DL_MESO_DPD: development and use of mesoscale modelling software

Michael A. Seaton

Published version information

Citation: MA Seaton. DL_MESO_DPD: development and use of mesoscale modelling software. Mol Simul 47, no. 2-3 (2018): 228-247
DOI: 10.1080/08927022.2018.1524143

This is an Accepted Manuscript of an article published by Taylor & Francis in Molecular Simulation on 20th September 2018, available online: http://www.tandfonline.com/10.1080/08927022.2018.1524143.
This version is made available in accordance with publisher policies. Please cite only the published version using the reference above. This is the citation assigned by the publisher at the time of issuing the AAM/APV. Please check the publisher's website for any updates.

Molecular Simulation
Vol. 47, No. 2-3, May 2021, 228–247

ARTICLE

DL_MESO_DPD: development and use of mesoscale modelling software

Michael A. Seaton[a]*

[a]*STFC Daresbury Laboratory, Sci-Tech Daresbury, Keckwick Lane, Daresbury, Warrington, WA4 4AD, UK*

(*Received 19 March 2018, Accepted 11 September 2018, Published online: 20 September 2018*)

DL_MESO is a highly-scalable general purpose software package for mesoscale modelling. Created and developed at Daresbury Laboratory for the UK Collaborative Computational Project CCP5, it was intended to be a companion package to the flagship molecular dynamics code DL_POLY. One of DL_MESO's component codes, DL_MESO_DPD, is based on dissipative particle dynamics, a mesoscale modelling technique with many similarities to classical molecular dynamics. While this code and DL_POLY were created with different applications in mind, they share a significant amount of functionality and development history. This article gives an overview on how DL_MESO_DPD has been developed, including its shared history with DL_POLYand information on its current performance, and a selection of applications for which the code has been used.

Keywords: DL_MESO; mesoscale modelling; dissipative particle dynamics; software development; DL_POLY

1. Introduction

Mesoscale modelling methods fit into a gap between atomistic and continuum methods of modelling materials, addressing intermediate length and timescales (10–1000 nm, 1 ns–10 ms) by including some vestige of atomistic detail to capture both essential microscopic physics and large-scale structural effects. As well as being of interest to the academic community, mesoscopic simulation methods are of great interest to industry and many of their key developments have originated from industrial research. One notable example is Dissipative Particle Dynamics (DPD), which was conceived at Shell [1, 2] and has subsequently been developed further by researchers from Unilever [3–5]. Many mesoscale modelling techniques are highly parallelisable, opening up the possibility of using high-performance computing (HPC) facilities to obtain length and time scales approaching the continuum.

The DL_MESO project [6] (www.ccp5.ac.uk/DL_MESO) originated as part of the United Kingdom Collaborative Computational Project for the computer simulation of condensed phases, known as CCP5 (www.ccp5.ac.uk). It was conceived in 2003 as a complementary package to CCP5's flagship classical molecular dynamics (MD) code, DL_POLY [7–10], with the aim of creating a comprehensive mesoscale modelling package that could exploit emerging parallel computing architectures as well as satisfy demands for verifiable and extendable 'open' software. DL_MESO currently consists of programs for two mesoscale modelling methods: Lattice Boltzmann Equation (LBE) methods [11–13] (DL_MESO_LBE) and DPD (DL_MESO_DPD). The programs are supplied as source code and are designed to require only `C++` and `Fortran` compilers for the LBE and DPD codes respectively, as well as message passing interface (MPI) libraries to control parallel communication on memory distributed parallel computers: this makes them highly portable and able to run on any available

*Corresponding author. Email: michael.seaton@stfc.ac.uk

operating system.

The DPD part of the DL_MESO package, DL_MESO_DPD, has a lot in common with DL_POLY. Both programs and their included methodologies (DPD and MD) depend upon calculating interaction forces acting on particles and integrating those forces over time to determine their motion. From the user's point of view, both programs use similar input file formats which are – to at least some degree – interchangeable and compatible. DL_MESO_DPD and DL_POLY_4 also share the same parallelisation strategy of domain decomposition with link-cell lists [14] to split calculations evenly among available processor cores and efficiently find interacting pairs of particles. It should be noted that while it is possible to modify DL_POLY to carry out DPD calculations [15], DL_MESO_DPD was conceived as a separate code with some significant differences in design philosophy and application. Nevertheless, the development of DL_MESO_DPD has never been in isolation to that of DL_POLY, and both codes can be argued to have influenced the other in various ways.

Since its first release, DL_MESO_DPD has been thought of as a general-purpose DPD 'simulation engine' – a feature-rich code designed primarily to carry out a particular form of computational modelling – in a similar manner to DL_POLY for molecular dynamics. As such, it has been applied to a wide range of soft matter problems acting at the mesoscale, including e.g. loading and release of drugs in amphiphilic vesicles [16], self-assembly of liquid crystals [17] and protonation effects in polyelectrolyte membranes [18].

This article will reflect on how DL_MESO_DPD came into being and how it has been developed, including its common history with DL_POLY. Starting with a brief outline of DPD to give context to the required calculations, the development approach for DL_MESO_DPD will be described. Features that are common to both codes, including more recent additions to DL_MESO_DPD, will be highlighted, as well as recent performance information on HPC platforms and a brief overview of applications that have made use of the code over the past few years.

2. Dissipative Particle Dynamics

DPD is a stochastic simulation technique originally devised by Hoogerbrugge and Koelman [1] in 1992 to simulate the dynamic and rheological properties of simple and complex fluids at time and space scales larger than those available with classical molecular dynamics. Español and Warren later demonstrated that the major part of the method could be formulated as a pairwise thermostat which automatically conserves momentum [19], ensuring Galilean invariance and correct hydrodynamic behaviour. This thermostat is frequently combined with soft repulsive potentials [3] between particles that can either be considered as coarse-grains of molecular segments (e.g. functional groups) or whole molecules, or as a mesoscopic representation of a continuous fluid.

A DPD system is typically described by a collection of particles (often denoted as 'beads'), each of which has an associated mass m_i, position $\vec{r}_i$ and velocity $\vec{v}_i$. We can assume reduced units with length, mass and energy scales (the latter usually given as the product of the Boltzmann constant and system temperature, k_BT) set equal to one, which can be mapped onto real units if the level of coarse-graining is known. The behaviour of these beads can be controlled predominately by pairwise forces exerted on particle i by particle j over a timestep Δt. Bead motion can be determined by integrating forces over the timestep: the most frequently used form of integration for DPD simulations is based on Velocity Verlet [20], with variations available to apply the pairwise thermostat more accurately [21, 22].

Strictly speaking, DPD refers to two particular pairwise forces: a dissipative force ($\vec{F}_{ij}^{D}$) and a random force ($\vec{F}_{ij}^{R}$). Their definitions are as follows:

$$\vec{F}_{ij}^{D} = -\gamma_{ij}\omega^{D}(r_{ij})\,(\hat{e}_{ij}\cdot\vec{v}_{ij})\,\hat{e}_{ij}, \quad (1)$$

$$\vec{F}_{ij}^{R} = \sigma_{ij}\omega^{R}(r_{ij})\xi_{ij}\,(\Delta t)^{-\frac{1}{2}}\,\hat{e}_{ij}, \quad (2)$$

where $\vec{r}_{ij} = \vec{r}_j - \vec{r}_i$, $r_{ij} = |\vec{r}_{ij}|$, $\hat{e}_{ij} = \vec{r}_{ij}/r_{ij}$ (unit vector between particles i and j), $\vec{v}_{ij} = \vec{v}_j - \vec{v}_i$ (relative velocity between particles i and j) and $\xi_{ij} = \xi_{ji}$ is a Gaussian random number with zero mean and variance of one. The weight functions for dissipative and random forces, $\omega^D(r_{ij})$ and $\omega^R(r_{ij})$ respectively, are normally chosen to disappear to zero beyond a cutoff distance r_c. The two weight functions and the dissipative and random force parameters, γ_{ij} and σ_{ij} respectively, are carefully chosen to obey a fluctuation-dissipation theorem, producing an equilibrium temperature and acting as a momentum-conserving thermostat within the canonical ensemble:

$$\sigma_{ij}^2 = 2k_B T \gamma_{ij}, \tag{3}$$

$$\omega^D(r_{ij}) = \left[\omega^R(r_{ij})\right]^2. \tag{4}$$

The most frequent form of random force weight function used in DPD simulations is $\omega^R(r_{ij}) = 1 - \frac{r_{ij}}{r_c}$ for $r_{ij} < r_c$ and $\omega^R(r_{ij}) = 0$ for $r_{ij} \geq r_c$. This often matches the pairwise force frequently used for conservative interactions (see below) and thus reduces the required number of computations per particle pair. While DPD is the most commonly-used pairwise thermostat for mesoscopic particle simulations, alternative pairwise schemes also exist [23–25] and can be applied alongside interaction forces.

Conservative interactions between particles in DPD simulations – including van der Waals-like pairwise conservative interactions – can be chosen to take any given form, including those typically used in classical molecular dynamics [26]. The vast majority of DPD simulations typically use soft repulsive Groot-Warren interactions [3] with the following pairwise force:

$$\vec{F}_{ij}^C = \begin{cases} A_{ij}\left(1 - \frac{r_{ij}}{r_c}\right)\hat{e}_{ij} & r_{ij} < r_c \\ 0 & r_{ij} \geq r_c \end{cases}, \tag{5}$$

where A_{ij} is the conservative force magnitude. This form of conservative interaction force gives a quadratic potential with a finite value at zero separation, as well as a quadratic equation of state for a fluid made up of these particles. Not only does this particular interaction allow DPD to use larger timesteps and access longer timescales than MD, but its parameters can also be mapped onto observable properties such as isothermal fluid compressibilities, the Flory-Huggins χ-parameter for polymeric systems [3], infinite dilution activity coefficients [27], or partition coefficients [28].

An extension to these 'standard' DPD interactions is many-body DPD (MDPD) [29, 30], which defines the conservative force as a function of localised densities for both beads in a pair, i.e.

$$\vec{F}_{ij}^C = \begin{cases} -\left(\psi'_{ex}(\rho_i) + \psi'_{ex}(\rho_j)\right)\omega^d(r_{ij}) & r_{ij} < r_d \\ 0 & r_{ij} \geq r_d \end{cases}, \tag{6}$$

where r_d is a cutoff distance for many-body DPD interactions, ρ_i is the localised density for particle i, ψ'_{ex} is the derivative of the selected excess free energy with respect to density and $\omega^d(r_{ij})$ is a many-body force weight function. The localised densities in MDPD are calculated as sums of a weight function between a given particle and its neighbours within the cutoff distance r_d: the derivative of this weight function with distance gives the many-body weight function $\omega^d(r_{ij})$. While MDPD requires an additional pass over all interacting pairs of particles for each timestep, it can more accurately represent the thermodynamics of a real fluid system and allow for multiple phases. A simple MDPD scheme that includes an additional density-dependent term to the above Groot-Warren conservative force [4] has been shown to give a van der Waals-like cubic equation of state and vapour-liquid coexistence: it has been parameterised for water to give correct phase densities and surface tension [31, 32].

Other interactions between particles can include bond stretching, bending and torsional interac-

tions between pairs, triples and quadruples of beads, which can be used to create representations of larger molecules out of DPD beads. By bonding together beads with different conservative force magnitudes, it is possible to model systems of amphiphilic molecules in solution and study how their morphologies affect the structures they form, such as micelles, hexagonal or lamellar phases [33], vesicles [34] and bilayers [35].

Charge-based electrostatic interactions can also be considered in DPD simulations. Due to the soft repulsive potentials typically used in conservative forces, point charges are normally eschewed to avoid the possibility of ion collapse (although higher values for conservative force parameters can also help [36]) and various forms of smearing function to apply the charge over a finite volume have been used [5]. Two approaches have been devised to determine long-range electrostatic interactions between charged DPD beads. The first approach is based on assigning smeared charges to a grid and iteratively solving the Poisson equation [37]:

$$\nabla \cdot (p(\vec{r})\nabla\psi) = -\Gamma\rho(\vec{r}) \tag{7}$$

where $p(\vec{r})$ is the relative permittivity at position $\vec{r}$, ψ is the electrostatic field, ρ is the charge density (ion concentration) and $\Gamma = \frac{e^2}{k_B T r_c \epsilon_0 \epsilon_r}$ is the permittivity coefficient for the background medium. The second approach is based on Ewald summation with small modifications to the real-space part of the potential to accommodate charge smearing [38]:

$$U_{ij}^{e,sr} = \frac{\Gamma q_i q_j}{4\pi r_{ij}} \left[\mathrm{erfc}(\alpha r_{ij}) - f(r_{ij})\right] \tag{8}$$

where q_i is the charge on particle i, α is the Ewald sum smearing coefficient and $f(r_{ij})$ is the selected charge smearing function (equal to zero for point charges). Recent developments involving charge interactions in DPD include the creation of mesoscopic models for water with polarisation effects [39, 40].

A fuller description and review of Dissipative Particle Dynamics, including its conceptualisation from microscopic and macroscopic techniques and outstanding challenges in parameterisation, can be found in the literature [41].

3. Development history of DL_MESO_DPD

CCP5 started the DL_MESO project in 2003 with the intention of creating a general-purpose software package for mesoscale modelling techniques along similar lines to its flagship molecular dynamics code DL_POLY. Each mesoscale modelling method was intended to be supplied by its own code, designed to run both on standalone computers and on large high performance computing architectures (clusters, supercomputers etc.).

Initial development of DL_MESO was funded as a three-year CCP5 flagship project, and its first release in November 2004 (version 1.0) consisted of a single code written in `C++`, DL_MESO_LBE, which employed the Lattice Boltzmann Equation method. A second release of this code with minor alterations was released a month later. The intention always existed, however, to supply additional mesoscale modelling methods to the package, and DPD was identified as the next most viable candidate (after LBE) based on ease of development and likelihood of take-up by academic and industrial users. Due to the similarities of molecular dynamics and DPD, a code for the latter modelling method was achieved by modifying a pre-existing MD code to include calculations of the additional dissipative and random forces – Equations (1) and (2) – required for the DPD thermostat.

The resulting DPD code, DL_MESO_DPD, has been developed extensively since its original creation in 2006, with new features added to each successive version, and optimised for improved

Table 1.: Versions of DL_MESO with DPD codes: release dates and periods, number of code lines in DL_MESO_DPD and numbers of registered users (to date)

Version	Release date	Release period (months)	Code lines	Users
2.2	November 2006	42	5179	270
2.3	April 2010	13	9266	134
2.4	June 2011	10	17843	113
2.5	April 2012	43	25269	410
2.6	November 2015	26[a]	36145	283[a]
2.7	Spring 2018 (TBC)	-	53947[a]	-

[a]Correct at time of writing (February 2018).

computational performance (both per core and in parallel scalability) on a variety of computing platforms. Table 1 gives a summarised list of the various versions of DL_MESO that have been released with DPD codes, including the number of lines in those codes and the number of academic users that have registered for each version of the software. Some details of the forthcoming release of DL_MESO, version 2.7, have also been included.

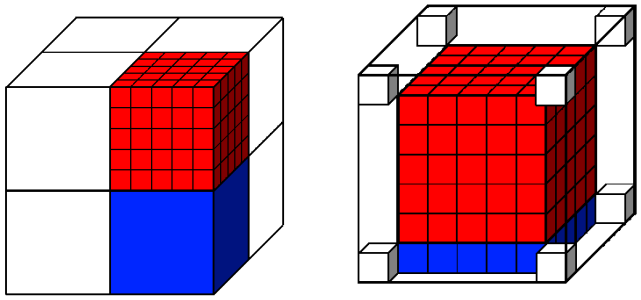
3.1. *Origins*

The base code for DL_MESO_DPD is MDMEGA [42], a molecular dynamics code in the CCP5 Program Library written in 1992 – notably the same year as the first version of DL_POLY [43] – designed to model large numbers of single atoms using Lennard-Jones potentials in a microcanonical (NVE) ensemble with leapfrog Verlet force integration. Initially written for the Intel IPSC/860 parallel computer in `FORTRAN77` with custom parallel communication libraries, it employs domain decomposition with link cells as its parallelisation strategy [14]. As illustrated in Figure 1, the entire simulation cell is divided up equally among the available processor cores into domains and each core holds all of the particles in that particular domain. The domain is then divided further into equal-sized link cells with widths of no less than the interaction cutoff distance r_c: lists of particles in each link cell are created and these lists are used to find all particle pairs within the cutoff distance (noting that no interacting pair of particles can exist beyond the nearest neighbours of a given link cell).

To find all interacting pairs of particles throughout the domain, additional particle data are copied into boundary halos around the domain from neighbouring processor cores in an 'export' communication step to form an additional layer of link cells. While searches between link cells are limited within the domain itself to ensure each particle pair is only dealt with once, link cells at the edge of the domain additionally search in those in the boundary halo [44]. These allow the total net forces to be calculated for each particle in a given domain (excluding those in boundary halos), which can then be integrated to give the new particle positions. An additional 'deport' communication step can then be used to move particles to neighbouring processor cores if they have left the current domain.

Some limitations exist in MDMEGA that restrict its use for general-purpose simulation problems:

- Processor cores are assumed to be a hypercube and thus limited to a power of 2 (up to 128 for the Intel IPSC/860).
- The simulation cell must be a cube, as must the total number of particles modelled (initialised as a face-centred cube).



(a) (b)

Figure 1.: Link cell and domain decomposition within DL_MESO. (a) The simulation cell is divided into geometric domains – two of them being highlighted in blue and red – which are then divided into link cells (only shown for the red domain). (b) The halo data around a given domain (red) are composed of the link cells from the edges of all neighbouring domains (one of them is highlighted in blue).

- No bond or charge interactions exist between pairs of particles.
- The total number of particles per processor core is limited to a set number given at compile-time. (`FORTRAN77` does not allow runtime allocation of arrays.)
- Only one type of particle can be modelled.
- Outputs from MDMEGA are limited to statistical properties (energies, pressures, temperatures etc.), and system configurations and statistical accumulators intended solely to restart simulations.

Nevertheless, this code is still a good starting point for a general-purpose MD or DPD code that can model many thousands or millions of particles: the use of domain decomposition overcomes the limitation of a replicated data code (e.g. DL_POLY_2 [7]) that requires every processor core to hold data for all particles in the simulation cell. Not only was it the base code for DL_MESO_DPD, but it also influenced the creation of DL_POLY_3 [9], the first domain decomposed version of DL_POLY.

The first DPD code derived from MDMEGA was the unreleased DL_DPD code, which was created in October 2001 (also written in `FORTRAN77`) and predates the DL_MESO project by nearly two years. This code removes a number of MDMEGA's restrictions: up to four particle species could be modelled, using conservative interactions between particle pairs that were either 'standard' Groot-Warren DPD (Equation (5)), Lennard-Jones or Weeks-Chandler-Andersen [45] potentials. Parameters for pairs of each species type could be supplied by the user in the required input file (`CONTROL`), while interactions between different species types were determined using the Lorentz-Berthelot mixing rule [46] ($A_{ij} = \sqrt{A_{ii}A_{jj}}$, $r_{c,ij} = \frac{1}{2}\left(r_{c,ii} + r_{c,jj}\right)$). The integration scheme was also modified from leapfrog Verlet to Velocity Verlet, as were the pairwise forces to include the DPD dissipative and random forces. To implement calculations of random forces as efficiently as possible, an approximation to Gaussian random numbers was used based on the central limit theorem with uniform random numbers (u) between zero and one [47]: $\xi \approx \sqrt{12}\left(u - \frac{1}{2}\right)$. This has been shown to give statistically indistinguishable results from a true Gaussian random number generator and is thus adequate for DPD simulations [3]. To improve portability, the Intel-specific parallel communication libraries were replaced with Message Passing Interface (MPI) libraries, which had since become a widely-supported portable standard.

A number of consequences resulted in making the above modifications. One of these was inclusion of velocities and particle species (as well as positions) in 'export' communications to correctly calculate dissipative forces. An additional 'import' communication step was also required to send

force contributions for particles in boundary halos back to their respective processor cores: this was needed to overcome a difficulty in synchronising random number generators between processor cores to produce the same (pseudorandom) number and resulting random force for a given pair of particles.

As the original project funding for DL_MESO came to an end in 2006, DL_DPD was modified to create the DPD component of DL_MESO, converting the `FORTRAN77` code to `C++` in keeping with DL_MESO_LBE. Two versions of the code in `C++` were also produced: one for parallel calculations and a modified version for serial calculations on a single processor core that did not require MPI libraries to compile. While the first two versions of DL_MESO with DPD codes (2.0 and 2.1) were not formally released, the second revision (version 2.2) was released in November 2006 and coincided with automatic online registration for academic users to download the software and the INFOMAIL mailing list [48] to send announcements about bug fixes and relevant software training events.

Further development of the converted `C++` serial version of DL_DPD took place at Daresbury Laboratory, including the addition of bonded interactions, but none of these changes made it into the release version of DL_MESO. No minor revisions to fix any bugs in DL_MESO_DPD were made, nor were any further versions of DL_MESO released until April 2010.

From 2009, CCP5 funded development of DL_MESO as part of the Service Level Agreement (SLA) between STFC Daresbury Laboratory and the Engineering and Physical Science Research Council (EPSRC), now named the Computational Science Centre for Research Communities (CoSeC) [49]. A major part of this work involved reviving DL_MESO_DPD and restarting its development, in particular making it a more general-purpose DPD 'simulation engine'. This required adding functionality to DL_MESO_DPD to expand the range of DPD-based systems that could be modelled.

In the process of this 'revival' (starting with version 2.3 of DL_MESO), the language of DL_MESO_DPD was switched back to `Fortran`, albeit to a more modern form than before. DL_POLY_2 and DL_POLY_3, also written in `Fortran`, had been developed significantly since the release of version 2.2 of DL_MESO and contained features and functionalities that were useful for DPD simulations (e.g. bonded interactions, Ewald summation for charged systems) and could be adopted by DL_MESO_DPD. Since the conversion of DL_DPD to `C++` did not take any real advantage of the language's features, it made more sense to start developing the `Fortran` version of DL_DPD directly rather than converting DL_POLY's various features into `C++`.

Another change to the code in DL_MESO_DPD was the use of modules to group together subroutines with similar purposes and functionalities (e.g. communications, pairwise force calculations). This allows common subroutines for the serial and parallel versions of the code to be shared between them, reducing the amount of code to maintain. This is in contrast to both the original version of DL_DPD, which had separate files for the various subroutines, and single files for the serial and parallel versions of the `C++` version of DL_MESO_DPD. Some of the code's original limitations, mainly the numbers of processor cores and unbonded particles, were also relaxed at this stage.

3.2. *Adding functionality*

The functionalities of DL_MESO_DPD have expanded considerably since its first formal release to allow ever-larger ranges of DPD-based systems to be modelled. From version 2.3 onwards, users have been able to specify conservative interactions between pairs of different particle species directly rather than solely relying on mixing rules. Many-body DPD (MDPD) interactions were added to version 2.4, which include calculations of localised densities (carried out in a similar fashion to normal pairwise forces with linked-cell lists) required for every possible form of MDPD interaction and sample routines to calculate the two-parameter form of MDPD [4][1]. The user has also been able to specify different conservative interaction types ('standard' Groot-Warren DPD, MDPD, Lennard-Jones and Weeks-Chandler-Andersen) and different dissipative force parameters for each

[1]Advanced users can modify the relevant force and potential calculation routines to use their own MDPD models.

pair of particle species from version 2.5 of DL_MESO onwards.

Bonded interactions – stretching between particle pairs, angles at junctions between pairs of bonds and dihedrals between two planes defined using the locations of four particles – were introduced in version 2.3 and have vastly expanded the range of simulations DL_MESO_DPD can carry out. Lists of particles involved in bonds, angles and dihedrals are constructed using their unique global indices, which can be searched for among the available particles in each processor core's domain and boundary halo. The user can also expand the size of the boundary halo to allow for bonds that stretch beyond the standard DPD cutoff distance (r_c); the user can (from version 2.4 onwards) alternatively invoke a replicated data approach for finding particles in bonded pairs that might be far apart. The latter strategy is certainly more costly in terms of performance and parallel scalability but can be used to ease system equilibration. The deport communication step has also been modified to transfer bond, angle and dihedral data along with associated key beads: for the default non-replicated data strategy, this ensures each bond, angle or dihedral is only calculated once per timestep.

Charge-based (electrostatic) interactions have also been added to DL_MESO_DPD. Ewald summation with both real-space and reciprocal-space components has been included in DL_MESO from version 2.4 onwards. The real-space part was implemented using link cells to find pairs of charged particles, albeit using a larger real-space cutoff distance than for other conservative interactions. The long-range reciprocal-space part has been based on analytical calculation of Fourier transforms [50]. While this reciprocal-space implementation is recognised as not being a particularly efficient algorithm, scaling at best $N^{\frac{3}{2}}$ with the number of particles N, its inclusion in DL_MESO_DPD has nevertheless opened up the possibility of modelling systems with e.g. polyelectrolytes and charged surfactants. Version 2.7 of DL_MESO will also include the computationally more efficient Smooth Particle Mesh Ewald (SPME) method [51] for the reciprocal-space section; this makes use of Fast Fourier Transform (FFT) routines on a discrete charge grid constructed from interpolated charges and typically scales $N \log N$.

Charge smearing schemes to avoid ion collapse are applied in real-space only. An approximate (truncated) form of Slater-type smearing [38] was initially used in versions 2.4 and 2.5, i.e. the smearing function in Equation (8) is given as

$$f(r_{ij}) = (1 + \beta r_{ij}) \exp(-2\beta r_{ij}) \tag{9}$$

where β is a smearing parameter inversely proportional to the charge smearing length λ. Gaussian charge smearing [5] was added in version 2.6 of DL_MESO, i.e.

$$f(r_{ij}) = \mathrm{erfc}\left(\frac{r_{ij}}{2\sigma_G}\right) \tag{10}$$

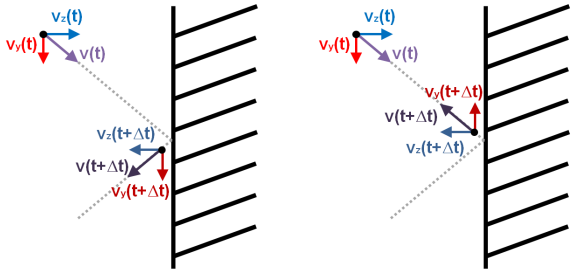
where σ_G is the charge smearing length. (Note that by setting $\alpha = \frac{1}{2\sigma_G}$, all Ewald real-space terms disappear and electrostatic calculations can be carried out in reciprocal space alone.) The upcoming version of DL_MESO_DPD (2.7) will also include the exact form of Slater-type smearing, linear smearing (in line with the original form of DPD electrostatics [37]) and a form based on a sinusoidal function [52].

Integrating dissipative and random forces with the standard Velocity Verlet integrator has been shown to produce deviations in expected temperature, radial distribution function and isothermal compressibility, particularly with larger timesteps [21, 53]. Alternative integrators and non-DPD pairwise thermostats have been made available since version 2.4 of DL_MESO, including the commonly-named 'DPD Velocity Verlet' integrator (Velocity Verlet with re-calculation of dissipative forces using predicted end-step particle velocities), Lowe-Andersen [23], Peters [24] and Stoyanov-Groot [25] thermostats. Version 2.7 will also include Shardlow splitting [22] as an addi-

tional option to more accurately integrate the DPD dissipative and random forces using particle velocities at the beginning of each timestep.

All of the above thermostat and integrator options are most frequently used in constant volume and temperature (NVT) ensembles. Since version 2.4 of DL_MESO, they have also been coupled to Langevin [54] and Berendsen [55] barostats to allow for constant pressure (NPT) ensembles. These were extended in version 2.6 to include constant surface area (NP_nAT) and constant surface tension ($NP_n\gamma T$) ensembles.

A number of DPD simulations require boundary conditions other than periodic ones, e.g. problems involving flow fields or slab geometries. Hard reflecting surfaces were added to version 2.4 of DL_MESO_DPD [56]. This required a change to the export communication step to omit boundary halos at the edges of the simulation cell and movement of particles back into the cell after the first stage of Velocity Verlet force integration, applying specular reflection to give a free-slip boundary (as shown in Figure 2(a)). An additional Groot-Warren potential is also applied to particles within a cutoff distance from each boundary to reduce particle density oscillations due to reflection and control the degree of particle adsorption onto the surface. Version 2.7 of DL_MESO_DPD will additionally offer bounce-back reflections (Figure 2(b)) at hard surfaces to provide no-slip boundaries. For systems with both hard reflecting surfaces and electrostatic interactions, version 2.7 will also include a correction term based on the component of the system dipole moment orthogonal to the boundary [57] along with a vacuum gap (actualised in the form of an adjusted volume for reciprocal space calculations) to reduce the periodicity in that dimension.



(a) Specular reflection (b) Bounce-back reflection

Figure 2.: Schematic representation of specular and bounce-back reflection schemes in two-dimensions. Note that only the velocity component normal to the wall is reversed in specular reflection, leading to a free-slip boundary, while reversing all velocity components in bounce-back reflection leads to a no-slip boundary.

Version 2.5 added both frozen beads (i.e. interacting particles that are fixed in position and whose velocities are kept at zero) and Lees-Edwards periodic shearing boundary conditions [58] for linearly shearing flow systems. The latter of these require special versions of the deport, export and import communication steps to allow particles moving through a Lees-Edwards boundary to re-enter into one of four possible cores' domains based on their required tangential shift, as well as adjustments to Ewald sum reciprocal vectors [59]. The problem of excessive dissipative forces across such boundaries due to large relative velocities was initially mitigated by switching off DPD dissipative and random forces for particle pairs straddling the shearing boundaries [60], but from version 2.6 of DL_MESO onwards the problem is removed in an Galilean invariant manner by adjusting the relative velocity [61].

It is possible to include additional external force fields to a DPD simulation. DL_MESO_DPD has allowed for constant gravitational fields since version 2.4, which applies a force on each particle whose magnitude depends on its mass ($\vec{F}_{i,grav} = m_i\vec{G}$): this form of force field can also be used to apply pressure-driven flows between solid plates. Version 2.6 introduced constant electric fields to

apply forces on particles based on their charges ($\vec{F}_{i,elec} = q_i\vec{E}$).

3.3. *Input and output files*

The original version of DL_MESO_DPD (2.2) required a single input file, `CONTROL`, which contains all of the various values required for the code to setup and run a DPD simulation. These include conservative interaction parameters for each bead type, the dissipative force parameter, system temperature, size and numbers of timesteps, as well as the expected runtime to allow the code to produce restart files for resuming an incomplete simulation. Most lines in this form of the `CONTROL` consist of a keyword (resembling the name of a variable in the code) followed by the required value, separated by at least one space (see Figure 3 (a)).

Versions 2.3 and 2.4 of DL_MESO_DPD continued using the same `CONTROL` file format, adding additional keywords and values to reflect expanding functionality (e.g. additional thermostats and barostats, MDPD). To accommodate newly-introduced bonded interactions in these versions of the code, two new input files were introduced: `MOLECULE` consists of molecule topologies and numbers with the bead species and their relative positions for randomised insertion into the simulation cell (Figure 3 (b)), and `FIELD` consists of the parameters for bonded interactions (stretching, angles and dihedrals) and the molecule beads involved in each of those interactions (Figure 3 (c)).

The restart files, whose names start with `export`, are produced periodically (every 1000 timesteps by default) during a simulation to allow its resumption if the computer inadvertently fails: one file was produced per processor core in binary to speed up later reading, which contains particle positions, velocities, forces, statistical accumulators and random number generator states. From version 2.3 onwards, the `export` file format was modified to include current bond, angle and dihedral tables.

All versions of DL_MESO_DPD create a general-purpose `OUTPUT` file to give a diagnostic summary of the simulation with instantaneous and running-average statistics (potential and kinetic energies, virial, temperature and pressure) at user-specified intervals, while instantaneous values of the statistical properties can also be written periodically to a file called `CORREL`. From version 2.3 onwards, an additional output file format for trajectory data (particle positions and velocities) consisting of binary `HISTORY` files (one per processor core) has been implemented.

An additional file format was also introduced with version 2.4 of DL_MESO to allow users to specify the initial configuration for DPD simulations instead of using the code's default setup. The optional `CONFIG` file is formatted in an identical manner to those used for DL_POLY, which opens up the possibility of sharing particle configurations between DL_MESO_DPD and DL_POLY. An additional post-processing utility to create `CONFIG` files from the `export` files of a previous simulation has made it possible to restart a simulation on a different number of processor cores or to switch off the replicated data form of bonded interaction calculations to improve the computational efficiency of a post-equilibrium 'production run'.

Version 2.5 of DL_MESO_DPD saw fundamental changes to input file formats, taking the lead from DL_POLY and using more human-readable forms with clearer keywords and phrases (see Figure 3 (d) and (e)). By moving bead species definitions and interaction data from the `CONTROL` file and the molecule definitions and insertion configurations from the `MOLECULE` file into the `FIELD` file, this makes the latter file format into the single repository for interaction definitions.

The upcoming version of DL_MESO (2.7) will radically change how trajectory and restart files will be written. Rather than each processor core writing its own `HISTORY` and `export` files, groups of processor cores will gather the required particle data onto a master core before the master cores (typically no more than four in total) collaboratively write their data to a single `HISTORY` or `export` file using MPI-I/O [62]. The restart data will also be split between two files: the `export` file will retain particle positions, velocities and forces, while statistical accmulators and states of barostats and random number generators will be included a new `REVIVE` file (similar to DL_POLY's). A new file

```
DL_MESO micelle self-assembly example

NSYST       87500
VOLM        15625.0
TEMP        1.0
TSTEP       0.01
RCUT        1.0
GAMMA       4.5
RHALO       1.5
TIMJOB      180000.0
TCLOSE      20000.0

KRES        0
NRUN        100000
NSEQL       20000
NSBPO       1000
LTEMP      .TRUE.
NSBTS       1000
NSTK        100
LSBSV      .TRUE.
NSBSV       1000
LCORR      .TRUE.
ISCORR      1000
NSPE        3
NMOL        1
KTYPE       2
LBOND      .TRUE.

SPEC        X     87500  1.0  25.0  1.0  0.0
SPEC        A     0      1.0  25.0  1.0  0.0
SPEC        B     0      1.0  25.0  1.0  0.0

MOLE        AB    3125

UNLIKE      1     2      0.0
UNLIKE      2     3      30.0
UNLIKE      1     3      50.0
```

(a) CONTROL

```
MOLECULE  2  2.000000E+00
2     -0.0518522       0.326715     -0.0332135
3     -0.0733795       0.349223       0.465816
```

(b) MOLECULE

```
BOND    1   100.0  0.5  0.0  0.0
MOLECULE  1
1    2    1
```

(c) FIELD

```
DL_MESO micelle self-assembly example

volume 25.0 25.0 25.0
temperature 1.0
cutoff 1.0
boundary halo 1.5

timestep 0.01
steps 100000
equilibration steps 20000
scale temperature every 1000
trajectory 20000 1000 1
stats every 1000
stack size 100
print every 1000
job time 180000.0
close time 20000.0

ensemble nvt mdvv

finish
```

(d) CONTROL

```
DL_MESO micelle self-assembly example

SPECIES 3
X 1.0 0.0 87500
A 1.0 0.0 0
B 1.0 0.0 0

MOLECULES 1
AB
nummols 3125
beads 2
A 0.0107636 -0.0112540 -0.249515
B -0.0107636 0.0112540 0.249515
bonds 1
harm  1 2 100.0 0.50
finish

INTERACTIONS 6
X X dpd  25.0 1.0 4.5
X A dpd   0.0 1.0 4.5
X B dpd  50.0 1.0 4.5
A A dpd  25.0 1.0 4.5
A B dpd  30.0 1.0 4.5
B B dpd  25.0 1.0 4.5

CLOSE
```

(e) FIELD

Figure 3.: Example representation of input file formats in various versions of DL_MESO_DPD self-assembly of a solution of two-bead amphiphilic molecules (dimers) into micelles. Files (a), (b) and (c) are respectively the CONTROL, MOLECULE and FIELD files used in versions 2.3 and 2.4. Files (d) and (e) are respectively the CONTROL and FIELD files used from version 2.5 onwards.

format to provide components of stress tensors separated by source (e.g. conservative, dissipative and random forces) will also be introduced to allow easier analysis of rheological properties such as the zero-shear viscosity [63].

A number of utilities have been included with DL_MESO to manipulate input and output files for its DPD code. All versions of DL_MESO_DPD since 2.3 have included a utility to create molecule configurations based on positioning beads in branched chains using random walks inside a cube, while utilities to create readable files for the molecular visualisation package VMD [64] from simulation restart (export) and trajectory (HISTORY) files, and to analyse localised properties (e.g. densities, velocities) from trajectory data based on dividing the simulation cell into user-specified cuboids have also been supplied. Version 2.4 included an additional utility to create initial simulation configurations (CONFIG files) from restart data, while version 2.5 added a utility to convert older input files into the new DL_POLY-like formats. Even more utilities were included with version 2.6 of DL_MESO_DPD to manipulate trajectory data: creating CONFIG files from trajectory frames, calculating radial distribution functions, dipole moments, molecular end-to-end distances and radii of gyration, and eigenvalues of second moments of the isosurface normal distribution [33] as sim-

ple order parameters for mesophasic behaviour. The upcoming version 2.7 will also include a new utility to determine excess chemical potentials based on Widom insertion [65] of particles or small molecules into trajectory data frames found in `HISTORY` files.

While several utilities are supplied with DL_MESO, users are free to modify these for their own purposes and devise new analyses to examine trajectory data obtained by its DPD code. Two such examples of external software designed to work with DL_MESO_DPD are UMMAP and some of the E-CAM Meso- and Multi-scale Modules. UMMAP [66] is a trajectory frame analysis tool created by David Bray at STFC Daresbury Laboratory for particle based simulations capable of producing a range of statistics and graphical output, and designed to be a one-stop fully integrated environment for performing analysis on atomistic and coarse-grained molecular simulations. Currently at version 2.7.4, it was originally designed to work with DL_MESO_DPD trajectory data but has since been expanded to work with other particle-based dynamics codes. The Meso- and Multi-scale Modules were created as part of the Horizon 2020 European HPC Centre of Excellence for simulation and modelling (E-CAM) [67, 68], and a number of these (written by Silvia Chiacchiera at STFC Daresbury Laboratory) are designed to work with DL_MESO_DPD. At the time of writing, these include a utility to obtain general information from `HISTORY` files, check DL_MESO_DPD input files for consistency and correctness, and more general-purpose utilities than those supplied with DL_MESO to examine charge dipole moments and their autocorrelation functions.

3.4. *Optimisations*

The earliest optimisations made to DL_MESO_DPD since its development restarted in 2009 had the effect of increasing the range of DPD-based systems the code could model and allowing it to be run more flexibly. For instance, version 2.3 of DL_MESO_DPD allowed simulations to take place in non-cubic orthorhombic boxes with arbitrary total numbers of particles. Version 2.4 allowed the code to use any number of processor cores (not just a power of 2) and increased the maximum number of particle species from four to six. The restrictions on the numbers of particle species and beads per processor core were removed in version 2.5 due to the use of runtime allocatable arrays for particle data: the size of these arrays is based on information given in the input files and the number of available processor cores.

More substantial changes were made in version 2.6 of DL_MESO due to extensive performance testing and optimisations carried out by IBM for BlueGene/Q systems between 2013 and 2016 under the aegis of STFC's Hartree Centre [69] Phase 1 programme ('High Performance Computing'). These changes include rearrangement of particle data arrays to put together x-, y- and z-components for a given particle's positions, velocities or forces next to each other in memory, improvements in automatically determining array sizes to reduce the code's memory footprint, changing core-to-core communications from blocked to unblocked MPI calls (to allow computations while a communication is taking place) and unrolling of nested loops for link cells and Ewald reciprocal vectors. The last of these has also allowed loops of link cells or reciprocal vectors to be divided among OpenMP threads, allowing for an additional parallelism of force calculations: assigning the forces on particles in a threadsafe manner requires additional memory per thread, which prevents different threads competing for the same memory locations and allows for summation afterwards. The calculations of localised densities for MDPD and re-calculations of dissipative forces for the DPD Velocity Verlet integration scheme have also been modified to include all pairwise contributions from particles in boundary halos, eliminating the need for import communication steps to complete calculations of these properties.

A significant portion of force calculations in DL_MESO_DPD is taken up by generating random numbers for the DPD random force (Equation (2)). Both DL_DPD and the original version of DL_MESO_DPD (2.2) make sole use of Marsaglia's universal random number generator [70], while subsequent versions of DL_MESO_DPD make use of the Mersenne Twister (MT19937) [71] for force

calculations: the latter has been found to be faster and have a longer period ($2^{19937}-1$ compared with 2^{144}) but is harder to initialise and make threadsafe for OpenMP multithreaded calculations, as well as requiring more memory to store its state.

An alternative random number generator (Saru) with the option to determine its state from three seeds (based on the timestep and particle numbers) has been proposed for DPD calculations [44]: along with a change to force calculations to include all pairwise contributions from particles in boundary halos and elimination of the import communication step, this random number generator has been shown to improve parallel scalability of DPD calculations [72] while also being threadsafe and giving reproducible results regardless of the number of processor cores in use. Due to its limited period ($2^{63.77}$) and extensive changes required to bond interaction book-keeping to completely remove force import communications, Saru has not yet been generally applied to DL_MESO_DPD: a modified form with a larger period or a similarly 'symmetric' random number generator could justify making these modifications to the code.

Further optimisations have subsequently been carried out to allow DL_MESO_DPD to be ported to heterogeneous computing architectures. The Intel Xeon Phi, a co-processor consisting of many integrated cores, is one such option for accelerating calculations normally carried out by a standard CPU: its use of low-powered cores with many available threads and vector processors brings various opportunities and challenges to obtain boosted performance [73]. Optimisation work carried out by the Intel Parallel Computing Centre (IPCC) at the Hartree Centre as part of its Phase 2 programme ('Energy Efficient Computing and Big Data Analytics') was able to obtain an improvement of around 24% in DL_MESO_DPD's performance on the Knight's Corner form of the Intel Xeon Phi – this exploited the previously added OpenMP multithreading of force calculations and improved it further by applying a threadsafe form of the Mersenne Twister random number generator [74]. Other available hardware accelerators include Graphical Processing Units (GPUs), and work is currently underway as part of the Horizon 2020 European HPC Centre of Excellence for simulation and modelling (E-CAM) [67] to port DL_MESO_DPD to NVIDIA GPUs: this has involved significant rewriting of the main calculation parts of the code into `CUDA` to allow offloading of work from a CPU to a multithreaded GPU and to better exploit the available cache when working through interacting particle pairs. Some of the above improvements will make it into the imminent release of DL_MESO version 2.7, while others will be included in future versions of the software package.

3.5. *Documentation and test cases*

All versions of DL_MESO with a DPD code have been supplied with a user manual, explaining what each of the codes in this package can do and how to use them. While the first of these manuals for DL_MESO version 2.2 had comparatively little information on the DPD code and mainly included a theoretical description of DPD itself, this was substantially expanded upon from version 2.3 onwards with additional chapters on available functionalities beyond the basic theory of DPD and details of the subroutines and functions in DL_MESO_DPD. Later versions have included chapters describing the formats of input and output files, as well as appendices listing the available error and warning messages from DL_MESO_DPD and providing more details about its utilities.

The user manual was originally the sole citeable reference available to indicate use of DL_MESO in published works. The 2013 article in Molecular Simulation [6] was written to make it easier for users to cite use of the code package, as well as provide some background details into DL_MESO_DPD and DL_MESO_LBE, their underlying methodologies and indications of how these codes scale in parallel. While this article more accurately describes the released version of the codes at the time it was written (version 2.5 of DL_MESO), it is still a useful description of the code package and, to date, has been cited more than 70 times since its publication.

Test cases have been supplied with DL_MESO to demonstrate the capabilities of DL_MESO_DPD: these include the required input files and some sample outputs. Illustrations of some of these test

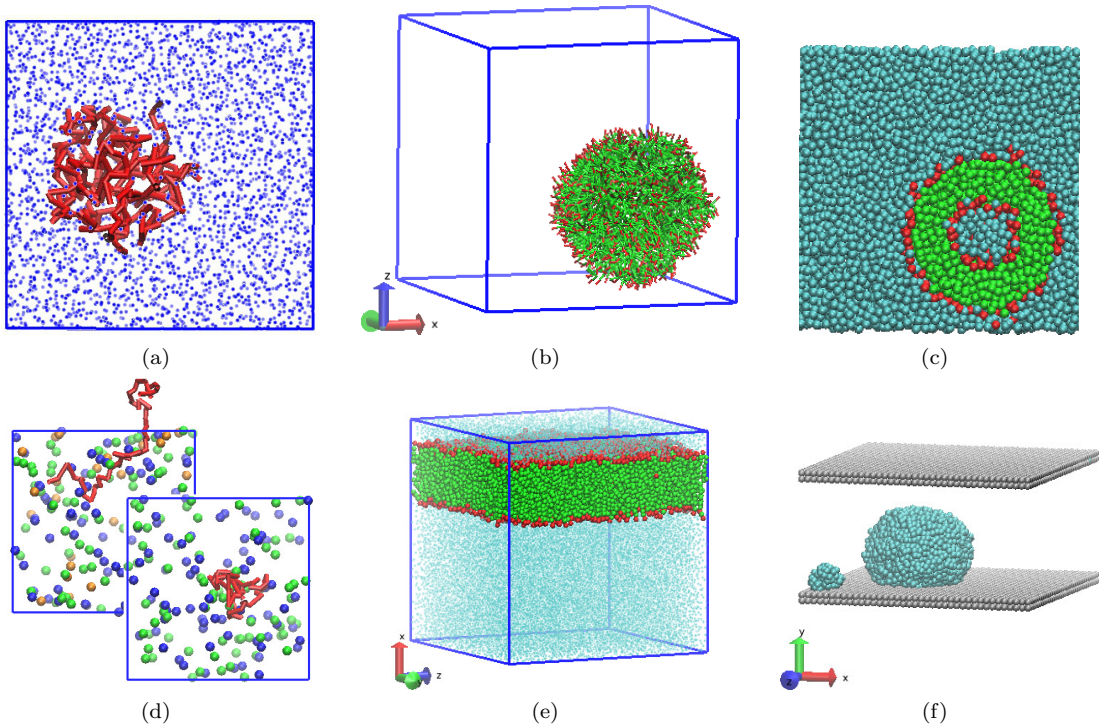


Figure 4.: Demonstration cases supplied with DL_MESO_DPD. (a) Aggregation of hydrophobic molecules (red) in water (blue). (b) Self-assembly of amphiphilic molecules (red = hydrophilic head group, green = hydrophobic tail) in water (cyan), (c) showing cross-section through vesicle with encapsulated water. (d) Hydrophobic polyelectrolyte (left) and neutral polymer (right) in salt solution (green = salt cations, blue = anions, orange = counterions for polyelectrolyte). (e) Formation of lipid bilayer in water. (f) Water drops settling onto hydrophobic wall of frozen particles (grey) under the influence of gravity (using MDPD interactions).

cases are shown in Figure 4. The original release version of the package (2.2) included two simple test cases with 1 000 and 512 000 particles respectively, representing mixtures of two or three bead species with different interaction strengths. These were expanded upon in version 2.3 of DL_MESO to make use of the DPD code's new features, including separation of two immiscible species, aggregation of hydrophobic molecules in water (Figure 4(a)) and self-assembly of amphiphilic dimers into an isotropic mesophase [33]. Version 2.4 expanded on the amphiphilic dimer test case to include additional hexagonal, lamellar and inverse isotropic phases by varying dimer concentrations in solution, as well as adding a case involving self-assembly of longer amphiphilic molecules into a vesicle [34] (Figures 4(b) and 4(c)) and a polyelectrolyte-based case [38] (Figure 4(d)).

Some of the functionality expansions in version 2.5 of DL_MESO_DPD were demonstrated with additional test cases for vapour/liquid equilibrium using MDPD, shear flow of a single fluid using Lees-Edwards boundary conditions and pressure-driven flow of fluid between two walls of frozen particles. The most recent release of DL_MESO (version 2.6) includes a test case based on the formation of a lipid bilayer across the simulation cell from amphiphilic molecules with bond angle potentials to increase hydrophobic tail stiffness [35] (Figure 4(e)) and an additional MDPD test case based on liquid droplets settling onto a hydrophobic frozen bead surface [75] (Figure 4(f)).

3.6. *Influences from* DL_POLY *and vice versa*

There are many points in common between DL_MESO_DPD and the domain decomposed versions of DL_POLY (DL_POLY_3 and DL_POLY_4). All of these codes are influenced by the same original program (MDMEGA) and carry out force calculations based predominately on pairwise contributions and their integration over a given timestep. These codes use the same parallelisation technique to divide the simulation cell and, consequently, particles and computational work between the available processor cores, as well as link-cell lists to find particle pairs within a cutoff distance and core-to-core communications to populate boundary halos with particle data for force calculations.

The manner in which the force calculations needed to be carried out to ensure DPD random forces were indeed pairwise – eliminating duplicate calculations on particle pairs crossing between processor cores and importing force contributions on particles in boundary halos back to their origins – meant that DL_POLY_3 (the first domain decomposed version of DL_POLY) could not readily be adapted to carry out DPD calculations. For that reason and to provide a more manageable code for mesoscale simulations, DL_MESO_DPD has been developed separately from DL_POLY. Modifications were available to allow the replicated data form of DL_POLY – DL_POLY_2 (now superseded by DL_POLY Classic) – to carry out DPD calculations [15], but the requirement for this form of parallelisation to have data for all particles on all processor cores meant that DPD calculations could only be carried out on up to around 30 000 to 50 000 particles based on the available memory per core.

In spite of their separate development, functionalities in both forms of DL_POLY have found their way into DL_MESO_DPD. This has been made easier due to the fact that all of these codes have been written in `Fortran`. While implementations of particular functionalities in DL_MESO_DPD do not have identical source code to those in DL_POLY, comparison of both codes does show similarities.

The clearest example of shared functionality between DL_MESO_DPD and DL_POLY that originates from the latter is Ewald summation, which is used to determine electrostatic interactions between charged particles. Version 2.4 of DL_MESO_DPD included an implementation of the analytical or direct form of Ewald sum. The reciprocal space part of this was based on the replicated-data implementation [50] found in DL_POLY_2 (which is still included in DL_POLY CLASSIC), albeit with some small modifications when calculating the required complex exponential structure factors for the domain-decomposed DPD code. Subsequent versions (2.6 onwards) have optimised this by denesting loops through reciprocal vectors, but the basic premise of the original algorithm remains intact. The upcoming version 2.7 of DL_MESO will also include Smooth Particle Mesh Ewald [51] as a more computationally efficient alternative: this happens to be the default form of reciprocal-space term used in DL_POLY_3 and DL_POLY_4, both of which exploit a method of applying FFTs to domain-decomposed charge grids [76].

More recently, DL_POLY_4 has incorporated DPD functionalities that have been 'borrowed' from DL_MESO_DPD. Version 4.06 (released in August 2014) included Groot-Warren conservative interactions – given in Equation (5) – as an additional force-field option, while version 4.07 (January 2015) has included the DPD thermostat using Shardlow splitting [22], Saru [44] as its random number generator and an additional communication routine to update velocities of particles in boundary halos. Both of these additional functionalities allow DL_POLY_4 to carry out DPD calculations, although these are currently restricted to constant volume (NVT) ensembles of charge-neutral particles as neither barostats nor charge smearing schemes have been applied.

Since version 2.5 of DL_MESO_DPD, both codes also share similar input file formats, making it reasonably straightforward to get them to use the same initial state (`CONFIG`), simulation controls (`CONTROL`) and even particle interactions (`FIELD`). While there are some small differences in `CONTROL` and `FIELD` files between DL_POLY and DL_MESO_DPD, there is nevertheless a high degree of mutual intelligibility between the codes. The fact that the former now includes an implementation of the DPD thermostat and the latter can use Lennard-Jones interactions between particles offers the possibility of both codes using DPD (or other pairwise thermostats) for MD simulations.

4. Current code performance

To give some idea of how the development of DL_MESO_DPD has been progressing, versions 2.5, 2.6 and 2.7 of this software have been tested on various machines. Two sets of tests have been carried out to examine the different forms of parallelism available: domain decomposition using MPI and OpenMP multithreading of force calculations.

4.1. *Domain decomposition tests*

The MPI-based parallelism of DL_MESO_DPD has been tested using the current Phase 2 incarnation of ARCHER (www.archer.ac.uk), a Cray XC30 MPP supercomputer consisting of compute nodes with two 12-core Intel Ivy Bridge series processors and 64 GB of memory each.

For these tests, the strong scalability test cases from the original article [6] – S-DPD-I, S-DPD-II and S-DPD-III – have been used: these are derived from the vesicle formation test case and include 714 984, 2 859 936 and 11 439 744 beads respectively. For a fair basis of comparison due to a lack of OpenMP multithreading in version 2.5, the MPI-only parallelised forms of DL_MESO_DPD have been used on up to 128 nodes (3072 cores). Trajectory frames were written every 2000 timesteps during these calculations, while restart files were written every 1000 timesteps. Due to the internal timing routines included in DL_MESO_DPD, it has been possible to identify the required start-up times to initialise the simulations as well as the time required per timestep due to calculations alone (i.e. without writing information to files).

Tables 2, 3 and 4 give the tabulated data obtained during this study for versions 2.5, 2.6 and 2.7 of DL_MESO_DPD respectively: Figure 5 shows a plot of the parallel scalability for the three versions of the code when file-writing is omitted and Figure 6 plots the overall parallel scalability. For a strong scalability test, perfect parallelisation corresponds to a constant relative speed factor of two when the number of processor cores is doubled, while good parallelisation corresponds to a constant factor of 1.2. These are indicated in Figures 5 and 6 by dashed lines as a visual guide to the reader.

Near-perfect or even super-perfect parallelisation is achieved for all three system sizes and versions of DL_MESO_DPD when using up to 96 processor cores, but the speed gain in using larger number of cores eventually slows and levels off. This type of behaviour is common to domain-decomposed particle codes (e.g. DL_POLY_3 [10]) and is related to the relative amount of time required for communication compared to computational work: this ratio increases as the number of cores increases, giving diminishing returns when applying more computational resources to the calculation.

When writing particle data to `HISTORY` and `export` files is omitted, it is clear that the both the time per timestep and strong parallel scalability of DL_MESO_DPD has markedly improved between versions 2.5 and 2.6, particularly for larger numbers of processor cores. The de-nesting of loops and memory layout of particle data have contributed significantly to the improved performance of the code at a given number of processor cores, while the change from blocked to unblocked MPI calls for the deport, export and import communication stages in each timestep has improved the scalability with numbers of cores. The improvements made for version 2.6 are still evident in version 2.7, as the per-core and parallel performance of the latter are still broadly similar to the former's. Start-up times have also generally decreased between version 2.5 and 2.6 due to minor changes in how the system is initialised, although a clear dependency on those times with processor core count still remains. More substantial changes were made to system initialisation for version 2.7, exploiting generation of the same random numbers on multiple cores for positioning molecules as opposed to broadcasting random numbers generated by a single core. These changes have led to start-up times for a particular system that are mostly independent of the number of cores used, even if these times are sometimes longer than those achieved with earlier versions of DL_MESO_DPD.

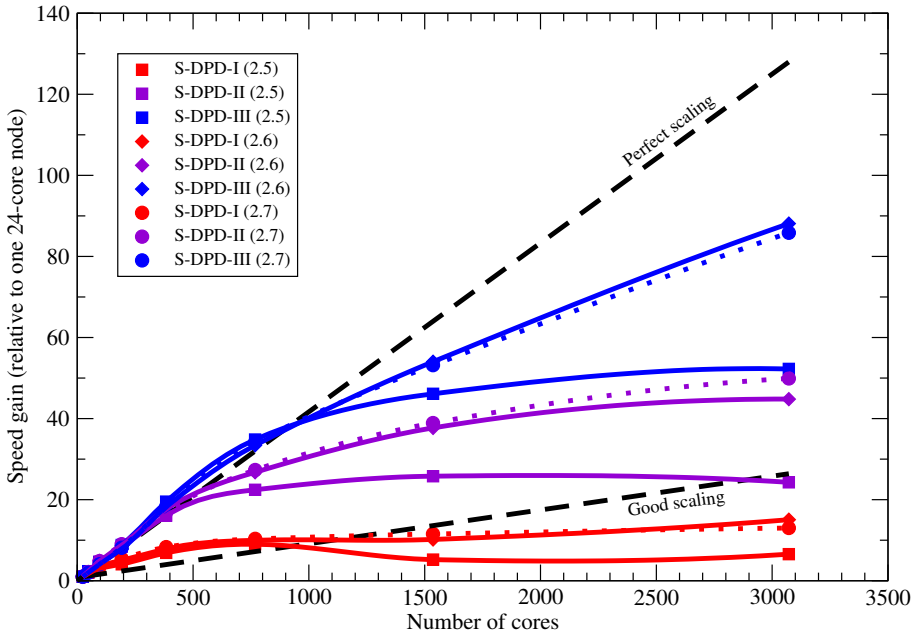


Figure 5.: Results of the strong scalability tests for versions 2.5, 2.6 and 2.7 of DL_MESO_DPD, excluding times for file writing (calculation times only), reporting the observed speed gains for small (S-DPD-I), medium-sized (S-DPD-II) and large (S-DPD-III) systems with increasing numbers of computer cores. (Results for version 2.7 shown as dashed lines for clarity.)

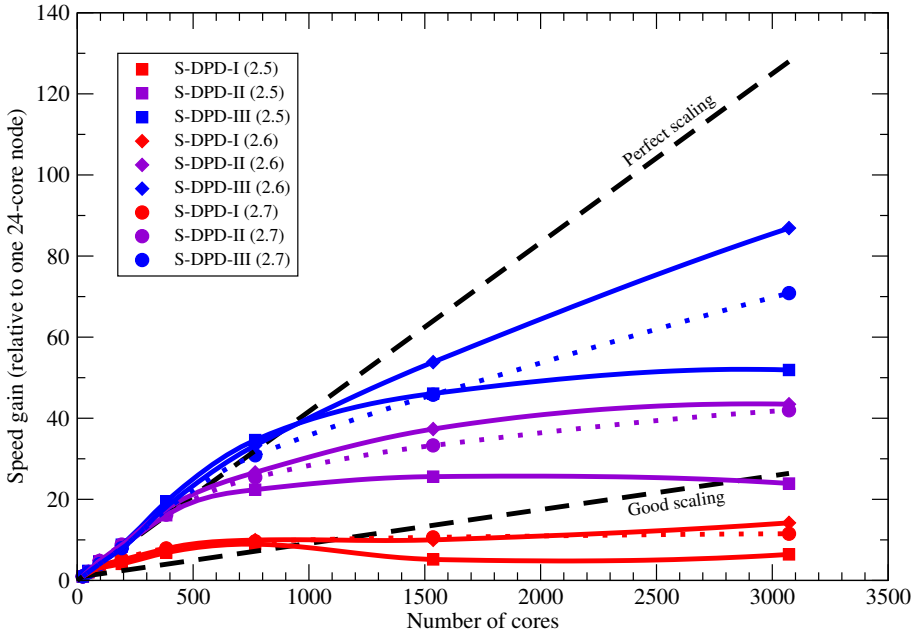


Figure 6.: Results of the strong scalability tests for versions 2.5, 2.6 and 2.7 of DL_MESO_DPD, using total times (including file writing), reporting the observed speed gains for small (S-DPD-I), medium-sized (S-DPD-II) and large (S-DPD-III) systems with increasing numbers of computer cores. (Results for version 2.7 shown as dashed lines for clarity.)

Table 2.: Results of strong scalability study on ARCHER Phase 2 of DL_MESO_DPD version 2.5: start-up times, times per timestep and speed gain relative to one 24-core node.

S-DPD-I: 714 984 beads

Number of nodes	Number of cores	Start-up time/s	Calculation time only		Total time	
			Time per timestep/ms	Speed gain	Time per timestep/ms	Speed gain
1	24	0.14	43.50	1.00	43.59	1.00
2	48	0.18	21.65	2.01	21.74	2.01
4	96	0.38	13.89	3.13	13.90	3.14
8	192	0.85	10.69	4.07	10.71	4.07
16	384	0.82	6.32	6.88	6.36	6.86
32	768	1.30	4.83	9.01	4.87	8.94
64	1536	3.31	8.34	5.22	8.40	5.19
128	3072	5.27	6.63	6.56	6.78	6.43

S-DPD-II: 2 859 936 beads

Number of nodes	Number of cores	Start-up time/s	Calculation time only		Total time	
			Time per timestep/ms	Speed gain	Time per timestep/ms	Speed gain
1	24	0.42	228.23	1.00	228.38	1.00
2	48	0.71	108.08	2.11	108.08	2.11
4	96	0.88	48.13	4.74	48.26	4.73
8	192	1.17	27.21	8.39	27.33	8.36
16	384	1.99	14.20	16.07	14.21	16.07
32	768	3.46	10.17	22.44	10.20	22.38
64	1536	6.11	8.85	25.79	8.92	25.60
128	3072	12.76	9.40	24.28	9.55	23.91

S-DPD-III: 11 439 744 beads

Number of nodes	Number of cores	Start-up time/s	Calculation time only		Total time	
			Time per timestep/ms	Speed gain	Time per timestep/ms	Speed gain
1	24	3.85	1020.99	1.00	1021.23	1.00
2	48	2.54	433.53	2.36	433.97	2.35
4	96	4.83	226.91	4.50	227.14	4.50
8	192	6.93	115.75	8.82	115.88	8.81
16	384	10.90	52.29	19.53	52.43	19.48
32	768	20.48	29.30	34.85	29.50	34.62
64	1536	28.79	22.14	46.12	22.16	46.09
128	3072	59.04	19.54	52.25	19.68	51.89

Writing trajectory and restart data does have a noticeable effect on the overall performance of the codes. The decrease in performance is comparably small – no more than 0.25ms per timestep – for versions 2.5 and 2.6 of DL_MESO_DPD, both of which rely on creating independent files for each processor core. There is a more significant time penalty for version 2.7 of up to 3ms per timestep for the large system (S-DPD-III), due to the need to gather together data among groups of processor cores and grant the root cores in those groups write access to single `HISTORY` and `export` files. Nevertheless, the resulting parallel scalability for version 2.7 still compares well to versions 2.5 and 2.6 and maintains the trends seen in the latter. (It should be noted that some machines may struggle to create large numbers of files per simulation due to limits in file system cache sizes.) The convenience of single output files and corresponding reductions in times taken to process them also outweigh the small loss in performance caused by writing them, as much as 86% less time for the large system S-DPD-III run on 3072 cores.

4.2. *OpenMP multithreading tests*

The addition of OpenMP multithreading to version 2.6 of DL_MESO_DPD was originally intended to take advantage of multiple threads per processor core available in computers with low-powered processors, such as the IBM BlueGene/Q, to compensate for slower clock speeds. To test how well DL_MESO_DPD performs when multiple threads per core are in use, identical simulations were run using 256 cores on each of two supercomputers previously available at the Hartree Centre: an IBM iDataPlex consisting of 2.6 GHz Intel i7 SandyBridge processors (Blue Wonder) and an IBM BlueGene/Q consisting of 1.6 GHz BGC processors (Blue Joule). The simulations consisted

Table 3.: Results of strong scalability study on ARCHER Phase 2 of DL_MESO_DPD version 2.6: start-up times, times per timestep and speed gain relative to one 24-core node.

S-DPD-I: 714 984 beads

Number of nodes	Number of cores	Start-up time/s	Calculation time only		Total time	
			Time per timestep/ms	Speed gain	Time per timestep/ms	Speed gain
1	24	0.11	44.11	1.00	44.22	1.00
2	48	0.20	21.58	2.04	21.66	2.04
4	96	0.28	11.87	3.72	11.88	3.72
8	192	0.78	8.73	5.05	8.75	5.06
16	384	1.52	5.62	7.85	5.65	7.83
32	768	2.13	4.42	9.98	4.45	9.94
64	1536	3.32	4.33	10.19	4.42	10.00
128	3072	5.04	2.93	15.05	3.11	14.20

S-DPD-II: 2 859 936 beads

Number of nodes	Number of cores	Start-up time/s	Calculation time only		Total time	
			Time per timestep/ms	Speed gain	Time per timestep/ms	Speed gain
1	24	0.36	223.13	1.00	223.57	1.00
2	48	0.55	104.66	2.13	104.87	2.13
4	96	1.08	45.56	4.90	45.68	4.89
8	192	1.12	25.69	8.69	25.79	8.67
16	384	2.60	12.61	17.69	12.66	17.66
32	768	5.30	8.34	26.75	8.39	26.64
64	1536	6.78	5.92	37.69	5.99	37.34
128	3072	13.35	4.98	44.81	5.14	43.46

S-DPD-III: 11 439 744 beads

Number of nodes	Number of cores	Start-up time/s	Calculation time only		Total time	
			Time per timestep/ms	Speed gain	Time per timestep/ms	Speed gain
1	24	1.14	871.38	1.00	871.89	1.00
2	48	1.46	430.22	2.03	430.62	2.02
4	96	4.83	215.73	4.04	216.14	4.03
8	192	8.97	111.28	7.83	111.66	7.81
16	384	10.17	47.81	18.23	47.95	18.18
32	768	25.56	26.08	33.41	26.18	33.30
64	1536	29.76	16.13	54.02	16.19	53.84
128	3072	48.65	9.89	88.11	10.03	86.89

of 192 000 particles made up of 11 different types, some of which were connected together in three different molecule types and included Slater-type smeared charges. Table 5 gives the times taken by each machine to complete a timestep with versions 2.5 and 2.6 of DL_MESO_DPD, the latter using MPI-only parallelisation or hybrid MPI/OpenMP parallelisation with 2 or 4 threads per core, while Figure 7 gives a plot of the relative speed-ups for version 2.6 compared with version 2.5.

On both machines, the single-thread (MPI only) form of DL_MESO_DPD version 2.6 was significantly quicker (two to three times faster) than version 2.5 due to the optimisations carried out on loops through link cells. Thread scalability depends strongly on the machine in use: the code performance actually declined on Blue Wonder when multiple threads per core were used, while some modest boosts were observed for Blue Joule. The BGC processors in Blue Joule included more physical threads per core (four) than the SandyBridge processors in Blue Wonder (two), meaning the latter machine had to resort to hyperthreading – i.e. the CPU has to schedule sharing of available resources between virtual OpenMP threads – which resulted in a significant performance hit (up to 28% for four threads). The use of additional memory per thread to assign forces in a threadsafe manner also led to some degradation of performance on both machines: this corresponds to the observed speed-ups during optimisation when the memory allocated for particle data was reduced.

Even with the speed-up obtained using four threads per processor core on Blue Joule (nearly four times compared to version 2.5), the same simulation ran approximately 3.5 times faster on Blue Wonder using the MPI-only version of DL_MESO_DPD version 2.6. This demonstrates the need to carefully consider the balance between domain decomposition (MPI-based) parallelism and multithreading of force calculations when running DL_MESO_DPD on a particular machine, based on the number of available *physical* threads that can be exploited with OpenMP: simply

Table 4.: Results of strong scalability study on ARCHER Phase 2 of DL_MESO_DPD version 2.7: start-up times, times per timestep and speed gain relative to one 24-core node.

S-DPD-I: 714 984 beads

Number of nodes	Number of cores	Start-up time/s	Calculation time only		Total time	
			Time per timestep/ms	Speed gain	Time per timestep/ms	Speed gain
1	24	1.43	45.14	1.00	45.29	1.00
2	48	2.09	21.99	2.05	22.16	2.04
4	96	1.50	12.63	3.57	12.81	3.53
8	192	1.56	8.29	5.45	8.49	5.34
16	384	1.78	5.44	8.30	5.72	7.91
32	768	2.13	4.37	10.33	4.69	9.66
64	1536	2.94	3.91	11.54	4.26	10.63
128	3072	4.69	3.47	13.01	3.93	11.52

S-DPD-II: 2 859 936 beads

Number of nodes	Number of cores	Start-up time/s	Calculation time only		Total time	
			Time per timestep/ms	Speed gain	Time per timestep/ms	Speed gain
1	24	6.01	227.55	1.00	228.09	1.00
2	48	5.65	107.04	2.13	107.57	2.12
4	96	5.71	46.79	4.86	47.36	4.82
8	192	5.58	25.24	9.02	25.76	8.85
16	384	5.90	13.45	16.92	14.06	16.22
32	768	6.28	8.33	27.32	8.97	25.42
64	1536	7.00	5.85	38.90	6.85	33.29
128	3072	8.62	4.56	49.90	5.44	41.96

S-DPD-III: 11 439 744 beads

Number of nodes	Number of cores	Start-up time/s	Calculation time only		Total time	
			Time per timestep/ms	Speed gain	Time per timestep/ms	Speed gain
1	24	21.88	890.16	1.00	892.37	1.00
2	48	22.16	446.38	1.99	448.71	1.99
4	96	21.00	219.06	4.06	221.32	4.03
8	192	22.11	109.54	8.13	111.96	7.97
16	384	22.88	48.61	18.31	51.04	17.48
32	768	23.45	26.01	34.22	28.92	30.86
64	1536	23.64	16.74	53.18	19.48	45.80
128	3072	24.88	10.37	85.84	12.59	70.86

Table 5.: Results of thread scalability study for version 2.6 of DL_MESO_DPD: comparing runtimes for MPI-only and MPI/OpenMP forms of the code against version 2.5 (MPI-only) and corresponding speed gains.

Software version	Threads per core	Blue Wonder (iDataPlex)		Blue Joule (BlueGene/Q)	
		Time per timestep/ms	Speed gain	Time per timestep/ms	Speed gain
2.5	1	21.04	1.00	138.5	1.00
2.6	1	10.31	2.04	42.77	3.24
2.6	2	12.18	1.73	38.71	3.58
2.6	4	13.20	1.59	35.65	3.88

using multiple threads with each core does not always improve performance!

5. Work carried out using DL_MESO_DPD

Both DL_POLY and DL_MESO were conceived as general-purpose 'simulation engines', and as such they were designed to be used for many different scientific problems by users across the world. The number of citations of either user manuals or descriptive articles (e.g. [8], [77], [6]) gives an idea of how widely used the codes happen to be and the types of simulations for which they are used.

At the time of writing (February 2018), DL_MESO is known to have been cited 84 times since 2012. While dwarfed by DL_POLY's hundreds of citations since 1992, this number is still considerable given the comparatively new modelling techniques (DPD and LBE) included in the software package.

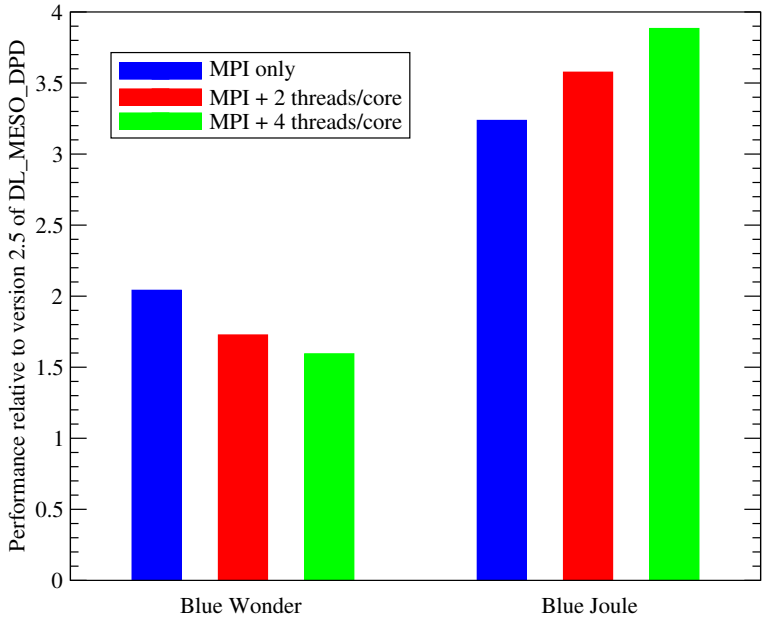


Figure 7.: Results of the OpenMP thread scalability tests for version 2.6 of DL_MESO_DPD, using relative performance against version 2.5 on Blue Wonder (iDataPlex) and Blue Joule (BlueGene/Q).

The majority of these citations refer to DL_MESO_DPD in some form: most of these have directly used the code to carry out DPD simulations, which will be described in more detail below.

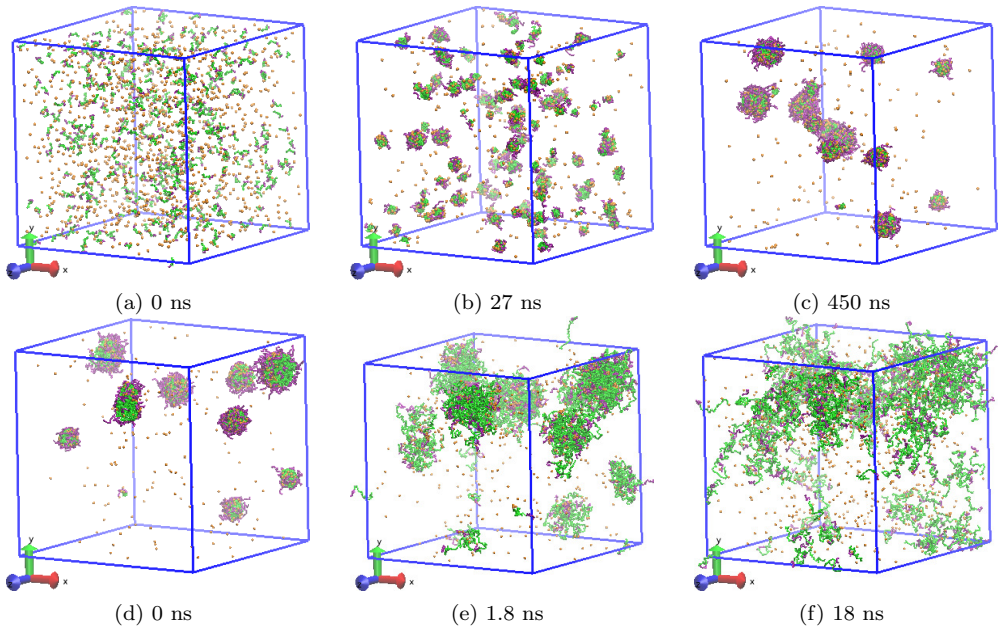
The articles that have not carried out simulations with DL_MESO_DPD mainly refer to the code as available DPD modelling software. Some of these mention DL_MESO_DPD as a code that can model coarse-grained systems [78], electrostatics [79] and many-body DPD [80], as well as an alternative to other developed DPD codes [81, 82]. Other articles refer to DL_MESO_DPD in the context of algorithmic developments – e.g. new pairwise thermostats [83], partial barostats [84], free-energy calculations [85] – or changes in how DPD calculations are carried out, such as random number generators [44], parameterisation with chemical reactivity [86] and parallelisation of Shardlow splitting [87]. A further article [88] did not use DL_MESO_DPD itself but made use of one of its utilities to carry out phase detection analysis, while two others looked at the DPD code's performance on a particular computing architecture (Intel Xeon Phi) [89] and its use on a cloud-based high-performance computing service [90].

The simplicity of DPD as a mesoscale modelling method makes it an ideal candidate to test phenomenological models for various kinds of molecules. A number of articles citing use of DL_MESO_DPD follow this approach, examining self-assembly of chromonic liquid crystals [17], nanorod assembly in gyroid phases of diblock copolymers [91], gold nanoparticles with grafted copolymers [92], antiparallel molecular association in the formation of smectic A phases [93], aqueous mixtures of inorganic nanoparticles with tethered hydrophobic and amphiphilic copolymer chains [94], conformational behaviour of various polymer chains (including stars and dendrimers) in endothermic solvent mixtures [95], and adsorption of amphiphilic graft copolymers onto lyophobic surfaces [96].

One notable group of examples of DL_MESO_DPD's use to solve scientific problems has involved parameterisation from Flory-Huggins χ-parameters between pairs of components [3], which were calculated from energies of mixing obtained from atomistic molecular dynamics or Monte Carlo simulations [97].

The first known article to cite DL_MESO [16] examined pH-driven loading and release of an anticancer drug (camptothecin) inside micelles of amphiphilic copolymers made up of hydrophobic

poly(β-amino ester) (PAE) and hydrophilic methyl ether-capped poly(ethylene glycol) (PEG), basing its parameterisation on the technique mentioned above. Different χ-parameters were obtained between PAE and other components at higher and lower pH from MD simulations, representing the different conditions for loading and release of camptothecin. DL_MESO_DPD was then applied to these processes over timescales (up to microseconds) not readily available with classical MD, providing good correspondence with experimentally-determined loading efficiency. (Demonstrations of the micelle self-assembly and destruction processes, calculated using DL_MESO_DPD version 2.6 on 256 cores of Blue Wonder, are given in Figure 8.) Further simulations of drug loading and release for different drugs and amphiphiles have since been carried out, including amphiphiles with branches grafted to the hydrophobic chain [98] and consideration of the degree of protonation for a zwitterionic drug [99].



(a) 0 ns (b) 27 ns (c) 450 ns

(d) 0 ns (e) 1.8 ns (f) 18 ns

Figure 8.: DPD simulations of pH-sensitive amphiphilic copolymer consisting of poly(β-amino ester) (PAE: green particles) and poly(ethylene glycol) (PEG: purple) in solution with camptothecin (orange) [16]. Initial loading – (a), (b), (c) – into vesicles simulated for system at pH=7.4 over 900 ns, and rapid drug release – (d), (e), (f) – observed at pH=6.4 due to hydrated PAE.

Further applications involving DL_MESO_DPD as the DPD simulation engine have made use of the same technique of parameterising conservative interactions by finding energies of mixing or Hildebrand solubility parameters using atomistic molecular dynamics or Monte Carlo simulations. These include systems involving carbon nanotubes [100, 101], Nafion [102–104], block copolymers with poly(carboxybetaine) and poly(ethylene glycol) [105], poly(amido amine) (PANAM) dendrimers and single-stranded DNA [106], poly(methyl methacrylate-co-butyl acrylate-co-styrene) for strengthening fragile papers [107], channel membranes of polystyrene-*block*-poly(4-vinyl pyridine) block copolymer [108] and phase behaviour of sophorolipids [109].

Implementation of Ewald sums with smeared charges has made DL_MESO_DPD a desirable code for DPD simulations with electrostatic interactions. Several articles have been published making use of this functionality, on systems involving self-assembly of polyelectrolytes [110], phase behaviours of diblock amphiphilic copolymers in ionic solutions [111, 112], use of surfactants to reduce interfacial tension between oil and water [113], membranes between Nafion and ionic liquids [114], pH-sensitive loading and release of doxorubicin by PANAM dendrimers [115] and self-assembly of the pH-responsive polyester dendrimer H_2O-COOH [116]. A recent article has modelled nano-colloid

electrophoresis [117], exploiting both electrostatic interactions and non-DPD pairwise thermostats (Lowe-Andersen and Stoyanov-Groot) available in DL_MESO_DPD.

The ability of DPD and other pairwise thermostats to correctly model hydrodynamics coupled with solid or shearing boundary conditions enables DL_MESO_DPD to model systems with flow fields. Most of the articles on flow-based systems involve Lees-Edwards shearing boundaries applied to simple fluids [118–120], colloidal suspensions [121, 122], polydispersed linear molecules [123] or micelles [124]. Two other articles took different approaches: one explicitly modelled electroosmotic flow in nanochannels by applying an electric field on charged particles between frozen particle walls [125], while the other applied a constant force on rod-like proteins inside slits and cylindrical pores of frozen particles coated with polymer brushes [126]. Both of these used DL_MESO_DPD with some modifications: the former applied dipole-based forces [57] and changes to the reciprocal-space part of Ewald sums to provide a vacuum gap, while the latter incorporated bounce-back reflections to the slits and pores[2].

Other articles citing use of DL_MESO_DPD have made modifications to the code to incorporate additional functionalities. One such modification has been the addition of an anharmonic Morse potential on top of the Groot-Warren conservative interaction to form dissociable bonds between particles [127]. This model can represent proton-base bonds, allowing for formation of intermediate complexes and artificial mimicking of the Grotthuss diffusion mechanism. The modified version of DL_MESO_DPD has subsequently been used to examine water diffusion and proton conductivity in membranes of hydrated polyelectrolytes and Nafion [18, 128].

DL_MESO_DPD has additionally been applied when new procedures in analysis and parameterisation have been developed. Calculations of critical micelle concentrations (CMCs) have been carried out for ionic surfactants [129], while more general protocols for measuring micelle cluster sizes and CMCs have been established [130]. Algorithms based on proper orthogonal decomposition have been developed to de-noise flow data obtained from particle-based simulations and tested with DPD simulations [131, 132]. A new top-down method of finding conservative force parameters has been devised [28], based on simulating phase separations to match experimentally determined partition coefficients: very large simulations (1.2 million particles) with DL_MESO_DPD were carried out to find A_{ij} values for various organic species and their accuracy tested by determining CMCs for various alkyl ethoxylate surfactants.

6. Summary and outlook

DL_MESO_DPD has been developed significantly since its first release in 2006. Its development has responded to the needs of its users, incorporating new functionalities to broaden the range of DPD-based simulations that can be carried out and applying various optimisations to improve its efficiency, parallel scalability and simulation throughput. While its development has not occurred identically to DL_POLY's due to different design philosophies and applications, the two codes do have quite a lot in common: a shared predecessor in MDMEGA [42], similarities in basic operation and some functionalities, input file formats, and generality within their modelling methods (i.e. they are capable of modelling wide ranges of systems).

Its primary parallelisation strategy of domain decomposition has made DL_MESO_DPD a highly-scalable simulation engine suitable for modelling a wide range of DPD-based systems. The majority of simulations carried out make use of the code as supplied, several taking advantage of functionalities such as smeared-charge electrostatic interactions, shearing flow boundaries and alternative pairwise thermostats to DPD (e.g. Lowe-Andersen). Some studies have made modifications to DL_MESO_DPD to further expand its available functionality, aided by its modular code structure and detailed documentation.

[2]All of these changes were made to versions 2.5 and 2.6 of DL_MESO_DPD: these will be standard features in the upcoming version (2.7).

Future releases of DL_MESO_DPD will include additional functionalities: examples of these include more efficient and scalable electrostatics solvers such as Particle-Particle Particle-Mesh [37] and Smooth Particle Mesh Ewald [51] methods, tabulated potentials, additional integrators and pairwise thermostats [22, 61, 83], and fixed-length bond constraints [133] as an alternative to stretching springs. Code optimisations continue both for standard CPU-based computers and for hybrid forms with accelerators attached to host CPUs, such as Intel Xeon Phi co-processors and Graphical Processor Units (GPUs). These changes will ensure that DL_MESO continues to be a valuable software package for researchers involved in mesoscale modelling.

DL_MESO can be obtained via its website at www.ccp5.ac.uk/DL_MESO. After registration, the software is available free of charge for non-profit organisations and delivered with the source code. Installation instructions, the user manual in PDF format and examples are included in the package. Both the website and the user manual include up-to-date lists of the features currently implemented in each code.

Acknowledgments

The author is grateful to a number of funding bodies, particularly EPSRC via the Computational Science Centre for Research Communities (CoSeC), whose financial support led to the creation of DL_MESO under the auspices of CCP5 and continues to support its further development via CCP5 and the High End Computing consortium UKCOMES. Many thanks are especially due to Rongshan Qin (currently at The Open University) and William Smith, the originators and original developers of DL_MESO.

Specifically for DL_MESO_DPD, the author would like to thank Ilian Todorov, Richard Anderson, David Bray, Annalaura Del Regno and Silvia Chiacchiera (all at STFC Daresbury Laboratory), Ard van Bergen (Novidec Ltd) and William Swope (IBM Research) for practical suggestions that have led to code improvements. Many thanks are due to Michael Johnston and Leopold Grinberg (IBM Research) for their general-purpose and IBM-specific optimisations of DL_MESO_DPD, as well as Luke Mason and Stephen Pickles (STFC Hartree Centre) for optimisations of DL_MESO_DPD for Intel Xeon Phi systems.

Thanks to Nidhi Raj for carrying out the drug loading/release simulations on the Hartree Centre's Blue Wonder machine with DL_MESO_DPD version 2.6 (Figure 8) in August 2016.

Many thanks are due to the many past and current users of DL_MESO_DPD who have helped to improve the code by reporting bugs and making suggestions for new features.

References

[1] Hoogerbrugge PJ, Koelman JMVA. Simulating microscopic hydrodynamic phenomena with dissipative particle dynamics. EPL (Europhysics Letters). 1992;19:155–160; Available from: http://stacks.iop.org/0295-5075/19/155.

[2] Schlijper AG, Hoogerbrugge PJ, Manke CW. Computer simulation of dilute polymer solutions with the dissipative particle dynamics method. Journal of Rheology. 1995;39:567–579; Available from: http://link.aip.org/link/?JOR/39/567/1.

[3] Groot RD, Warren PB. Dissipative particle dynamics: Bridging the gap between atomistic and mesoscopic simulation. Journal of Chemical Physics. 1997;107:4423–4435; Available from: http://aip.scitation.org/doi/10.1063/1.474784.

[4] Warren PB. Vapor-liquid coexistence in many-body dissipative particle dynamics. Physical Review E. 2003 Dec;68:066702.

[5] Warren PB, Vlasov A. Screening properties of four mesoscale smoothed charge models, with application to dissipative particle dynamics. Journal of Chemical Physics. 2014;140:084904; Available from: http://scitation.aip.org/content/aip/journal/jcp/140/8/10.1063/1.4866375.

[6] Seaton M, Anderson R, Metz S, et al. DL_MESO: highly scalable mesoscale simulations. Molecular Simulation. 2013;39:796–821.
[7] Smith W, Forester TR. DL_POLY_2.0: A general-purpose parallel molecular dynamics simulation package. Journal of Molecular Graphics. 1996 Jun;14:136–141; Available from: http://www.sciencedirect.com/science/article/B6VNC-3VV619Y-3/2/e80d1e3054af4333e3a108aa8f593163.
[8] Smith W, Yong CW, Rodger PM. DL_POLY: Application to molecular simulation. Molecular Simulation. 2002;28:385–471; Available from: http://www.informaworld.com/10.1080/08927020290018769.
[9] Todorov IT, Smith W. DL_POLY_3: the CCP5 national UK code for molecular-dynamics simulations. Philosophical Transactions of the Royal Society of London A. 2004;362:1835–1852; Available from: http://rsta.royalsocietypublishing.org/content/362/1822/1835.abstract.
[10] Todorov IT, Smith W, Trachenko K, et al. DL_POLY_3: new dimensions in molecular dynamics simulations via massive parallelism. Journal of Materials Chemistry. 2006;16:1911–1918.
[11] Succi S. The lattice boltzmann equation for fluid dynamics and beyond. Oxford: Clarendon Press; 2001.
[12] Chen S, Doolen GD. Lattice Boltzmann method for fluid flows. Annual Review of Fluid Mechanics. 1998;30:329–364; Available from: http://arjournals.annualreviews.org/doi/abs/10.1146/annurev.fluid.30.1.329.
[13] Chen S, Doolen GD, Eggert KG. Lattice-Boltzmann fluid dynamics: a versatile tool for multiphase and other complicated flows. Los Alamos Science. 1994;22:98–111; Available from: http://www.fas.org/sgp/othergov/doe/lanl/pubs/00285550.pdf.
[14] Pinches MRS, Tildesley DJ, Smith W. Large scale molecular dynamics on parallel computers using the link-cell algorithm. Molecular Simulation. 1991;6:51–87; Available from: https://doi.org/10.1080/08927029108022139.
[15] Elliott JA, Benedict M, Dutt M. Applications of DL_POLY to modelling of mesoscopic particulate systems. Molecular Simulation. 2006;32:1113–1121; Available from: http://www.informaworld.com/10.1080/08927020600978846.
[16] Luo Z, Jiang J. pH-sensitive drug loading/releasing in amphiphilic copolymer PAE–PEG: Integrating molecular dynamics and dissipative particle dynamics simulations. Journal of Controlled Release. 2012 Aug;162:185–193; Available from: http://www.sciencedirect.com/science/article/pii/S0168365912005275.
[17] Walker M, Masters AJ, Wilson MR. Self-assembly and mesophase formation in a non-ionic chromonic liquid crystal system: insights from dissipative particle dynamics simulations. Physical Chemistry Chemical Physics. 2014 Nov;16:23074–23081; Available from: http://dx.doi.org/10.1039/C4CP03092C.
[18] Lee MT, Vishnyakov A, Neimark AV. Coarse-grained model of water diffusion and proton conductivity in hydrated polyelectrolyte membrane. Journal of Chemical Physics. 2016;144:014902; Available from: http://scitation.aip.org/content/aip/journal/jcp/144/1/10.1063/1.4938271.
[19] Español P, Warren P. Statistical mechanics of dissipative particle dynamics. EPL (Europhysics Letters). 1995;30:191–196; Available from: http://stacks.iop.org/0295-5075/30/191.
[20] Swope WC, Andersen HC, Berens PH, et al. A computer simulation method for the calculation of equilibrium constants for the formation of physical clusters of molecules: Application to small water clusters. Journal of Chemical Physics. 1982;76:637–649; Available from: https://doi.org/10.1063/1.442716.
[21] Besold G, Vattulainen I, Karttunen M, et al. Towards better integrators for dissipative particle dynamics simulations. Physical Review E. 2000 Dec;62:R7611–R7614.
[22] Shardlow T. Splitting for dissipative particle dynamics. SIAM Journal on Scientific Computing. 2003;24:1267–1282; Available from: http://link.aip.org/link/?SCE/24/1267/1.
[23] Lowe CP. An alternative approach to dissipative particle dynamics. EPL (Europhysics Letters). 1999 Jul;47:145–151; Available from: http://dx.doi.org/10.1209/epl/i1999-00365-x.
[24] Peters EAJF. Elimination of time step effects in DPD. EPL (Europhysics Letters). 2004 May;66:311–317; Available from: http://dx.doi.org/10.1209/epl/i2004-10010-4.
[25] Stoyanov SD, Groot RD. From molecular dynamics to hydrodynamics: A novel Galilean invariant thermostat. Journal of Chemical Physics. 2005;122:114112 (pages 8); Available from: http://link.aip.org/link/?JCP/122/114112/1.
[26] Vattulainen I, Karttunen M, Besold G, et al. Integration schemes for dissipative particle dynamics

simulations: From softly interacting systems towards hybrid models. Journal of Chemical Physics. 2002;116:3967–3979; Available from: http://link.aip.org/link/?JCP/116/3967/1.
[27] Vishnyakov A, Lee MT, Neimark AV. Prediction of the critical micelle concentration of nonionic surfactants by dissipative particle dynamics simulations. Journal of Physical Chemistry Letters. 2013; 4:797–802; Available from: http://pubs.acs.org/doi/abs/10.1021/jz400066k.
[28] Anderson RL, Bray DJ, Ferrante AS, et al. Dissipative particle dynamics: systematic parametrization using water-octanol partition coefficients. Journal of Chemical Physics. 2017;147:094503; Available from: http://dx.doi.org/10.1063/1.4992111.
[29] Pagonabarraga I, Frenkel D. Dissipative particle dynamics for interacting systems. Journal of Chemical Physics. 2001;115:5015–5026; Available from: http://link.aip.org/link/?JCP/115/5015/1.
[30] Trofimov SY, Nies ELF, Michels MAJ. Thermodynamic consistency in dissipative particle dynamics simulations of strongly nonideal liquids and liquid mixtures. Journal of Chemical Physics. 2002; 117:9383–9394; Available from: http://link.aip.org/link/?JCP/117/9383/1.
[31] Ghoufi A, Malfreyt P. Calculation of the surface tension from multibody dissipative particle dynamics and Monte Carlo methods. Physical Review E. 2010 Jul;82:016706.
[32] Ghoufi A, Malfreyt P. Mesoscale modeling of the water liquid-vapor interface: A surface tension calculation. Physical Review E. 2011 May;83:051601; Available from: http://link.aps.org/doi/10.1103/PhysRevE.83.051601.
[33] Prinsen P, Warren PB, Michels MAJ. Mesoscale simulations of surfactant dissolution and mesophase formation. Physical Review Letters. 2002 Sep;89:148302.
[34] Yamamoto S, Maruyama Y, Hyodo Sa. Dissipative particle dynamics study of spontaneous vesicle formation of amphiphilic molecules. Journal of Chemical Physics. 2002;116:5842–5849; Available from: http://link.aip.org/link/?JCP/116/5842/1.
[35] Shillcock JC, Lipowsky R. Equilibrium structure and lateral stress distribution of amphiphilic bilayers from dissipative particle dynamics simulations. Journal of Chemical Physics. 2002;117:5048–5061.
[36] Terrón-Mejía KA, López-Rendón R, Goicochea AG. Electrostatics in dissipative particle dynamics using Ewald sums with point charges. Journal of Physics: Condensed Matter. 2016;28:425101; Available from: http://stacks.iop.org/0953-8984/28/i=42/a=425101.
[37] Groot RD. Electrostatic interactions in dissipative particle dynamics—simulation of polyelectrolytes and anionic surfactants. Journal of Chemical Physics. 2003;118:11265–11277; Available from: http://link.aip.org/link/?JCP/118/11265/1.
[38] González-Melchor M, Mayoral E, Velázquez ME, et al. Electrostatic interactions in dissipative particle dynamics using the Ewald sums. Journal of Chemical Physics. 2006;125:224107 (pages 9); Available from: http://link.aip.org/link/?JCP/125/224107/1.
[39] Peter EK, Pivkin IV. A polarizable coarse-grained water model for dissipative particle dynamics. Journal of Chemical Physics. 2014;141:164506; Available from: http://scitation.aip.org/content/aip/journal/jcp/141/16/10.1063/1.4899317.
[40] Peter EK, Lykov K, Pivkin IV. A polarizable coarse-grained protein model for dissipative particle dynamics. Physical Chemistry Chemical Physics. 2015;17:24452–24461; Available from: http://dx.doi.org/10.1039/C5CP03479E.
[41] Español P, Warren PB. Perspective: dissipative particle dynamics. Journal of Chemical Physics. 2017; 146:150901; Available from: http://dx.doi.org/10.1063/1.4979514.
[42] Smith W. MDMEGA: CCP5 Program Library molecular dynamics program for large LJ systems. 1992; Available from: ftp://ftp.dl.ac.uk/ccp5/MDMEGA/mdmega.f.
[43] Smith W, Dean C, Fincham D, et al. DL_POLY: a macromolecular simulation package. CCP5 Quarterly Newsletter: Information Quarterly for Computer Simulation of Condensed Phases. 1992 Sep; 35:15–16.
[44] Afshar Y, Schmid F, Pishevar A, et al. Exploiting seeding of random number generators for efficient domain decomposition parallelization of dissipative particle dynamics. Computer Physics Communications. 2013 Apr;184:1119–1128; Available from: http://www.sciencedirect.com/science/article/pii/S0010465512003992.
[45] Weeks JD, Chandler D, Andersen HC. Role of repulsive forces in determining the equilibrium structure of simple liquids. Journal of Chemical Physics. 1971;54:5237–5247; Available from: http://link.aip.org/link/?JCP/54/5237/1.
[46] Mohammad-Aghaie D, Papari MM, Ebrahimi AR. Determination of transport proper-

ties of dilute binary mixtures containing carbon dioxide through isotropic pair potential energies. Chinese Journal of Chemical Engineering. 2014;22:274–286; Available from: http://www.sciencedirect.com/science/article/pii/S1004954114600185.

[47] Dünweg B, Paul W. Brownian dynamics simulations without Gaussian random numbers. International Journal of Modern Physics C. 1991;2:817–827.

[48] Seaton M. DL_MESO INFOMAIL mail shots. 2018; Available from: https://www.scd.stfc.ac.uk/Pages/DL_MESO-infomail.aspx.

[49] CoSeC - Computational Science Centre for Research Communities. 2017; Available from: https://www.scd.stfc.ac.uk/Pages/CoSeC.aspx.

[50] Smith W. A replicated data molecular dynamics strategy for the parallel Ewald sum. Computer Physics Communications. 1992 Jan;67:392–406; Available from: http://www.sciencedirect.com/science/article/B6TJ5-46FXBY4-7Y/2/25e9ab24d0f908dc0789effebcfbed6d.

[51] Essmann U, Perera L, Berkowitz ML, et al. A smooth particle mesh Ewald method. Journal of Chemical Physics. 1995;103:8577–8593; Available from: http://link.aip.org/link/?JCP/103/8577/1.

[52] Gavrilov AA, Chertovich AV, Kramarenko EY. Dissipative particle dynamics for systems with high density of charges: Implementation of electrostatic interactions. Journal of Chemical Physics. 2016;145:174101; Available from: http://scitation.aip.org/content/aip/journal/jcp/145/17/10.1063/1.4966149.

[53] Nikunen P, Karttunen M, Vattulainen I. How would you integrate the equations of motion in dissipative particle dynamics simulations? Computer Physics Communications. 2003;153:407–423; Available from: http://www.sciencedirect.com/science/article/B6TJ5-48WPW1K-1/2/d25e30f09258658a0a85172c11d83f06.

[54] Jakobsen AF. Constant-pressure and constant-surface tension simulations in dissipative particle dynamics. Journal of Chemical Physics. 2005;122:124901 (pages 8); Available from: http://link.aip.org/link/?JCP/122/124901/1.

[55] Berendsen HJC, Postma JPM, van Gunsteren WF, et al. Molecular dynamics with coupling to an external bath. Journal of Chemical Physics. 1984;81:3684–3690; Available from: http://link.aip.org/link/?JCP/81/3684/1.

[56] Warren PB, Prinsen P, Michels MAJ. The physics of surfactant dissolution. Philosophical Transactions of the Royal Society of London A. 2003;361:665–676; Available from: http://rsta.royalsocietypublishing.org/content/361/1805/665.abstract.

[57] Yeh IC, Berkowitz ML. Ewald summation for systems with slab geometry. Journal of Chemical Physics. 1999;111:3155–3162; Available from: http://link.aip.org/link/?JCP/111/3155/1.

[58] Lees AW, Edwards SF. The computer study of transport processes under extreme conditions. Journal of Physics C. 1972;5:1921–1928; Available from: http://stacks.iop.org/0022-3719/5/1921.

[59] Wheeler DR, Fuller NG, Rowley RL. Non-equilibrium molecular dynamics simulation of the shear viscosity of liquid methanol: adaptation of the Ewald sum to Lees-Edwards boundary conditions. Molecular Physics. 1997;92:55–62; Available from: http://www.tandfonline.com/doi/abs/10.1080/002689797170608.

[60] Chatterjee A. Modification to Lees-Edwards periodic boundary condition for dissipative particle dynamics simulation with high dissipation rates. Molecular Simulation. 2007;33:1233–1236; Available from: http://www.tandfonline.com/doi/abs/10.1080/08927020701713894.

[61] Leimkuhler B, Shang X. Pairwise adaptive thermostats for improved accuracy and stability in dissipative particle dynamics. Journal of Computational Physics. 2016 Nov;324:174–193; Available from: http://www.sciencedirect.com/science/article/pii/S0021999116303291.

[62] Todorov I, Bush I, Porter A. The need for parallel I/O in classical molecular dynamics. In: Cray User Group (CUG) 2008: Crossing the Boundaries; 2008. Available from: https://cug.org/5-publications/proceedings_attendee_lists/2008CD/S08_Proceedings/pages/Authors/01-5Monday/Todorov-Monday3B/Todorov-Monday3B-paper.pdf.

[63] Español P. Hydrodynamics from dissipative particle dynamics. Physical Review E. 1995 Aug;52:1734–1742.

[64] Humphrey W, Dalke A, Schulten K. VMD: Visual molecular dynamics. Journal of Molecular Graphics. 1996;14:33–38; Available from: http://www.sciencedirect.com/science/article/pii/0263785596000185.

[65] Widom B. Some topics in the theory of fluids. Journal of Chemical Physics. 1963;39:2808–2812; Avail-

able from: http://link.aip.org/link/?JCP/39/2808/1.
[66] Bray D. UMMAP. 2017; Available from: www.scd.stfc.ac.uk/Pages/UMMAP.aspx.
[67] E-CAM - European HPC Centre of Excellence. 2017; Available from: www.e-cam2020.eu.
[68] E-CAM Software Repositories. 2017; Available from: www.e-cam2020.eu/software-library/.
[69] Hartree Centre. 2018; Available from: www.hartree.stfc.ac.uk.
[70] Marsaglia G, Zaman A, Tsang WW. Toward a universal random number generator. Statistics and Probability Letters. 1990 Jan;9:35–39; Available from: http://www.sciencedirect.com/science/article/B6V1D-45DHJ42-S/2/65eff74c666aaddad67c802c9175ee62.
[71] Matsumoto M, Nishimura T. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. ACM Transactions on Modeling and Computer Simulation. 1998;8:3–30; Available from: http://www.math.sci.hiroshima-u.ac.jp/ m-mat/MT/ARTICLES/earticles.html.
[72] Seaton M, Todorov I, Afshar Y. Efficient domain decomposition of dissipative particle dynamics via choice of pseudorandom number generator. In: Malyshkin V, editor. Parallel Computing Technologies. 12th International Conference, PaCT 2013, St Petersburg, Russia, September 30 – October 4, 2013. Proceedings; Sep.; Lecture notes in computer science (LNCS), 6873. St Petersburg, Russia: Springer Berlin / Heidelberg; 2013. p. 250–257; Available from: http://link.springer.com/chapter/10.1007/978-3-642-39958-9_23.
[73] Seaton M, Mason L, Matveev ZA, et al. High Performance Parallelism Pearls. Multicore and Many-core Programming Approaches. Volume Two. Elsevier; 2015. Chapter 23: Vectorization Advice; p. 441–462.
[74] Saito M, Matsumoto M. SIMD-oriented fast mersenne twister: a 128-bit pseudorandom number generator. Vol. Monte Carlo and Quasi-Monte Carlo Methods 2006. Heidelberg: Springer Verlag; 2008. Chapter 36; p. 607–622; Available from: http://dx.doi.org/10.1007/978-3-540-74496-2_36.
[75] Johansson E. Simulating fluid flow and heat transfer using dissipative particle dynamics. Department of Energy Sciences, Faculty of Engineering, Lund University, Box 118, 22100 Lund, Sweden: Department of Energy Sciences, Faculty of Engineering, Lund University; 2012. Project report; Available from: http://www.ht.energy.lth.se/fileadmin/ht/Kurser/MVK160/2012/Erik_Johansson.pdf.
[76] Bush I, Todorov I, Smith W. A DAFT DL_POLY distributed memory adaptation of the smoothed particle mesh Ewald method. Computer Physics Communications. 2006 Sep; 175:323–329; Available from: http://www.sciencedirect.com/science/article/B6TJ5-4K4WMYH-2/2/ad58d65c1265b0de7673c031b7523496.
[77] Smith W, Todorov I. A short description of DL_POLY. Molecular Simulation. 2006;32:935–943; Available from: http://dx.doi.org/10.1080/08927020600939830.
[78] Soper A, Edler K. Coarse-grained empirical potential structure refinement: application to a reverse aqueous micelle. Biochimica et Biophysica Acta (BBA) - General Subjects. 2017;1861:1652–1660; recent Advances in Bionanomaterials; Available from: http://www.sciencedirect.com/science/article/pii/S030441651730082X.
[79] Meneses-Juárez E, Márquez-Beltrán C, Rivas-Silva JF, et al. The structure and interaction mechanism of a polyelectrolyte complex: a dissipative particle dynamics study. Soft Matter. 2015;11:5889–5897; Available from: http://dx.doi.org/10.1039/C5SM00911A.
[80] Warren PB. No-go theorem in many-body dissipative particle dynamics. Physical Review E. 2013 Apr; 87:045303; Available from: http://link.aps.org/doi/10.1103/PhysRevE.87.045303.
[81] Palmer TL, Baardsen G, Skartlien R. Reduction of the effective shear viscosity in polymer solutions due to crossflow migration in microchannels: Effective viscosity models based on DPD simulations. Journal of Dispersion Science and Technology. 2018;39:190–206; Available from: http://dx.doi.org/10.1080/01932691.2017.1306784.
[82] Doi H, Saito T, Hiroki O, et al. New development and performance evaluation of dissipative particle dynamics (DPD) program CAMUS. Journal of Computer Chemistry, Japan. 2018;16:126–128.
[83] Leimkuhler B, Shang X. On the numerical treatment of dissipative particle dynamics and related systems. Journal of Computational Physics. 2015;280:72–95; Available from: http://www.sciencedirect.com/science/article/pii/S0021999114006445.
[84] Lin Y, Pan D, Li J, et al. Application of Berendsen barostat in dissipative particle dynamics for nonequilibrium dynamic simulation. Journal of Chemical Physics. 2017;146:124108; Available from: http://dx.doi.org/10.1063/1.4978807.

[85] Cheng J, Vishnyakov A, Neimark AV. Adhesion of nanoparticles to polymer brushes studied with the ghost tweezers method. Journal of Chemical Physics. 2015;142:034705; Available from: http://scitation.aip.org/content/aip/journal/jcp/142/3/10.1063/1.4905894.
[86] Brennan JK, Lísal M, Moore JD, et al. Coarse-grain model simulations of nonequilibrium dynamics in heterogeneous materials. Journal of Physical Chemistry Letters. 2014;5:2144–2149; Available from: http://pubs.acs.org/doi/abs/10.1021/jz500756s.
[87] Larentzos JP, Brennan JK, Moore JD, et al. Parallel implementation of isothermal and isoenergetic Dissipative Particle Dynamics using Shardlow-like splitting algorithms. Computer Physics Communications. 2014;185:1987–1998; Available from: http://www.sciencedirect.com/science/article/pii/S001046551400109X.
[88] Mu J, Motokawa R, Akutsu K, et al. A novel microemulsion phase transition: toward the elucidation of third-phase formation in spent nuclear fuel reprocessing. Journal of Physical Chemistry B. 2018; 122:1439–1452; pMID: 29216427; Available from: http://dx.doi.org/10.1021/acs.jpcb.7b08515.
[89] Elisseev V, Baker J, Morgan N, et al. Energy aware scheduling study on BlueWonder. In: Proceedings of the 4th International Workshop on Energy Efficient Supercomputing; Salt Lake City, Utah; E2SC '16. Piscataway, NJ, USA: IEEE Press; 2016. p. 61–68; Available from: https://doi.org/10.1109/E2SC.2016.14.
[90] AbdelBaky M, Diaz-Montes J, Johnston M, et al. Exploring HPC-based scientific software as a service using CometCloud. In: 10th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing; Nov. IEEE; 2014.
[91] Chakraborty S, Roy S. Structure of nanorod assembly in the gyroid phase of diblock copolymer. Journal of Physical Chemistry B. 2015;119:6803–6812; pMID: 25965314; Available from: http://dx.doi.org/10.1021/acs.jpcb.5b01338.
[92] Posel Z, Posocco P, Lisal M, et al. Highly grafted polystyrene/polyvinylpyridine polymer gold nanoparticles in a good solvent: effects of chain length and composition. Soft Matter. 2016;12:3600–3611; Available from: http://dx.doi.org/10.1039/C5SM02867A.
[93] Walker M, Wilson MR. Simulation insights into the role of antiparallel molecular association in the formation of smectic A phases. Soft Matter. 2016;12:8876–8883; Available from: http://dx.doi.org/10.1039/C6SM01920J.
[94] Šindelka K, Limpouchová Z, Štěpánek M, et al. Stabilization of coated inorganic nanoparticles by amphiphilic copolymers in aqueous media. Dissipative particle dynamics study. Colloid and Polymer Science. 2017;:1–13Available from: http://dx.doi.org/10.1007/s00396-017-4090-0.
[95] Suchá L, Limpouchová Z, Procházka K. Conformational behavior of polymer chains of different architectures in strongly endothermic solvent mixtures: specific solvation effects. Colloid and Polymer Science. 2017;:1–13Available from: http://dx.doi.org/10.1007/s00396-017-4083-z.
[96] Posel Z, Svoboda M, Limpouchova Z, et al. Adsorption of amphiphilic graft copolymers in solvents selective for the grafts on a lyophobic surface: a coarse-grained simulation study. 2018; accepted for publication in *Physical Chemistry Chemical Physics*; Available from: http://dx.doi.org/10.1039/C7CP08327K.
[97] Akkermans RLC. Mesoscale model parameters from molecular cluster calculations. Journal of Chemical Physics. 2008;128:244904 (pages 13); Available from: http://link.aip.org/link/?JCP/128/244904/1.
[98] Luo Z, Li Y, Wang B, et al. pH-sensitive vesicles formed by amphiphilic grafted copolymers with tunable membrane permeability for drug loading/release: a multiscale simulation study. Macromolecules. 2016;49:6084–6094; Available from: http://dx.doi.org/10.1021/acs.macromol.6b01211.
[99] Min W, Zhao D, Quan X, et al. Computer simulations on the pH-sensitive triblock copolymer containing zwitterionic sulfobetaine as a novel anti-cancer drug carrier. Colloids and Surfaces B: Biointerfaces. 2017;152:260–268; Available from: http://www.sciencedirect.com/science/article/pii/S0927776517300425.
[100] Chakraborty S, Choudhury CK, Roy S. Morphology and dynamics of carbon nanotube in polycarbonate carbon nanotube composite from dissipative particle dynamics simulation. Macromolecules. 2013; 46:3631–3638; Available from: http://pubs.acs.org/doi/abs/10.1021/ma302425s.
[101] Roy S, Chakraborty S. Self-assembly of polymer carbon nanotube composites and block co-polymers: from multiscale simulations. SMC Bulletin. 2014 Dec;5:14–20.
[102] Johansson EO, Yamada T, Sundén B, et al. Dissipative particle dynamics approach for nano-scale membrane structure reconstruction and water diffusion coefficient esti-

mation. International Journal of Hydrogen Energy. 2015;40:1800–1808; Available from: http://www.sciencedirect.com/science/article/pii/S0360319914031085.
[103] Hassanzadeh Afrouzi H, Moshfegh A, Farhadi M, et al. Dissipative particle dynamics simulation hydrated Nafion EW 1200 as fuel cell membrane in nanoscopic scale. Transport Phenomena in Nano and Micro Scales. 2017;5:44–53; Available from: http://tpnms.usb.ac.ir/article_2862.html.
[104] Vanya P, Sharman J, Elliott JA. Mesoscale simulations of confined Nafion thin films. Journal of Chemical Physics. 2017 Dec;147:214904; Available from: https://doi.org/10.1063/1.4996695.
[105] Liao M, Liu H, Guo H, et al. Mesoscopic structures of poly(carboxybetaine) block copolymer and poly(ethylene glycol) block copolymer in solutions. Langmuir. 2017;33:7575–7582; pMID: 28689413; Available from: http://dx.doi.org/10.1021/acs.langmuir.7b01610.
[106] Su Y, Quan X, Li L, et al. Computer simulation of DNA condensation by PAMAM dendrimer. Macromolecular Theory and Simulations. 2018;:17000701700070; Available from: http://dx.doi.org/10.1002/mats.201700070.
[107] Qiao L, Chen K, Zhao D, et al. The application of poly(methyl methacrylate-co-butyl acrylate-co-styrene) in reinforcing fragile papers: experiments and computer simulations. Cellulose. 2017 Nov; 24:5157–5171; Available from: https://doi.org/10.1007/s10570-017-1470-z.
[108] Wang C, Quan X, Liao M, et al. Computer simulations on the channel membrane formation by nonsolvent induced phase separation. Macromolecular Theory and Simulations. 2017;:17000271700027; Available from: http://dx.doi.org/10.1002/mats.201700027.
[109] Sarkar S, Chakraborty S, Roy S. Phase diagram of self-assembled sophorolipid morphologies from mesoscale simulations. Journal of Molecular Liquids. 2018;254:198–207; Available from: https://www.sciencedirect.com/science/article/pii/S016773221734223X.
[110] Šindelka K, Limpouchová Z, Lísal M, et al. Dissipative particle dynamics study of electrostatic self-assembly in aqueous mixtures of copolymers containing one neutral water-soluble block and one either positively or negatively charged polyelectrolyte block. Macromolecules. 2014 Aug;47:6121–6134; Available from: http://dx.doi.org/10.1021/ma501018x.
[111] Posel Z, Limpouchová Z, Šindelka K, et al. Dissipative particle dynamics study of the pH-dependent behavior of poly(2-vinylpyridine)-block-poly(ethylene oxide) diblock copolymer in aqueous buffers. Macromolecules. 2014 Mar;47:2503–2514; Available from: http://pubs.acs.org/doi/abs/10.1021/ma402293c.
[112] Mai J, Sun D, Li L, et al. Phase behavior of an amphiphilic block copolymer in ionic liquid: a dissipative particle dynamics study. Journal of Chemical & Engineering Data. 2016;61:3998–4005; Available from: http://dx.doi.org/10.1021/acs.jced.6b00522.
[113] Martiz A, Samaniego S, Aray Y, et al. Synergism between ionic and nonionic surfactants for producing low interfacial tension at oil-water interface. In: SPE Latin American and Caribbean Petroleum Engineering Conference, 18-20 November, Quito, Ecuador. Society of Petroleum Engineers; 2015.
[114] Mai JL, Sun DL, Quan XB, et al. Mesoscopic structure of nafion-ionic liquid membrane using dissipative particle dynamics simulations. Acta Physico-Chimica Sinica. 2016;32:1649–1657; Available from: http://www.whxb.pku.edu.cn/EN/abstract/abstract29456.shtml.
[115] Su Y, Quan X, Min W, et al. Dissipative particle dynamics simulations on the loading and release of doxorubicin by PAMAM dendrimers. CIESC Journal. 2017;68:1757–1766.
[116] Yu C, Ma L, Li K, et al. Computer simulation studies on the pH-responsive self-assembly of amphiphilic carboxy-terminated polyester dendrimers in aqueous solution. Langmuir. 2017;33:388–399; pMID: 28001081; Available from: http://dx.doi.org/10.1021/acs.langmuir.6b03480.
[117] Hassanzadeh Afrouzi H, Moshfegh A, Farhadi M, et al. Dissipative particle dynamics: effects of thermostating schemes on nano-colloid electrophoresis. Physica A: Statistical Mechanics and its Applications. 2018;497:285–301; Available from: https://www.sciencedirect.com/science/article/pii/S037843711830027X.
[118] Moshfegh A, Jabbarzadeh A. Dissipative particle dynamics: effects of parameterization and thermostating schemes on rheology. Soft Materials. 2015;13:106–117; Available from: http://dx.doi.org/10.1080/1539445X.2015.1022898.
[119] Moshfegh A, Jabbarzadeh A. Modified Lees–Edwards boundary condition for dissipative particle dynamics: hydrodynamics and temperature at high shear rates. Molecular Simulation. 2015;41:1264–1277; Available from: http://dx.doi.org/10.1080/08927022.2014.976762.
[120] Moshfegh A, Ahmadi G, Jabbarzadeh A. Thermostatic and rheological responses of DPD fluid to

extreme shear under modified Lees-Edwards boundary condition. European Physical Journal E. 2015 Dec;38:134.

[121] Moshfegh A, Jabbarzadeh A. Simulation of semidilute suspensions by dissipative particle dynamics. In: Oñate E, Oliver J, Huerta A, editors. Proceedings of 11th World Congress on Computational Mechanics (WCCM XI), 5th European Conference on Computational Mechanics (ECCM V) and 6th European Conference on Computational Fluid Dynamics (ECFD VI); Jul.; 2014. p. 1511–1522; Available from: http://www.wccm-eccm-ecfd2014.org/admin/files/fileabstract/a3009.pdf.

[122] Moshfegh A, Jabbarzadeh A. Calibration of dissipative particle dynamics method to study rheology of dense suspensions. In: Advances of Computational Mechanics in Australia; Jul.; Applied Mechanics and Materials; Vol. 846. Trans Tech Publications; 2016. p. 163–168.

[123] Gooneie A, Schuschnigg S, Holzer C. Coupled orientation and stretching of chains in mesoscale models of polydisperse linear polymers in startup of steady shear flow simulations. Macromolecular Theory and Simulations. 2016 Mar;25:170–186; Available from: http://dx.doi.org/10.1002/mats.201500060.

[124] Prhashanna A, Khan SA, Chen SB. Micelle morphology and chain conformation of triblock copolymers under shear: LA-DPD study. Colloids and Surfaces A: Physicochemical and Engineering Aspects. 2016 Oct;506:457—-466; Available from: http://www.sciencedirect.com/science/article/pii/S0927775716305167.

[125] Moshfegh A, Jabbarzadeh A. Fully explicit dissipative particle dynamics simulation of electroosmotic flow in nanochannels. Microfluidics and Nanofluidics. 2016;20:1–17; Available from: http://dx.doi.org/10.1007/s10404-016-1733-2.

[126] Posel Z, Svoboda M, Colina CM, et al. Flow and aggregation of rod-like proteins in slit and cylindrical pores coated with polymer brushes: insight from dissipative particle dynamics. Soft Matter. 2017; 13:1634–1645.

[127] Lee MT, Vishnyakov A, Neimark AV. Modeling proton dissociation and transfer using dissipative particle dynamics simulation. Journal of Chemical Theory and Computation. 2015;11:4395–4403; Available from: http://dx.doi.org/10.1021/acs.jctc.5b00467.

[128] Vishnyakov A, Mao R, Lee MT, et al. Coarse-grained model of nanoscale segregation, water diffusion, and proton transport in Nafion membranes. Journal of Chemical Physics. 2018;148:024108; Available from: https://doi.org/10.1063/1.4997401.

[129] Mao R, Lee MT, Neimark AV, et al. Modeling aggregation of ionic surfactants using a smeared charge approximation in dissipative particle dynamics simulations. Journal of Physical Chemistry B. 2015 Sep;119:11673–11683; pMID: 26241704; Available from: http://dx.doi.org/10.1021/acs.jpcb.5b05630.

[130] Johnston MA, Swope WC, Jordan KE, et al. Toward a standard protocol for micelle simulation. Journal of Physical Chemistry B. 2016;120:6337–6351; pMID: 27096611; Available from: http://dx.doi.org/10.1021/acs.jpcb.6b03075.

[131] Zimoń M, Reese J, Emerson D. A novel coupling of noise reduction algorithms for particle flow simulations. Journal of Computational Physics. 2016;321:169–190; Available from: http://www.sciencedirect.com/science/article/pii/S0021999116301942.

[132] Zimoń M, Prosser R, Emerson D, et al. An evaluation of noise reduction algorithms for particle-based fluid simulations in multi-scale applications. Journal of Computational Physics. 2016;325:380–394; Available from: http://www.sciencedirect.com/science/article/pii/S0021999116303709.

[133] Andersen HC. RATTLE: a "velocity" version of the SHAKE algorithm for molecular dynamics calculations. Journal of Computational Physics. 1983;52:24–34; Available from: http://www.sciencedirect.com/science/article/pii/0021999183900141.