

# ESPResSo 3.1: Molecular Dynamics Software for Coarse-Grained Models

Axel Arnold, Olaf Lenz, Stefan Kesselheim, Rudolf Weeber,  
Florian Fahrenberger, Dominic Roehm, Peter Košov, and Christian Holm

**Abstract** ESPRESSO is a package for Molecular Dynamics (MD) simulations of coarse-grained models. We present the most recent version 3.1 of our software, highlighting some recent algorithmic extensions to version 1.0 presented in a previous paper (Limbach et al. Comput Phys Commun 174:704–727, 2006). A major strength of our package is the multitude of implemented methods for calculating Coulomb and dipolar interactions in periodic and partially periodic geometries. Here we present some more recent additions which include methods for systems with dielectric contrasts that frequently occur in coarse-grained models of charged systems with implicit water models, and an alternative, completely local electrostatic solver that is based on the electrodynamic equations. We also describe our approach to rigid body dynamics that uses MD particles with fixed relative positions. ESPRESSO now gained the ability to add bonds during the integration, which allows to study e.g. agglomeration. For hydrodynamic interactions, a thermalized lattice Boltzmann solver has been built into ESPRESSO, which can be coupled to the MD particles. This computationally expensive algorithm can be greatly accelerated by using Graphics Processing Units. For the analysis of time series spanning many orders of magnitude in time scales, we implemented a hierarchical generic correlation algorithm for user-configurable observables.

**Keywords** Molecular dynamics • Coarse-graining • Lattice-Boltzmann

---

A. Arnold (✉) · O. Lenz · S. Kesselheim · R. Weeber · F. Fahrenberger · D. Roehm · P. Košov · C. Holm

Institute for Computational Physics, Universität Stuttgart, Allmandring 3, 70569 Stuttgart, Germany

e-mail: [arnolda@icp.uni-stuttgart.de](mailto:arnolda@icp.uni-stuttgart.de)

# 1 Introduction

Nowadays, computer simulations are a well established tool in theoretical physics. Here we intend to give an introduction to our Molecular Dynamics package ESPRESSO [6, 26]; ESPRESSO is an acronym for **E**xtensible **S**imulation **P**ackage for **R**esearch on **S**oft matter systems.

The term *soft matter*, or *complex fluids*, as they are called in the American literature, describes a large class of materials, such as polymers, colloids, liquid crystals, glasses, hydrogels and dipolar fluids; familiar examples of such materials are glues, paints, soaps or baby diapers. Also most biological materials are soft matter – DNA, membranes, filaments and other proteins belong to this class. Soft matter research has experienced an increased interest in the last two decades due to its potentially high usefulness in many areas such as technology, biophysics, and nanoscience.

Many properties of soft matter materials emerge on the molecular rather than the atomistic level: the elasticity of rubber is the result of entropy of the long polymer molecules, and the superabsorbing materials used in modern diapers store water inside a polyelectrolyte network. To reproduce these physical effects on the atomistic level in computer simulations, one would have to incorporate many millions of atoms and simulate them for time scales of up to seconds in some cases, which is not possible even with the most powerful modern computers. However, in many cases a much simpler description of the material is sufficient. Polymers such as polyelectrolytes or rubber often can be modeled by simple *bead-spring models*, i.e. (charged) spheres connected by springs, where each of the spheres represents a whole group of atoms, sometimes a complete monomer or even larger compounds. Although these models hide most of the chemical properties, they are quite successful in the description of polymers and other soft matter systems. The process of removing degrees of freedom (here the atomistic description) from a system to obtain a simpler model is called *coarse-graining*.

Computer simulations of such coarse-grained models sometimes still require the incorporation of several thousands of beads and springs, and many time steps (in case of MD) for data acquisition, and therefore one still is in the need of an efficient simulation software. Furthermore, the generation of sufficient amounts of uncorrelated data might still require the usage of non-standard algorithms (i.e. Gibbs ensemble, expanded ensemble techniques, AdResS [22] just to name a few) which in some cases even have been developed especially for a specific model. Therefore it is necessary that the simulation software is much more flexible than standard atomistic simulation packages (as for example GROMACS [18]<sup>1</sup> or NAMD [35]<sup>2</sup>). ESPRESSO was designed and implemented specifically to address

---

<sup>1</sup><http://www.gromacs.org>

<sup>2</sup><http://www.ks.uiuc.edu/Research/namd/>

the requirements of simulating such coarse-grained models, and it has a number of unique characteristics that distinguish it from any other simulation package that we know of.

One should be aware that the flexibility of ESPRESSO also costs some performance: compared to highly optimized MD programs such as GROMACS, ESPRESSO is slower by a factor of about 2. However, one should keep in mind that most of the problems, that we designed ESPRESSO for, can normally not be treated without massive changes to the simulation engines of these fast codes.

ESPRESSO is not a self-contained package but relies on other open source packages. Most prominent is the use of the Tcl scripting language interpreter for the simulation control, which is required to run ESPRESSO. Other packages are optional. If ESPRESSO is to be executed in parallel, an implementation of the MPI standard [32] is required. The P<sup>3</sup>M algorithm for electrostatic interactions relies on the FFTW package.<sup>3</sup> The development process is supported heavily by the use of the git version control system,<sup>4</sup> which allows several developers to work simultaneously on the code, L<sup>A</sup>T<sub>E</sub>X and doxygen<sup>5</sup> for the documentation, and the GNU Autotools<sup>6</sup> for compilation.

In the following Sect. 2, we will detail the unique characteristics of the ESPRESSO simulation software, while in Sect. 3 we will give an overview of the various algorithms and methods implemented in the software. The subsequent Sects. 4–8 will describe the recent algorithmic developments in ESPRESSO. We conclude with an outlook on future plans for our software in Sect. 9.

## 2 Characteristics

*Optimized for coarse-grained models* ESPRESSO was explicitly designed for simulating coarse-grained models. On the one hand, this means that the software has a few characteristics that are otherwise uncommon to most simulation software, such as the fact that the software is *controlled by a scripting language* or that it is *extensible* (see below). On the other hand, the software implements a number of special methods that only make sense in the context of coarse-grained models, such as the possibilities to create rigid bodies (see Sect. 5), to dynamically generate new bonds during the simulation (see Sect. 6), or to couple the many-particle simulation to a Lattice Boltzmann (LB) fluid (see Sect. 7), to name just a few.

---

<sup>3</sup><http://fftw.org>

<sup>4</sup><http://git-scm.com>

<sup>5</sup><http://doxygen.org>

<sup>6</sup><http://ftp.gnu.org/gnu/automake/>

Also note that the software does not enforce any specific length, energy or time unit system, as such units are often inappropriate for coarse-grained models. Instead, the values can be given in any consistent unit system.

*Controlled by a scripting language* The software uses the scripting language Tcl<sup>7</sup> to control the simulations. However, as the Tcl language is mostly outdated nowadays, we are currently working on a new interface to the scripting language Python<sup>8</sup> for one of the next versions of ESPRESSO.

The simulation control script determines all simulation parameters such as the number and type of particles, the type of interactions between these particles, and how the system is propagated; most of the parameters can be changed even during runtime of the simulation. By that, one can perform highly complex simulation procedures, such as adapting the interaction parameters to the current configuration during the simulation, applying or removing spatial constraints, or even complex schemes like parallel tempering or hybrid Monte Carlo. This flexibility is unmatched by any other simulation package that we know of.

*Extensible* Users can read and modify the code to meet their own needs. Throughout the source code of ESPRESSO, readability is preferred over code optimizations. This allows users to extend the code with more ease. Furthermore, we have defined a number of interfaces that allow to implement extensions to the core code (e.g. new potentials, new thermostats or new analysis routines). The extensibility allows researchers to easily incorporate new methods and algorithms into the software, so that ESPRESSO can be used both as a production platform to generate simulation data as well as a research platform to test new algorithms.

*Free and open source* ESPRESSO is an open-source program that is published under the GNU public license. Both the source code of release versions of the software as well as the bleeding-edge development code, are available through our web page<sup>9</sup> and the GNU Savannah project page.<sup>10</sup>

*Parallelized* ESPRESSO is parallelized, allowing for simulations of millions of particles on hundreds of CPUs. ESPRESSO scales well, it can achieve an efficiency of about 70 % on 512 Power4+ processors, and even better values on the more recent BlueGene/P or Cray XT6 systems. Since it contains some of the fastest currently available simulation algorithms, it also scales well with the number of particles, allowing for the simulation of large systems.

*Portable* The code kernel is written in simple ANSI C, therefore it can be compiled on a wide variety of hardware platforms like desktop workstations, convenience clusters and high performance supercomputers based on POSIX operating systems (for example all variants of Unix that we know of, including GNU/Linux).

---

<sup>7</sup><http://www.tcl.tk>

<sup>8</sup><http://www.python.org>

<sup>9</sup><http://espressomd.org>

<sup>10</sup><http://savannah.nongnu.org/projects/espressomd/>

### 3 Methods and Algorithms

In the following, we will give an overview of the algorithms and methods that ESPRESSO provides. Some of these are standard algorithms, like they are described in, e.g., the book Understanding Molecular Simulation [15].

*Ensembles* The software can perform MD simulations in a number of different physical ensembles, using the Velocity-Verlet integration scheme. Besides the microcanonical (NVE) ensemble, it is possible to simulate the canonical (NVT) ensemble via the Langevin thermostat and the NPT (constant pressure) ensemble via a barostat algorithm by Kolb and Dünweg [24]. The constant chemical potential ensemble ( $\mu$ VT) can be realized by hybrid Molecular Dynamics/Monte Carlo on the script language level.

*Nonbonded potentials* For nonbonded interactions between different particle species, a number of different potentials are implemented, for example the Lennard-Jones, Morse and Buckingham potentials. In addition, it is possible to use tabulated potentials of arbitrary form.

*Bonded potentials* ESPRESSO provides a number of interactions between two or more specific particles, including the FENE and harmonic bond potentials, bond angle and dihedral interactions. As in the nonbonded case, potentials in arbitrary form can be specified via tables.

*Anisotropic interactions* Besides the standard integrator, the software also has a quaternion integrator [30] where particles have an orientation represented by a quaternion and rotational degrees of freedom. Therefore, particles cannot only represent isotropic spheres, but also objects that interact via anisotropic interactions such as Gay-Berne ellipsoids [16].

*Electro- and Magnetostatics* The package implements a number of fast parallel state-of-the-art algorithms for electrostatic and magnetostatic interactions that can handle full or partial periodicity, and that even allow to treat systems with dielectric contrast. The methods are detailed in Sect. 4.

*Constraints* ESPRESSO has the ability to fix some or all coordinates of a particle, or to apply an arbitrary external force on each particle. In addition, various spatial constraints, such as spherical or cubic compartments, can be used. These constraints interact with the particles by any nonbonded interaction.

*Rigid bodies* Since version 3.0, it is possible to form rigid bodies out of several particles. This feature allows for arbitrarily complex extended objects in the model, and is described in Sect. 5.

*Dynamic bonding* Agglomeration can be modeled by dynamically adding bonds during the simulation when particles collide. This special feature is described in Sect. 6.

*Hydrodynamics* Hydrodynamic interactions can be modeled via a thermal Lattice Boltzmann method (see Sect. 7) that can be coupled to the particles. To accelerate the algorithm, it is possible to run it on a graphics processor (GPU). Alternatively, particles can use Dissipative Particle Dynamics (DPD) [14, 40] to simulate hydrodynamic effects.

*Analysis* All analysis routines are available in the simulation engine, allowing for both online analysis (during the simulation) as well as offline analysis. ESPRESSO can measure various observables of the system, such as the energy and (isotropic) pressure, or the forces acting on particles or spatial constraints. There are routines to determine particle distributions and structure factors, and some polymer-specific measures such as the end-to-end distance or the radius of gyration. For visualization, ESPRESSO can output files that can be read by visualization software such as VMD<sup>11</sup> [20]. It is simple for users to add their own observables. To allow for measuring correlations in timeseries (such as the mean-square displacement of particles), ESPRESSO contains a generic implementation of *correlators*, which is detailed in Sect. 8.

*AdResS* ESPRESSO contains an implementation of the **Ad**aptive **R**esolution **S**cheme (AdResS) that allows to simulate systems that contain areas with different levels of coarse-graining [22].

## 4 Advanced Electrostatics

Coulomb interactions cannot be calculated by the same straightforward neighborhood strategies as short-ranged interactions, due to the slow decay of the Coulomb potential. Especially in conjunction with periodic (or partially periodic) boundary conditions, one needs special algorithms that are adapted to the particular periodic structure. For systems with three periodic dimensions, which is the most common case as it is used for investigating bulk systems, ESPRESSO offers the P<sup>3</sup>M [10, 11] and MEMD algorithms [28, 34]. P<sup>3</sup>M is also able to treat dipolar systems, such as ferrofluids [9]. For systems with only two periodic dimensions and a third dimension with finite extension, which are commonly used to investigate thin films or surfaces, ESPRESSO features the MMM2D [3] and ELC methods [5], and for rod-like systems, such as nanotubes or pores, only one periodic and two nonperiodic dimensions are required. For these systems, ESPRESSO offers the MMM1D algorithm [4].

Since the last publication on ESPRESSO [26], the electrostatics algorithms have been considerably enhanced. The P<sup>3</sup>M implementation allows now to use arbitrary simulation box dimensions, i.e. not just cubic boxes. Still, our implementation features full error control and automatic parameter tuning, as described by Deserno et al. [11]. Other extensions that are to our knowledge unique to ESPRESSO will be described in this section. First, we will discuss extensions to the MMM2D and ELC methods, which allow to take into account the presence of planar dielectric interfaces. Next, we will introduce the ICC★ method, which allows to

---

<sup>11</sup><http://www.ks.uiuc.edu/Research/vmd/>

take into account arbitrarily shaped dielectric interfaces. Finally, we will discuss the MEMD electrostatics algorithm, a local approach to electrostatics, based on the electrodynamic equations that use an appropriately adapted speed of light.

## 4.1 Dielectric Contrasts

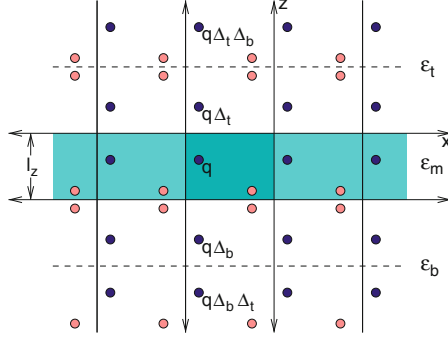
In coarse-grained systems with only partially periodic boundaries and implicit water models, where the water is modeled as a dielectric continuum with a relative dielectric constant, the dielectric contrasts between the embedding medium and the outside can be quite considerable. For example, the relative dielectric constant for bulk water at room temperature is 80, whereas it has a value of  $\approx 1$  in the surrounding air. When studying ions in front of a metallic electrode, the latter even has an infinite dielectric constant. Due to the different dielectric media, polarization occurs, which has a non-negligible influence on the charges in these systems. For example, Messina [31] has shown that image charges due to polarization may lead to considerable reduction in the degree of polyelectrolyte adsorption onto an oppositely charged surface and by that inhibit charge inversion of the substrate.

In computational studies, these polarization effects need to be taken into account. At present, ESPRESSO supports this by extensions to the ELC and MMM2D algorithms, or by the novel ICC $\star$  algorithm, as described below. The first extensions reach a higher precision and are faster, but less flexible than the ICC $\star$  method and can handle only two planar, parallel dielectric interfaces that can have, however, arbitrary dielectric contrast. These need to be parallel to the two periodic dimensions, which is sufficient to model both the two interfaces surrounding a thin film, or the single dielectric jump at a wall. The dielectric boundary conditions are taken into account by the method of image charges [39]. We assume a system, where the charges are embedded in a medium characterized by a dielectric constant  $\varepsilon_m$  which is confined from above by a medium of dielectric constant  $\varepsilon_t$  and from below by a medium of dielectric constant  $\varepsilon_b$ , compare Fig. 1. In case only a single interface should be considered, this can be achieved by choosing  $\varepsilon_t = \varepsilon_m$  or  $\varepsilon_b = \varepsilon_m$ .

In the general case of two dielectric boundaries, an infinite number of image charges arises for each single physical charge, compare Fig. 1. For example, a charge  $q$  at a position  $z$  gives rise to an image charge of  $\Delta_b q$  at  $-z$  in the lower dielectric layer and an image charge of  $\Delta_t q$  at  $(2l_z - z)$  in the upper dielectric layer. The image charge  $\Delta_b q$  gives rise to another image charge  $\Delta_t \Delta_b q$  in the top dielectric domain. Similarly  $\Delta_t q$  gives rise to an image charge  $\Delta_b \Delta_t q$  in the bottom dielectric domain, and so on. Here, the prefactors  $\Delta_b$  and  $\Delta_t$  are defined as

$$\Delta_b = \frac{\varepsilon_m - \varepsilon_b}{\varepsilon_m + \varepsilon_b}, \quad \text{and} \quad \Delta_t = \frac{\varepsilon_m - \varepsilon_t}{\varepsilon_m + \varepsilon_t}.$$

These polarization image charges constitute simple, but infinite geometric series that can be taken into account analytically by the far formula of the MMM2D method [3]. This formula gives a simple expression for the interaction energy



**Fig. 1** Image charges generated by dielectric contrasts. The dielectric interfaces are characterized by the planar  $\varepsilon_t-\varepsilon_m$  and  $\varepsilon_m-\varepsilon_b$  boundaries. Polarization leads to an infinite number of image charges along the  $z$ -direction due to multiple reflections. Additionally, periodic boundary conditions apply in  $x$  and  $y$  directions. The *dotted lines* are only provided to visualize the positioning of the image charges

or force between a charge and a periodic array of charges. Since this formula is also at the heart of the ELC method, both MMM2D and ELC can take these additional image charges into account by prefactors to the already present far formula implementations. These prefactors can be conveniently derived from analytic summations of the geometric sums [42, 43].

MMM2D and ELC allow to specify  $\Delta_b$  and  $\Delta_t$  directly instead of specifying the three dielectric constants. By choosing these values as  $\pm 1$ , one can obtain effectively infinite dielectric constants inside or outside. Finally, note that the method of image charges describes only the correct electrostatic potential inside the dielectric slab; therefore, particles need to be constrained to the area between the two dielectric surfaces.

## 4.2 MEMD

The MEMD algorithm (**M**axwell **E**quations **M**olecular **D**ynamics) is a rather uncommon way to compute electrostatic interactions, based on the full electrodynamic Maxwell equations. Of course, simulating with actual realistic electrodynamics is by far too time consuming, but it can be made more effective with some adaptations. Most notably, the speed of light needs to be reduced to become comparable to the other degrees of freedom, that is, the speed of the atoms and molecules in the simulation.

### Algorithm

The algorithm starts with a regular discretization of the Maxwell equations. To overcome the problem of the large value for the light speed, A. Maggs has shown [28] that the speed of light can be tuned to very small values, as long as



it stays well above the speed of the charges. In this case, the full dynamics of the theory (i.e. for the magnetic fields) can be unrealistic while the electrostatic limit is still valid. In MD simulations, the speed of light can be chosen by about an order of magnitude larger than typical particle velocities, which however are small compared to the actual speed of light.

Another very important adaptation is, that this method actually consists of two different combined methods: initially, a solution of the Poisson equation for the system is computed with a numerical relaxation scheme. Afterwards the correct solution can be obtained by only applying temporal updates every time step. The time derivative and some physical arguments as laid out in Ref. [34] lead to the following very simple constraint that can be applied to the propagation of the system:

$$\frac{\partial}{\partial t} \mathbf{D} + \mathbf{j} - \frac{1}{c^2} \nabla \times \mathbf{B} = 0$$

with the electric field  $\mathbf{D}$ , the current  $\mathbf{j}$  and a magnetic field component  $\mathbf{B}$  that is propagating in dual space.

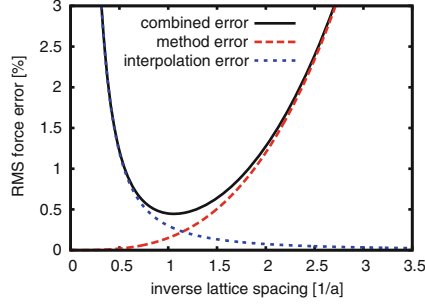
This constraint is now ensured by interpolating the charges (or to be more specific, the electric currents) onto a regular lattice. Then the magnetic fields that are created from the current are propagated on the lattice. From these magnetic-type fields, temporal updates for the electric fields on the lattice can be deduced and backinterpolated to the charges for the force calculation.

This algorithm is not used as widely as many other well known electrostatics algorithms, such as the various particle-mesh approaches [12]. But it comes with some benefits that have become very interesting over the past few years. Since the Maxwell Equations for electrodynamics are intrinsically local and require no global information on the system, one gains two main advantages:

- First, unlike for all Ewald-based algorithms, the parallelization for such a local system of equations is trivial. And the scaling of the algorithm is only dependent on the lattice mesh size and does therefore scale linearly  $\mathcal{O}(N)$  for a fixed particle density. This is a very intriguing feature in a time where massively parallel grid computing and systems in the order of  $10^9$  charges need to be considered.
- Second, a periodic box geometry can be dealt with very naturally by mathematically connecting each boundary to its oppositely placed boundary and propagating the fields accordingly. Another welcomed feature is that because of its locality the method allows for arbitrary changes of the dielectric properties within the system.

## Performance and Precision

As expected from an intrinsically local algorithm, MEMD scales linearly with the mesh size, even to high particle numbers. For homogeneously distributed charges in a dense system (e.g. a molten salt), it outperforms P<sup>3</sup>M for a comparable precision at about 5,000 and more charges. However, the MEMD algorithm cannot be tuned



**Fig. 2** The two error sources of the algorithm. This graph shows two things: for once, the error cannot be tuned to arbitrarily small values, but only to about 1 %. One can also observe that the method error increases for finer lattice spacings and therefore a smaller field propagation speed. On the other hand, the interpolation error increases if the mesh becomes too coarse

to a given precision, although some statements can be made on the systematic errors. The two main systematic errors stem from the speed of light parameter not being infinite (which would mean perfect electrostatic limit) and from the charge interpolation on the lattice.

The first error is of algorithmic origin and is merely the difference between electrodynamics and the static limit. As can be seen e.g. in Ref. [21], this error scales with  $1/c^2$ , where  $c$  is the speed of light. The second error is larger in comparison and diverges for very fine meshes, since only next neighbor interpolation is performed. It scales with  $1/a^3$ , where  $a$  is the lattice spacing. Since the speed of light (i.e. the propagation speed of the magnetic fields) also directly depends on the lattice spacing, these two errors can be combined to find the best suited mesh for the system (see Fig. 2).

In conclusion one can see that for dense systems or at high particle numbers, the MEMD algorithm provides a very flexible and fast alternative method to calculate Coulomb interactions.

### 4.3 The ICC★ Algorithm for Dielectric Interfaces

Taking into account dielectric interfaces of arbitrary shape in coarse grained simulations is a challenging task. The ICC★ algorithm [23, 44] allows to do so with acceptable extra cost and inherent consistency with the desired periodic boundary conditions. At dielectric interfaces the normal component of the electric field has a discontinuity of the following form:

$$\varepsilon_1 \mathbf{E}_1 \cdot \mathbf{n} = \varepsilon_2 \mathbf{E}_2 \cdot \mathbf{n}, \quad (1)$$

where  $\mathbf{n}$  is the normal vector pointing from regions 2 to 1. This discontinuity can also be interpreted as an effective charge distribution on the surface: the induced charge. For the induced charge density the following equation has to hold:

$$\sigma_{\text{ind}} = \frac{\varepsilon_1}{2\pi} \frac{\varepsilon_1 - \varepsilon_2}{\varepsilon_1 + \varepsilon_2} \mathbf{E} \cdot \mathbf{n}. \quad (2)$$

The idea of the ICC★ algorithm is to self-consistently determine this charge distribution in every MD step. The points of a discretized boundary play the role of usual particles, except they are fixed in space. A simple relaxation scheme is applied, where a new guess of the charge density on every boundary element is calculated from

$$\sigma_{\text{ind}}^i = (1 - \lambda) \sigma_{\text{ind}}^{i-1} + \lambda \left( \frac{\varepsilon_1}{2\pi} \frac{\varepsilon_1 - \varepsilon_2}{\varepsilon_1 + \varepsilon_2} \mathbf{E} \cdot \mathbf{n} \right), \quad (3)$$

where  $\lambda$  is a relaxation parameter, that is typically chosen in the range between 0.5 and 0.9 and  $\mathbf{E}$  denotes the electric field caused by all other charges. This iteration usually takes only a few steps, because the position of the charged particles in the system changes only slightly.

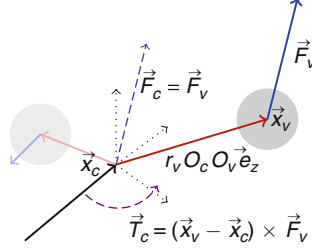
In many cases just one update per MD steps yields sufficient accuracy. In this case, the extra computational cost of taking into account the dielectric boundary forces is only due to the increased number of charges in the system stemming from the surface discretization. In a system with an intrinsic surface charge, i.e. due to dissociation of surface groups, this process is computationally free, because the electrostatic field created by the surface charges needs to be calculated anyways.

To create dielectric interfaces ESPRESSO offers a set of commands that create an almost equidistant discretization of geometric primitives. This allows to create objects of a very complex geometry. We also plan to combine the algorithm with the concept of rigid bodies as described in Sect. 5. This would allow also to study moving dielectric objects, which is of interest e.g. in colloidal electrophoresis.

## 5 Rigid Bodies

We have added the ability to simulate rigid bodies built up from particles to ESPRESSO. This is useful to model clusters of particles that move as a whole, to build extended bodies which interact with a fluid (like the raspberry model for colloidal particles [27]), and to construct “non-sliding” bonds, which, for instance, attach polymer chains to specific spots on the surface of a colloidal particle. Some of these cases have traditionally been handled by constraint solvers or by very stiff bond potentials. However, our rigid bodies have a smaller computational overhead, and allow for larger aggregates.

The simulation of rigid bodies is implemented in ESPRESSO using the concept of virtual sites. These are usual MD particles with the one exception that their position, orientation, and velocity is not obtained from the integration of Newton’s equations of motion. Instead, they are calculated from the position and orientation of another particle in the simulation. This non-virtual particle should be located in the center of



**Fig. 3** Illustration of the rigid body implementation. The position  $\mathbf{x}_v$  of the virtual particle is calculated from the position of the center particle,  $\mathbf{x}_c$ , its current orientation,  $O_c$ , and the virtual particle's relative position in the body frame,  $r_v O_v \mathbf{e}_z$ . A force  $\mathbf{F}_v$  acting on the virtual site induces the force  $\mathbf{F}_c$  and the torque  $\mathbf{T}_c$

mass of the rigid body and should carry the mass and inertial tensor of the complete body. The position and orientation of this center particle are obtained from the integration of Newton's equation, just as any other normal MD particle. The virtual sites, that give the shape and interactions of the rigid body, are placed relative to the position and orientation of the center particle, and forces acting on them are translated back to forces and torques acting on the center particle. Orientation in this case here refers not just to a director, but to a full local three dimensional reference frame, in which the virtual sites can be placed.

The virtual sites making up the rigid body are placed according to the following rules (compare Fig. 3). The position of a virtual site  $\mathbf{x}_v$  is obtained from

$$\mathbf{x}_v = \mathbf{x}_c + r_v O_c O_v \mathbf{e}_z, \quad (4)$$

where  $\mathbf{x}_c$  is the position of the center particle.  $O_c$  is the rotation operator that relates the particle-fixed, co-rotating coordinate system of the center particle to the lab frame and represents the orientation of the rigid body.  $\mathbf{x}_c$  and  $O_c$  are the two quantities that are integrated using Newton's equations of motion. The relative position of the virtual site, is represented by  $O_v$ , which is the rotation operator which rotates the  $\mathbf{e}_z$  unit vector such that it becomes co-aligned with the vector connecting the center particle with the virtual site, and by  $r_v$ , which is the distance between the virtual particle and the center of mass. In the ESPRESSO package, the rotations  $O_c$  and  $O_v$  are represented as quaternions.

Accordingly, the velocity of the virtual site is given by

$$\mathbf{v}_v = \boldsymbol{\omega}_c \times (\mathbf{x}_v - \mathbf{x}_c), \quad (5)$$

where  $\boldsymbol{\omega}_c$  is the rotational velocity of the center particle.

The force on the center of mass due to the force  $\mathbf{F}_v$  acting on a virtual site is given by

$$\mathbf{F}_c = \mathbf{F}_v, \quad (6)$$

so that the total force on the center particle  $j$  is

$$\mathbf{F}_j^{\text{total}} = \sum_{i \text{ virtual site of } j} \mathbf{F}_i. \quad (7)$$

The torque generated on the center of mass by a force  $\mathbf{F}_v$  acting on a virtual site is

$$\mathbf{T}_c = (\mathbf{x}_v - \mathbf{x}_c) \times \mathbf{F}_v, \quad (8)$$

so that the total torque acting on the center particle  $j$  is

$$\mathbf{T}_j^{\text{total}} = \sum_{i \text{ virtual site of } j} (\mathbf{x}_i - \mathbf{x}_j) \times \mathbf{F}_i. \quad (9)$$

In other words, the force acting on the virtual site is copied to the center particle. In addition, the component of the force which is orthogonal to the vector connecting center of mass and virtual site creates a torque on the center of mass.

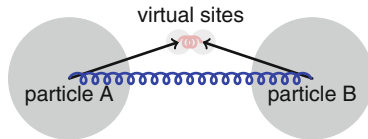
Combining these formulas, the force calculation proceeds as follows:

1. Place all the particles of the rigid body as virtual sites according to the position and orientation of the center of mass particle (Eqs. (4) and (5)).
2. Calculate the forces in the system. This includes forces between parts of the rigid body (i.e., the virtual sites) and the rest of the system.
3. Collect the forces which have accumulated on the virtual sites and convert them to a force and a torque acting on the center of mass particle (Eqs. (6) and (8)).

Using these forces and torques, the Newton's equations can be integrated for the center of mass of the rigid bodies along with those for all the other non-virtual particles in the system.

## 6 Dynamic Bonding

For the study of reaction kinetics and irreversible agglomeration, it is necessary that bonds can be created dynamically during the simulation, for example, when particles come closer than a given bonding distance. For this purpose, a collision detection feature has been introduced in ESPRESSO. Breakable bonds, that would be required for studying reaction kinetics of weaker bonds, are at present not implemented, but planned for a future release. Using dynamic bonds is computationally cheaper than using e.g. reactive potentials [8, 17], since one does not need many body potentials, but it is also less natural in terms of atomistic dynamics. For example, our present method does not allow to differentiate between single or double bonds. However, the main aim of ESPRESSO are coarse-grained simulations, where large macromolecules such as soot particles agglomerate. On the size of these particles, the attractive interactions are very short ranged, but strong, which is modeled well by such a discrete on-off bonding, while it is unclear how



**Fig. 4** Construction of the non-sliding bond. Virtual particles are constructed at the point of collision and connected by a zero length bond. The two colliding particles are also directly bound at collision distance

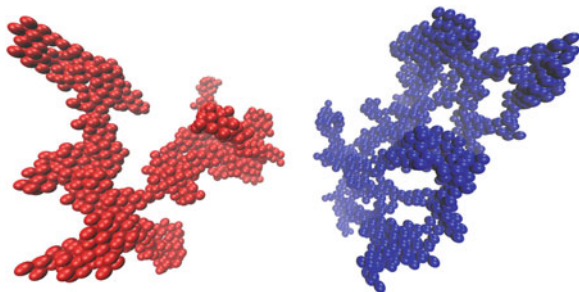
to parametrize e.g. a Tersoff-like potential for macromolecules. Also, our dynamic bonding optionally allows to fix the internal frames of particles against each other, so that the positions of the contacts remains fixed. Again, this makes little sense in terms of atomistic simulations, but is realistic for soot particles, where the sticking is due to permanent deformation of the particles.

When dynamic bonding is turned on, particles that during the simulation come closer than a specified distance get connected by bonds, which at present cannot break again. Therefore, larger and larger clusters are formed. In the simplest case, particles are bonded by a distance-dependent potential like a harmonic or FENE bond. In this case, the particles in the cluster can slide around each other. If the particles do not have a strong repulsive interaction, this might lead to the formation of further bonds within the cluster, resulting in a rather bulky agglomerate. A snapshot of such a cluster can be seen in the left part of Fig. 5. The cluster clearly has a branched structure, which we found to be stable for half a million MD time steps, but the branches are at least three particle diameters wide. For the snapshots, we performed MD simulations of thousand Weeks-Chandler-Andersen (purely repulsive Lennard-Jones) beads in the NVT ensemble, for a total of a million MD steps.

This sliding of connected particles, however, is not desirable in many applications. When, for instance, larger particles collide and stick together due to local surface effects, the particles should always be connected at the particular point where they initially touched. This is achieved by using the rigid body implementation described in Sect. 5: when two particles collide, two virtual sites are created at the point of collision. Each of them is rigidly connected to one of the colliding particles, while these two virtual sites are bound together by a bond potential with an equilibrium distance equal to zero. In addition, a normal bond is formed between the two colliding particles directly (compare Fig. 4). While the bond between the virtual sites prevents the particles from sliding around each other, the bond between the centers of the colliding particle prevents significant motion around the point of collision. The resulting agglomerates (shown in Fig. 5 on the right) have a much more fine and branched structure, with connections as thin as single particles. The depicted picture was stable for also more than 500,000 MD time steps, and initial conditions were the same as for the conventional bonding strategy. Therefore, the finer structure is clearly only due to the non-sliding bonds.

While the current implementation works only on a single CPU, the method is in principle scalable, because at no point information about the complete

**Fig. 5** Snapshots of agglomerates generated by dynamic bonding with harmonic bonds (*left*) and with non-sliding bonds (*right*). The structures formed by the non-sliding bonding are finer and more branched, visible by the high amount of voids



agglomerating cluster is necessary on a single processor, and with time, agglomeration becomes a less and less likely event. We plan to extend the implementation for parallel computing in the near future.

## 7 Lattice Boltzmann

After 20 years of development the Lattice Boltzmann Method (LBM) has proven to be a powerful tool to investigate dynamics in soft matter systems. In coarse grained simulations it allows to replace the solvent by a mesoscopic fluid, whose dynamics are computed on a discrete lattice. On large length and timescales the LBM leads to a hydrodynamic flow field that satisfies the Navier-Stokes equation. The relaxation of fluid degrees of freedom in liquid systems is much faster than the transport of particles. This separation of timescales allows to neglect the microscopic details of the fluid and is the very reason why different mesoscopic solvent models, such as DPD [19], SRD [29] and the LBM produce equivalent results.

Due to restricted amount of space we only state the properties of the implementation in ESPRESSO and suggest to read the review by Dünweg and Ladd [13] as well as other literature [7, 25, 41, 46]. In ESPRESSO we have implemented a three-dimensional LB model, where velocity space is discretized in 19 velocities (D3Q19). This is sufficient for isothermal flow, as present in almost all soft matter applications. The collision step is performed after a transformation of the populations into mode space as this is necessary for a consistent thermalization [1, 38]. This means, that ESPResSo implements the multiple relaxation time (MRT) LB concept [25]. It also allows to use different relaxation times for different modes, thus to adjust the bulk and shear viscosity of the fluid separately.

As long time scales of fluid relaxation and particle transport are well separated, it is possible to execute the update of the LB fluid less often than the MD steps. Typically one lattice update per five MD steps is sufficient, however, this may depend on other parameters of the system. Besides the update rate, the user conveniently specifies the grid resolution, the dynamic viscosity as well as the density in the usual MD unit system. Additionally an external force density can be applied on the fluid, corresponding to pressure gradients or the effect of gravity. Particles are treated as point particles and are coupled to the LB fluid via a friction

term [2] with an adjustable friction constant  $\gamma$ . Note that the particle mobility is not simply the inverse of  $\gamma$ , but due to the feedback from the moving fluid, it also depends on the viscosity  $\eta$  and the lattice constant of the LB lattice. It was shown in [2, 45] that it is well described by

$$\mu = \frac{1}{\gamma} + \frac{1}{g\eta a}. \quad (10)$$

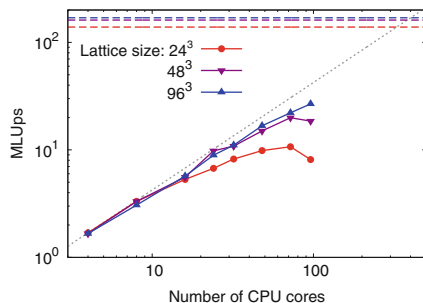
where  $g$  is a numerical factor that was determined to be  $\approx 25$  and  $a$  is the grid spacing. LB reproduces hydrodynamics reasonably well independent of the grid spacing  $a$ . However, there are essentially no hydrodynamic interactions between two particles interacting with the same LB cell, therefore one typically chooses  $a$  of the order of the size of the particles.

Boundaries with zero velocity are incorporated easily into the LBM with the *link bounce back* rule introduced in [47]. Boundaries can also have a nonzero velocity. This is achieved by adding an appropriate bias to the reflected population [41]. Compared to other implementations of the LBM, this boundary method is less sophisticated and there are methods that produce hydrodynamic boundaries with higher accuracy and larger robustness. However as ESPRESSO users usually focus on particle dynamics, this is acceptable. The LBM interface offers simple geometric shapes (spheres, cylinders, rhomboids, ...) that can be combined to surfaces of arbitrary complexity. It is fully consistent with the *constraint* concept, that creates extended objects that act as obstacles or also attractive surfaces for MD particles, thus the geometry can easily be made consistent for particles and the fluid. Also the creation of charged or dielectric boundaries (see Sect. 4.3) is consistent.

The computational effort for the LBM scales linearly with the system size. This allows sizes far beyond the possibilities of traditional methods for hydrodynamic interactions in MD simulations, such as Brownian or Stokesian dynamics with the Oseen- or Rotne-Prager-based mobility matrix. The computational effort is considerable, but the local and lattice-based character of the method allows to optimally exploit the computational possibilities of parallel computing and general-purpose graphics processing unit (GPGPU) programming. As very different programming concepts for massive parallel computers and graphics processing units are necessary, the core implementation is separate for both architectures: The implementation for conventional parallel computer applies the MPI communication interface like the rest of ESPRESSO, while the GPU implementation uses the CUDA programming model and language.

As most available LBM implementations use MPI, we only give a brief description of the GPU implementation here, which uses NVIDIA's CUDA framework [33]. GPU computing is based on the idea of executing the same code massively parallel but with different data, i.e. the Single Instruction Multiple Data (SIMD) paradigm. Both steps of the LBM are optimally suited for this scheme, as identical instructions without conditionals are executed on every node. These instructions are performed in many parallel threads on the Streaming Multiprocessors (SM) of the GPU. The LB fluid resides in the video ram of the GPU. The MD code itself is not altered and still running on the CPU. Only in appropriate intervals the particle positions and





**Fig. 6** Scaling of the LB speed with the number of CPU cores (*solid lines*), compared to a single NVIDIA Tesla C2050 GPU for three different lattice sizes (*dashed lines*). The speed is measured in million lattice node updates per second (MLUps). The *dotted line* marks the ideal linear scaling of the speed with the number of cores

velocities are transferred to the GPU to calculate the interaction with the LB fluid, and the resulting forces are transferred back. Due to the dynamic execution order of the threads, schemes like the double buffering method or using atomic operations are essential to avoid race conditions between concurrently running threads.

A comparison of the computing time of the fluctuating LB fluid on a single NVIDIA Tesla C2050 GPU and an AMD CPU cluster with 1.9 GHz Opteron processors is shown in Fig. 6. The size of the GPU memory of 3 GB limits the maximum lattice size to  $240^3$ . For large lattices and few processors the CPU code scales nearly ideally with the system size, while small lattices and many processors result in a large communication overhead, and therefore longer computational times. The speedup of using a state-of-the-art GPU is however striking: For all lattices that are small enough to fit into the GPU memory, the performance of a single NVIDIA GPU can not be reached with the AMD Opteron CPU cluster, no matter how many CPUs are used. The reason is the communication overhead, which becomes dominating at a performance that is a factor of 5–20 below the single GPU, while using more than 50 cores of recent processors.

## 8 Correlator

Time correlation functions are ubiquitous in statistical mechanics and molecular simulations when dynamical properties of many-body systems are concerned. The velocity autocorrelation function,  $\langle \mathbf{v}(t) \cdot \mathbf{v}(t + \tau) \rangle$  which is used in the Green-Kubo relations is a typical example. In general, time correlation functions are of the form

$$C(\tau) = \langle A(t) \otimes B(t + \tau) \rangle, \quad (11)$$

where  $t$  is time,  $\tau$  is the lag time between the measurements of the (vector) observables  $A$  and  $B$ , and  $\otimes$  is an operator which produces the vector quantity

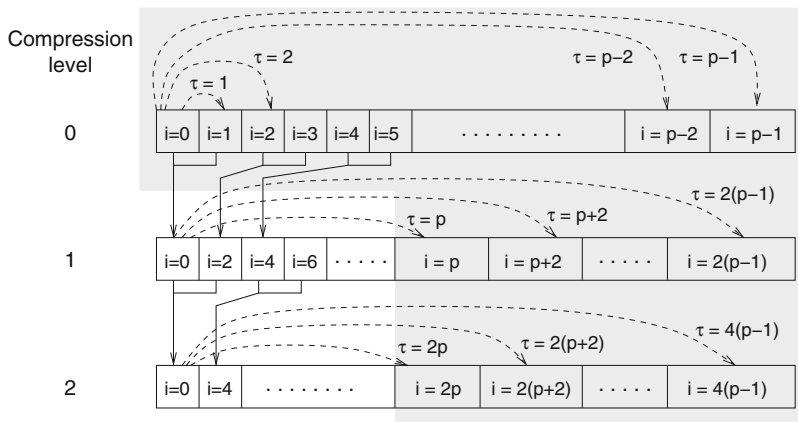
$C$  from  $A$  and  $B$ . The ensemble average  $\langle \cdot \rangle$  is taken over all time origins  $t$ . Correlation functions describing dynamics of large and complex molecules such as polymers span many orders of magnitude, ranging from MD time step up to the total simulation time.

It is trivial to compute a correlation function which spans a short time scale, say less than three decades. In this case storing the configurations in a file and using an external analysis tool is common. However, computation of correlation functions which span many decades raises technical problems, namely: (1) A trivial algorithm which correlates all possible pairs of samples scales quadratically with the maximum correlation time and it easily becomes more time-consuming than the actual simulation; (2) Storing configurations too often (each few time steps) produces significant read/write overhead both at the simulation time and post-processing and requires enormous storage space; (3) Specifically for ESPRESSO, storing configurations can only be done at the script level and produces additional overhead when performed too often. Problem 1 can be resolved by using an efficient correlation algorithm; problems 2 and 3 can be resolved by correlating on the fly, without storing configurations too often and passing control to the scripting interface and back. An apparent drawback is that the post-processing of data is no longer possible and the simulation has to be re-done if a new correlation should be computed. However, repeating the simulation is often less computationally expensive than reading the positions of the particles from disk again, due to the large amounts of data.

Since version 3.1 ESPRESSO features an interface for efficient computation of time correlation functions. First, the user has to define an observable at the script level, which creates the corresponding variable in the kernel and makes it available to the correlator. In the next step, he defines which observables shall be correlated, what is the correlation operation, sampling rate (minimum lag time) and the maximum lag time. Optionally, the correlation estimates (and the respective observables) can be updated automatically without the need for an explicit update call at the script level. Furthermore, parameters of the correlation algorithm can be chosen as described below, which influence the effort needed to compute the correlations as well as the statistical quality of the result. In addition, the correlator can also process data input from the scripting interface or from a file, which enables ESPRESSO to be used as an efficient correlator for data produced by other programs.

### Algorithm: Multiple Tau Correlator

Here we briefly sketch the multiple tau correlator which is implemented in ESPRESSO. For a more detailed description and discussion of the properties of the algorithm concerning parameter values, statistical and systematic errors, we refer the reader to a recent excellent paper by Ramírez et al. [36]. This type of correlator has been used for years in dynamic light scattering [37]. As a special case, its application to the velocity autocorrelation function is described in detail in the textbook of Frenkel and Smit [15]. Nevertheless, despite its obvious advantages, it has been used scarcely by the simulation community.



**Fig. 7** Schematic representation of the correlator algorithm. Values of  $i$  indicate the time of the measurement. *Dashed lines* show values which are correlated and the corresponding lag times. *Solid lines* show values which are merged together in one compression step. At the lowest level, all possible pairs of entries are correlated. At higher levels, only entries separated by more than  $p/2$  values are correlated. This is indicated by the *gray shaded* background

The algorithm of the multiple tau correlator is schematically shown in Fig. 7. Index  $i$  in the figure denotes an observable which was measured at time  $i \tau_{\min}$  where  $\tau_{\min}$  is the sampling interval. We further drop  $\tau_{\min}$  to simplify the notation. The main idea of the multiple tau correlator is to correlate all possible pairs of observables within the relatively short interval of lag times,  $[0 : (p - 1)]$ , which we refer to as compression level 0. For lag times greater than  $p - 1$ , we first apply a compression to the original data in order to produce compression level 1. In the compression step, one value which goes to the higher level is produced from two values in the lower level, e.g. by taking an average of the two or by discarding one of them. In compression level 1, data with lag times  $[p : 2(p - 1)]$  are correlated. Data with lag times  $[2p : 4(p - 1)]$  are correlated in compression level 2 and so on. The number of correlated values at each level is constant but the lag time and the time span of the level increases by a factor of 2. Hence the computational effort increases only logarithmically with the maximum lag time. Thus adding one decade in the lag times increases the effort by a constant amount. The same holds for the amount of memory required to store the compressed history. In the implementation, each compression level is a buffer with  $(p + 1)$  entries. When a new value arrives to the full buffer, two values are pushed to the higher level.

There are several relevant parameters which influence the performance of the correlation algorithm. Their influence on statistical quality of the data has been critically examined in the paper by Ramírez et al. [36]. Here we just state the main points. The following parameters are most important:

*Sampling interval,  $\tau_{\min}$ .* When set to MD time step, it produces a noticeable overhead in computational time. When set to more than ten MD steps, the overhead becomes negligible.

*Maximum lag time of the correlation,  $\tau_{\max}$ .* With a choice of  $p \ll \tau_{\max}/\tau_{\min}$ , it is possible to set  $\tau_{\max}$  to the total simulation time without producing a significant overhead.

*Number of values in 0-th level,  $p$ ,* critically influences the algorithm performance. Small  $p$  makes worse quality of the result, big  $p$  makes the algorithm utterly slow. A reasonable compromise is  $p = 16$  (see also Ref. [36]). Using  $p = \tau_{\max}/\tau_{\min}$ , reduces the algorithm to the trivial correlator.

*Correlation operation.* Defines what the operator in Eq. 11 does with the observables  $A$  and  $B$ . An example could be scalar product, vector product, distance in 3d, ...

*Compression function.* Discarding one of the values and keeping the other is safe for all correlation operations but reduces the statistical quality at long lag times. Taking an average of the values improves statistics but produces a systematic error which can be anything between negligible and disastrous, depending on the physical properties of the correlation function and other parameters of the algorithm.

With this algorithm we have been able to compute on the fly correlations spanning eight decades in the lag times. The overhead in computer time produced by the correlations was up to a factor of 2 when many correlations were computed simultaneously with  $\tau_{\min}$  equal to MD time step and  $p = 16$ . The overhead was caused mainly by frequent updates of the observables based on system configuration and could be largely reduced by taking a longer sampling interval.

## 9 Conclusions and Outlook

To summarize our contribution, we have described some recent additions to our ESPRESSO software package, Version 3.1. Compared to the last version that was published in [26], a number of new features have been added which make ESPRESSO a unique software for coarse-grained simulations. It can now treat systems with varying dielectric media, and has the first implementation of the scalable MEMD algorithm for electrostatic interactions. We have included rigid bodies and dynamic bond formation for studying agglomeration. Also, the hydrodynamic interactions of a coarse-grained particle within an implicit solvent can be handled via the Lattice Boltzmann method.

Our LB solver can use the computing power of GPUs, of which a single card is in most cases sufficient to replace a compute cluster. We plan to make this impressive speedup available for other time-consuming parts of the simulation, namely the electrostatic algorithms P<sup>3</sup>M and ELC. Also, GPUs finally provide enough computational power to allow for solving the Poisson-Boltzmann equations on the fly, which makes it possible to study electrokinetic effects even in complex multi-particle systems on realistic time scales.

Unfortunately, the Tcl scripting language that provides the user interface for ESPRESSO has come to age and is slow when it comes to numerical computation. Although most of the ESPRESSO core is written in C, this affects many users, who implement their own analysis routines or other computations in Tcl. Therefore, we have decided to switch ESPRESSO's user interface from Tcl to Python in one of the upcoming versions of the software. The more modern scripting language Python<sup>12</sup> has drawn a lot of attention to itself over the last decade and is widely embraced by the scientific community.<sup>13</sup> It provides many useful packages, for example for scientific computing, for two- and three dimensional plotting, for statistical analysis, for visualization, etc., which will then be easily available to the ESPRESSO users.

In addition, we and several other groups contributing to ESPRESSO are planning many more improvements in the future, and we hope to attract more users to actively contribute to the code. The current status of the package and the latest code can be found on our web site <http://espressomd.org>, which serves as our collaborative development platform.

**Acknowledgements** We want to thank the more than 20 researchers, that have contributed to the ESPRESSO code so far. Without these contributions and, of course, the feedback of our users, ESPRESSO would never have reached its current status.

## References

1. R. Adhikari, K. Stratford, M.E. Cates, A.J. Wagner, Fluctuating Lattice Boltzmann. *Europhys. Lett.* **71**, 473 (2005)
2. P. Ahlrichs, B. Dünweg, Simulation of a single polymer chain in solution by combining lattice boltzmann and molecular dynamics. *J. Chem. Phys.* **111**, 8225–8239 (1999)
3. A. Arnold, C. Holm, MMM2D: a fast and accurate summation method for electrostatic interactions in 2d slab geometries. *Comput. Phys. Commun.* **148**, 327–348 (2002)
4. A. Arnold, C. Holm, Efficient methods to compute long range interactions for soft matter systems, in *Advanced Computer Simulation Approaches for Soft Matter Sciences II*, ed. by C. Holm, K. Kremer. *Advances in Polymer Sciences*, vol. II (Springer, Berlin, 2005), pp. 59–109
5. A. Arnold, J. de Joannis, C. Holm, Electrostatics in periodic slab geometries I. *J. Chem. Phys.* **117**, 2496–2502 (2002)
6. A. Arnold, B.A. Mann, H. Limbach, C. Holm, *ESPResSo – An Extensible Simulation Package for Research on Soft Matter Systems*, ed. by K. Kremer, V. Macho. *Forschung und wissenschaftliches Rechnen 2003, GWDG-Bericht*, vol. 63 (Gesellschaft für wissenschaftliche Datenverarbeitung mbh, Göttingen, 2004), pp. 43–59
7. R. Benzi, S. Succi, M. Vergassola, The lattice Boltzmann equation: Theory and applications. *Phys. Rep.* **222**, 145–197 (1992)
8. D.W. Brenner, O.A. Shenderova, J.A. Harrison, S.J. Stuart, B. Ni, S.B. Sinnott, A second-generation reactive empirical bond order (REBO) potential energy expression for hydrocarbons. *J. Phys. Condens. Matter* **14**, 783 (2002)

---

<sup>12</sup><http://python.org>

<sup>13</sup><http://www.scipy.org/>

9. J.J. Cerdà, V. Ballenegger, O. Lenz, C. Holm, P3M algorithm for dipolar interactions. *J. Chem. Phys.* **129**, 234104 (2008)
10. M. Deserno, C. Holm, How to mesh up Ewald sums I: a theoretical and numerical comparison of various particle mesh routines. *J. Chem. Phys.* **109**, 7678 (1998)
11. M. Deserno, C. Holm, How to mesh up Ewald sums II: an accurate error estimate for the Particle-Particle-Particle-Mesh algorithm. *J. Chem. Phys.* **109**, 7694 (1998)
12. M. Deserno, C. Holm, S. May, The fraction of condensed counterions around a charged rod: comparison of Poisson-Boltzmann theory and computer simulations. *Macromolecules* **33**, 199–206 (2000)
13. B. Dünweg, A.J.C. Ladd, Lattice Boltzmann simulations of soft matter systems, in *Advanced Computer Simulation Approaches for Soft Matter Sciences III*, ed. by C. Holm, K. Kremer. Advances in Polymer Science, vol. 221 (Springer, Berlin, 2009), pp. 89–166
14. P. Español, P. Warren, Statistical mechanics of dissipative particle dynamics. *Europhys. Lett.* **30**, 191 (1995)
15. D. Frenkel, B. Smit, *Understanding Molecular Simulation*, 2nd edn. (Academic, San Diego, 2002)
16. J.G. Gay, B.J. Berne, Modification of the overlap potential to mimic a linear site-site potential. *J. Chem. Phys.* **74**, 3316–3319 (1981)
17. M. Griebel, J. Hamaekers, F. Heber, A molecular dynamics study on the impact of defects and functionalization on the young modulus of boron-nitride nanotubes. *Comput. Mater. Sci.* **45**, 1097–1103 (2009)
18. B. Hess, C. Kutzner, D. van der Spoel, E. Lindahl, GROMACS 4: algorithms for highly efficient, load-balanced, and scalable molecular simulation. *J. Chem. Theory Comput.* **4**, 435–447 (2008)
19. P.J. Hoogerbrugge, J.M.V.A. Koelman, Simulating microscopic hydrodynamic phenomena with dissipative particle dynamics. *Europhys. Lett.* **19**, 155–160 (1992)
20. W. Humphrey, A. Dalke, K. Schulten, VMD: visual molecular dynamics. *J. Mol. Graph.* **14**, 33–38 (1996)
21. J.D. Jackson, *Classical Electrodynamics*, 3rd edn. (Wiley, New York, 1999)
22. C. Junghans, S. Poblete, A reference implementation of the adaptive resolution scheme in ESPResSo. *Comput. Phys. Commun.* **181**, 1449–1454 (2010)
23. S. Kesselheim, M. Sega, C. Holm, Applying ICC to DNA translocation: effect of dielectric boundaries. *Comput. Phys. Commun.* **182**, 33–35 (2011)
24. A. Kolb, B. Dünweg, Optimized constant pressure stochastic dynamics. *J. Chem. Phys.* **111**, 4453–4459 (1999)
25. P. Lallemand, D. d’Humières, L.S. Luo, R. Rubinstein, Theory of the lattice Boltzmann method: three-dimensional model for linear viscoelastic fluids. *Phys. Rev. E* **67**, 021203 (2003)
26. H.J. Limbach, A. Arnold, B.A. Mann, C. Holm, ESPResSo – an extensible simulation package for research on soft matter systems. *Comput. Phys. Commun.* **174**, 704–727 (2006)
27. V. Lobaskin, B. Dünweg, A new model for simulating colloidal dynamics. *New J. Phys.* **6**, 54 (2004)
28. A.C. Maggs, V. Rosseto, Local simulation algorithms for Coulombic interactions. *Phys. Rev. Lett.* **88**, 196402 (2002)
29. A. Malevanets, R. Kapral, Continuous-velocity lattice-gas model for fluid flow. *Europhys. Lett.* **44**, 552 (1998)
30. N.S. Martys, R.D. Mountain, Velocity verlet algorithm for dissipative-particle-dynamics-based models of suspensions. *Phys. Rev. E* **59**, 3733–3736 (1999)
31. R. Messina, Effect of image forces on polyelectrolyte adsorption at a charged surface. *Phys. Rev. E* **70**, 051802 (2004)
32. MPI Forum, *MPI: A Message-Passing Interface (MPI) Standard Version 1.3* (2008)
33. NVIDIA Corporation, *NVIDIA CUDA reference manual version 3.2* (2010)
34. I. Pasichnyk, B. Dünweg, Coulomb interactions via local dynamics: a molecular-dynamics algorithm. *J. Phys. Condens. Matter* **16**, 3999–4020 (2004)

35. J.C. Phillips, R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R.D. Skeel, L. Kalé, K. Schulten, Scalable molecular dynamics with NAMD. *J. Comput. Chem.* **26**, 1781–1802 (2005)
36. J. Ramirez, S.K. Sukumaran, B. Vorselaars, A.E. Likhtman, Efficient on the fly calculation of time correlation functions in computer simulations. *J. Chem. Phys.* **133**, 154103 (2010)
37. K. Schätzel, M. Drewel, S. Stimac, Photon-correlation measurements at large lag times – improving statistical accuracy. *J. Mod. Opt.* **35**, 711–718 (1988)
38. U.D. Schiller, Thermal fluctuations and boundary conditions in the lattice Boltzmann method. Ph.D. thesis, Johannes Gutenberg-Universität Mainz, Fachbereich 08: Physik, Mathematik und Informatik (2008)
39. E.R. Smith, Electrostatic potentials for thin layers. *Mol. Phys.* **65**, 1089–1104 (1988)
40. T. Soddemann, B. Dünweg, K. Kremer, Dissipative particle dynamics: a useful thermostat for equilibrium and nonequilibrium molecular dynamics simulations. *Phys. Rev. E* **68**, 46702 (2003)
41. S. Succi, *The Lattice Boltzmann Equation for Fluid Dynamics and Beyond* (Oxford University Press, New York, 2001)
42. S. Tyagi, A. Arnold, C. Holm, ICMMD2D: an accurate method to include planar dielectric interfaces via image charge summation. *J. Chem. Phys.* **127**, 154723 (2007)
43. S. Tyagi, A. Arnold, C. Holm, Electrostatic layer correction with image charges: a linear scaling method to treat slab  $2d + h$  systems with dielectric interfaces. *J. Chem. Phys.* **129**, 204102 (2008)
44. C. Tyagi, M. Sützen, M. Sega, M. Barbosa, S. Kantorovich, C. Holm, An iterative, fast, linear-scaling method for computing induced charges on arbitrary dielectric boundaries. *J. Chem. Phys.* **132**, 154112 (2010)
45. O.B. Usta, A.J.C. Ladd, J.E. Butler, Lattice-Boltzmann simulations of the dynamics of polymer solutions in periodic and confined geometries. *J. Chem. Phys.* **122**, 094902 (2005)
46. D.A. Wolf-Gladrow, *Lattice-Gas Cellular Automata and Lattice Boltzmann Models: An Introduction*, vol. 1725 (Springer, New York, 2000)
47. D.P. Ziegler, Boundary conditions for lattice Boltzmann simulations. *J. Stat. Phys.* **71**, 1171–1177 (1993)

Meshfree Methods for Partial Differential Equations VI

Griebel, M.; Schweitzer, M.A. (Eds.)

2013, VIII, 244 p., Hardcover

ISBN: 978-3-642-32978-4