL_POLY/index.shtml.

This project has been financially supported by grants from the EPSRC, the Computational Science Initiative and the NERC. Much support and encouragement in this project has been provided by the *e*-Science Project and its director Martin Dove. Daresbury Laboratory is thanked for time on various parallel computers. The advice and encouragement from the participants in CCP5 and the EPSRC Materials Consortium is gratefully acknowledged.

Appendix A. External fields available in DL_POLY

1. Electric:

$$\boldsymbol{F}_i = \boldsymbol{F}_i + q_i \boldsymbol{H}. \tag{A 1}$$

2. Oscillating shear:

$$\boldsymbol{F}_x = A\cos(2n\pi z/L_z). \tag{A 2}$$

3. Continuous shear:

$$\boldsymbol{v}_x = \tfrac{1}{2}A|z|/z, \quad |z| > z_0. \tag{A 3}$$

4. Gravitational:

$$\boldsymbol{F}_i = \boldsymbol{F}_i + m_i \boldsymbol{H}. \tag{A 4}$$

5. Magnetic:

$$\boldsymbol{F}_i = \boldsymbol{F}_i + q_i(\boldsymbol{v}_i \wedge \boldsymbol{H}). \tag{A 5}$$

6. Containing sphere:

$$\boldsymbol{F} = A(R_0 - r)^{-n}, \quad r > R_{\mathrm{cut}}. \tag{A 6}$$

7. Repulsive wall:

$$\boldsymbol{F} = A(z_0 - z), \quad z > z_0. \tag{A 7}$$

Appendix B. *Bond* (*intra*) potential forms available in DL_POLY

1. Harmonic bond:

$$U(r_{ij}) = \tfrac{1}{2}k(r_{ij} - r_0)^2. \tag{B 1}$$

2. Morse potential:

$$U(r_{ij}) = E_0[\{1 - \exp(-k(r_{ij} - r_0))\}^2 - 1]. \tag{B 2}$$

3. 12–6 potential:

$$U(r_{ij}) = \left\{\frac{A}{r_{ij}^{12}}\right\} - \left\{\frac{B}{r_{ij}^{6}}\right\}. \tag{B 3}$$

4. Restrained harmonic:

$$\left.\begin{aligned} U(r_{ij}) &= \tfrac{1}{2}k(r_{ij} - r_0)^2, & |r_{ij} - r_0| &\leqslant r_{\mathrm{c}}, \\ U(r_{ij}) &= \tfrac{1}{2}kr_{\mathrm{c}}^2 + kr_{\mathrm{c}}(|r_{ij} - r_0| - r_{\mathrm{c}}), & |r_{ij} - r_0| &> r_{\mathrm{c}}. \end{aligned}\right\} \tag{B 4}$$

5. Quadratic potential:

$$U(r_{ij}) = \tfrac{1}{2}k(r_{ij} - r_0)^2 + \tfrac{1}{3}k'(r_{ij} - r_0)^3 + \tfrac{1}{4}k''(r_{ij} - r_0)^4. \tag{B 5}$$

Appendix C. *Two-body* potential forms available DL_POLY

1. 12–6 potential: with a form the same as equation (B 3).
2. Lennard-Jones:

$$U(r_{ij}) = 4\varepsilon\left\{\left\{\frac{\sigma}{r_{ij}}\right\}^{12} - \left\{\frac{\sigma}{r_{ij}}\right\}^{6}\right\}. \tag{C 1}$$

3. n–m potential:

$$U(r_{ij}) = \frac{E_0}{n-m}\left\{m\left\{\frac{r_0}{r_{ij}}\right\}^{n} - n\left\{\frac{r_0}{r_{ij}}\right\}^{m}\right\}. \tag{C 2}$$

4. Buckingham potential:

$$U(r_{ij}) = A\exp\left(-\frac{r_{ij}}{\rho}\right) - \frac{C}{r_{ij}^6}. \tag{C 3}$$

5. Born–Huggins–Meyer potential:

$$U(r_{ij}) = A\exp(B\{\sigma - r_{ij}\}) - \frac{C}{r_{ij}^6} - \frac{D}{r_{ij}^8}. \tag{C 4}$$

6. Hydrogen-bond (12–10) potential:

$$U(r_{ij}) = \left\{\frac{A}{r_{ij}^{10}}\right\} - \left\{\frac{B}{r_{ij}^{12}}\right\}. \tag{C 5}$$

7. Shifted force n–m potential (Clarke *et al.* 1986):

$$\begin{aligned} U(r_{ij}) = {} & \frac{\alpha E_0}{n-m}\left[m\beta^n\left\{\left(\frac{r_0}{r_{ij}}\right)^n - \left(\frac{1}{\gamma}\right)^n\right\} - n\beta^m\left\{\left(\frac{r_0}{r_{ij}}\right)^m - \left(\frac{1}{\gamma}\right)^m\right\}\right] \\ & + \frac{nm\alpha E_0}{n-m}\left(\frac{r_{ij}-\gamma r_0}{\gamma r_0}\right)\left\{\left(\frac{\beta}{\gamma}\right)^n - \left(\frac{\beta}{\gamma}\right)^m\right\} \end{aligned} \tag{C 6}$$

with

$$\gamma = \frac{r_{\mathrm{cut}}}{r_0}, \qquad \beta = \gamma\left(\frac{\gamma^{m+1}-1}{\gamma^{n+1}-1}\right)^{1/(n-m)}$$

and

$$\alpha = \frac{m-n}{n\beta^m(1+(m/\gamma-m-1)/\gamma^m) - m\beta^n(1+(n/\gamma-n-1)/\gamma^n)}.$$

8. Morse potential: with a form the same as equation (B 2).
9. Tabulated potentials are also an option in DL_POLY†.

For simulation of metals and metal-like alloys, DL_POLY offers a *metal* (*density-dependent*) potential in the Sutton and Chen form (Finnis & Sinclair 1984; Sutton & Chen 1990):

$$U_{\mathrm{SC}} = \varepsilon\left\{\sum_j\sum_{i<j}\left(\frac{a}{r_{ij}}\right)^n - C\sum_i \rho_i^{1/2}\right\}, \tag{C 7}$$

† DL_POLY Reference Manual available at http://www.cse.clrc.ac.uk/msi/software/DL_POLY/index.shtml.

where (*local density*)

$$\rho_i = \sum_i \left(\frac{a}{r_{ij}}\right)^m.$$

Appendix D. *Valence angle* potential forms available DL_POLY

1. Harmonic:

$$U(\theta_{ijk}) = \tfrac{1}{2}k(\theta_{ijk} - \theta_0)^2. \tag{D 1}$$

2. Quadratic:

$$U(\theta_{ijk}) = \tfrac{1}{2}k(\theta_{ijk} - \theta_0)^2 + \tfrac{1}{3}k'(\theta_{ijk} - \theta_0)^3 + \tfrac{1}{4}k''(\theta_{ijk} - \theta_0)^4. \tag{D 2}$$

3. Truncated harmonic:

$$U(\theta_{ijk}) = \tfrac{1}{2}k(\theta_{ijk} - \theta_0)^2 \exp[-(r_{ij}^8 + r_{jk}^8)/\rho^8]. \tag{D 3}$$

4. Screened harmonic:

$$U(\theta_{ijk}) = \tfrac{1}{2}k(\theta_{ijk} - \theta_0)^2 \exp[-(r_{ij}/\rho_1 + r_{jk}/\rho_2)]. \tag{D 4}$$

5. Screened Vessal (Vessal *et al.* 1990):

$$U(\theta_{ijk}) = \frac{k}{8(\theta_{ijk} - \theta_0)^2}\{[(\theta_0 - \pi)^2 - (\theta_{ijk} - \pi)^2]^2\} \exp[-(r_{ij}/\rho_1 + r_{jk}/\rho_2)]. \tag{D 5}$$

6. Truncated Vessal (Vessal 1994):

$$\begin{aligned} U(\theta_{ijk}) = k[&\theta_{ijk}^a(\theta_{ijk} - \theta_0)^2(\theta_{ijk} + \theta_0 - 2\pi)^2 \\ &- \tfrac{1}{2}a\pi^{a-1}(\theta_{ijk} - \theta_0)^2(\pi - \theta_0)^3] \exp[-(r_{ij}^8 + r_{jk}^8)/\rho^8]. \end{aligned} \tag{D 6}$$

7. Harmonic cosine:

$$U(\theta_{ijk}) = \tfrac{1}{2}k(\cos(\theta_{ijk}) - \cos(\theta_0))^2. \tag{D 7}$$

8. Cosine:

$$U(\theta_{ijk}) = A[1 + \cos(m\theta_{ijk} - \delta)]. \tag{D 8}$$

Three-body potentials in DL_POLY are available in forms that include those from equations (D 3)–(D 6) as well as the hydrogen-bond potential form (Mayo *et al.* 1990):

$$U(\theta_{ijk}) = D_{\mathrm{hb}} \cos^4(\theta_{ijk})[5(R_{\mathrm{hb}}/r_{jk})^5 - 6(R_{\mathrm{hb}}/r_{jk})^{10}]. \tag{D 9}$$

Appendix E.

DL_POLY offers the following choice of *dihedral* (and *improper dihedral*) *angle* potential forms.

1. Cosine:

$$U(\phi_{ijkl}) = A[1 + \cos(m\phi_{ijkl} - \delta)]. \tag{E 1}$$

2. Harmonic:

$$U(\phi_{ijkl}) = \tfrac{1}{2}k(\phi_{ijkl} - \phi_0)^2. \tag{E 2}$$

3. Harmonic cosine:

$$U(\phi_{ijkl}) = \tfrac{1}{2}k(\cos(\phi_{ijkl}) - \cos(\phi_0))^2. \tag{E 3}$$

4. Triple cosine (standard) and triple cosine plus constant term:

$$U(\phi_{ijkl}) = \tfrac{1}{2}A_1(1 + \cos(\phi_{ijkl})) + \tfrac{1}{2}A_2(1 + \cos(2\phi_{ijkl})) + \tfrac{1}{2}A_3(1 + \cos(3\phi_{ijkl})); \tag{E 4}$$

$$U(\phi_{ijkl}) = A_0 + \tfrac{1}{2}A_1(1 + \cos(\phi_{ijkl})) + \tfrac{1}{2}A_2(1 + \cos(2\phi_{ijkl})) + \tfrac{1}{2}A_3(1 + \cos(3\phi_{ijkl})). \tag{E 5}$$

5. Ryckeart–Bellemans (Ryckaert & Bellemans 1978) and fluorinated Ryckeart–Bellemans (Schmidt *et al.* 1996) potential forms.

Inversion angle potentials and *four-body* potentials in DL_POLY are available in (E 1)–(E 3) forms.

References

Allen, M. P. & Tildesley, D. J. 2002 *Computer simulation of liquids*. Oxford: Clarendon.

Anderson, H. C. 1983 *J. Computat. Phys.* **52**, 24.

Bush, I. J., Todorov, I. T. & Smith, W. 2004 A DAFT DL_POLY distributed memory adaptation of the smoothed particle mesh Ewald method. (In preparation.)

Berendsen, H. J. C., Postma, J. P. M., van Gunsteren, W. & Hermans, J. 1981 *Intermolecular forces* (ed. B. Pullman), p. 311 (SPC Water Standard Reference). Dordrecht: Reidel.

Berendsen, H. J. C., Postma, J. P. M., van Gunsteren, W., Dinola, A. & Haak, J. R. 1984 *J. Chem. Phys.* **81**, 3684.

Brigham, E. O. 1988 *The fast Fourier transform and its applications*. Englewood Cliffs, NJ: Prentice Hall.

Bush, I. J. 1999 The Daresbury Advanced Fourier Transform. Daresbury Laboratory.

Clarke, J. H. R., Smith, W. & Woodcock, L. V. 1986 *J. Chem. Phys.* **84**, 2290.

Essmann, U., Perera, L., Berkowtz, M. L., Darden, T., Lee, H. & Pedersen, L. G. 1995 *J. Chem. Phys.* **103**, 8577.

Evans, D. J. & Morriss, G. P. 1984 *Comput. Phys. Rep.* **1**, 297.

Fincham, D. 1992 *Mol. Simulat.* **8**, 165.

Fincham, D. & Mitchell, P. J. 1993 *J. Phys. Condens. Matter* **5**, 1031.

Finnis, M. W. & Sinclair, J. E. 1984 *Phil. Mag.* A **50**, 45.

Forester, T. R. & Smith, W. 1998 *J. Computat. Chem.* **19**(1), 102.

Gear, C. W. 1971 *Numerical initial value problems in ordinary differential equations*. Englewood Cliffs, NJ: Prentice Hall.

Hockney, R. W. & Eastwood, J. W. 1981 *Computer simulation using particles*. McGraw-Hill.

Hoover, W. G. 1985 *Phys. Rev.* A **31**, 1695.

Leslie, M. 2004 *Mol. Phys.* (In the press.)

Martyna, G. J., Tuckerman, M. E., Douglas, J. T. & Klein, M. L. 1996 *Mol. Phys.* **87**(5), 1117.

Matsuoka, O., Clementi, E. & Yoshimine, M. 1976 *J. Chem. Phys.* **64**, 1351.

Mayo, S., Olafson, B. & Goddard, W. 1990 *J. Phys. Chem.* **94**, 8897.

Pinches, M. R. S., Tildesley, D. & Smith, W. 1991 *Mol. Simulat.* **6**, 51.

Rapaport, D. 1991 *Comput. Phys. Commun.* **62**, 217.
Ryckaert, J. P. & Bellemans, A. 1978 *Faraday Disc.* **66**, 95.
Ryckaert, J. P., Ciccotti, G. & Berendsen, H. J. C. 1977 *J. Computat. Phys.* **23**, 327.
Schmidt, M. E., Shin, S. & Rice, S. A. 1996 *J. Phys. Chem.* **104**, 2101.
Smith, W. 1987 *Mol. Graph.* **5**, 71.
Smith, W. 1991 *Comput. Phys. Commun.* **62**, 229.
Smith, W. & Fincham, D. 1993 *Mol. Simulat.* **10**, 67.
Smith, W. & Forester, T. R. 1994*a* *Comput. Phys. Commun.* **79**, 52.
Smith, W. & Forester, T. R. 1994*b* *Comput. Phys. Commun.* **79**, 63.
Smith, W. & Forester, T. R. 1996 *J. Mol. Graph.* **14**, 136.
Smith, W., Young, C. W. & Rodger, P. M. 2002 *Mol. Simulat.* **28**(5), 385.
Stillinger, F. H. & Rahman, A. 1974 *J. Chem. Phys.* **60**, 1545.
Sutton, A. P. & Chen, J. 1990 *Phil. Mag. Lett.* **61**, 139.
Swope, W. C., Andersen, H. C., Berens, P. H. & Wilson, K. R. 1982 *J. Chem. Phys.* **76**, 63.
Tironi, I. G., Sperb, R., Smith, P. E. & van Gunsteren, W. F. 1995 *J. Phys. Chem.* **102**, 5451.
Vessal, B. 1994 *J. Non-cryst. Solids* **177**, 103.
Vessal, B., Amini, M., Leslie, M. & Catlow, C. R. A. 1990 *Mol. Simulat.* **5**, 1.