Computer Physics Communications 238 (2019) 66–76

Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

ESPResSo++ 2.0: Advanced methods for multiscale molecular simulation

Horacio V. Guzman [a], Nikita Tretyakov [a], Hideki Kobayashi [a], Aoife C. Fogarty [a], Karsten Kreis [a], Jakub Krajniak [b], Christoph Junghans [c], Kurt Kremer [a], Torsten Stuehn [a,*]

[a] *Max Planck Institute for Polymer Research, Ackermannweg 10, 55128 Mainz, Germany*
[b] *KU Leuven Department of Computer Science, Celestijnenlaan 200A, 3001 Leuven, Belgium*
[c] *Computer, Computational, and Statistical Sciences Division, Los Alamos National Laboratory, Los Alamos, NM 87545, USA*

A R T I C L E I N F O

Article history:
Received 7 July 2018
Received in revised form 18 December 2018
Accepted 21 December 2018
Available online 3 January 2019

Keywords:
Molecular dynamics
Multiscale modeling
Coarse graining
Soft matter
Lattice Boltzmann
Molecular simulations
High performance computer
Computational physics

A B S T R A C T

Molecular simulation is a scientific tool used in many fields including material science and biology. This requires constant development and enhancement of algorithms within molecular simulation software packages. Here, we present computational tools for multiscale modeling developed and implemented within the ESPResSo++ package. These include the latest applications of the adaptive resolution scheme, the hydrodynamic interactions through a lattice Boltzmann solvent coupled to particle-based molecular dynamics, the implementation of the hierarchical strategy for equilibrating long-chained polymer melts and a heterogeneous spatial domain decomposition.

The software design of ESPResSo++ has kept its highly modular C++ kernel with a Python user interface. Moreover, it has been enhanced by automatic scripts that parse configurations from other established packages, providing scientists with the ability to rapidly set up their simulations.

1. Introduction

Molecular simulation methods [1–9] have facilitated the study, exploration and co-design [10] of diverse materials. The functional and dynamic properties of biological and non-biological materials at diverse length and time scales can be simulated with sequential coarse-graining methods [1,11] or with concurrent coarse-graining and atomistic methods [12,13]. Such multiscale methods often involve coarse-graining the atomistic degrees of freedom into effective degrees of freedom representing a collection of atoms, entire monomers or even molecules [1]. An important benefit of multiscale methods is to achieve computational speed-up, which comes from both the coarse-graining method itself and also from optimized algorithms [14].

Within the past decades numerous researchers have contributed to simulations packages like GROMACS [15], LAMMPS [16], NAMD [17], ESPResSo [18], ESPResSo++ [6], among many others. These packages have been devoted to the development of Molecular Dynamics (MD) atomistic and coarse-graining simulations, giving rise to highly parallelizable and flexible codes. The latter, flexibility, is the main design goal of ESPResSo++. Thanks to the flexibility of the ESPResSo++ package, users can easily extend simulation methods in order to meet theoretically and experimentally driven goals, such as in the case of multiscale simulations [7]. In addition, ESPResSo++ combines flexibility and extensibility with the computational requirements of high-performance computing platforms via an MPI-based parallelization. One proof of ESPResSo++'s flexibility for multiscale simulations is the implementation and extension of the Adaptive Resolution Scheme (AdResS) to its Hamiltonian-based version (H-AdResS) and the addition of features such as the flexible spatial atomistic resolution regions approach [19]. ESPResSo++ can be easily used as a molecular dynamic engine and combined with other algorithms, for example, to study complex chemical reactions at the coarse-grained scale [20,21], or to do reverse mapping from a coarse-grained to an atomistic scale using an adaptive resolution approach [22,23].

In this article we have selected two multiscale simulation methods that have been implemented in ESPResSo++, namely, concurrent multiple resolution simulations using AdResS [24–27] and the lattice Boltzmann technique which can be coupled to particle-based simulations [28].

The Adaptive Resolution Scheme has been used for simulations using diverse techniques, ranging from concurrent simulations of classical atomistic and coarse-grained models [12,29–35], to interfacing classical atomistic with the path-integral formulation of quantum models [36,37], as well as interfacing particle-based simulations with continuum mechanics [38,39]. Systems that have

* Corresponding author.
E-mail address: stuehn@mpip-mainz.mpg.de (T. Stuehn).

https://doi.org/10.1016/j.cpc.2018.12.017
0010-4655/© 2019 Elsevier B.V. All rights reserved.

been simulated with the Adaptive Resolution Schemes implementation in the ESPResSo++ package include homogeneous fluids [29–31], biomolecules in solution [32,33,40] and DNA molecules in salt solution [35]. In the present ESPResSo++ release, we come closer to the requirements of adaptive resolution schemes in terms of scalability. Here we also included the Heterogeneous Spatial Domain Decomposition Algorithm (HeSpaDDA), namely a density-aware spatial domain decomposition with moving domain boundaries, for AdResS and H-AdResS simulations, applicable to heterogeneous systems like nucleation, evaporation, and crystal growth.

The second simulation technique introduced in the present release of ESPResSo++ is the Lattice Boltzmann (LB) method, which accounts for hydrodynamic interactions in fluids and can be applied to many problems, ranging from studies of turbulence on the macroscale to soft matter investigations on the microscale. The latter includes hybrid simulations of particle-based systems, *e.g.* colloids or polymers, in a solvent. The virtue of the LB method with respect to the explicit solvent treatment is its methodological locality and, as a consequence, computational efficiency. The LB-module of ESPResSo++ can be used: (i) as a stand-alone method for studies of turbulence and liquids driven by body-forces or (ii) in combination with molecular dynamics providing correct hydrodynamics (in contrast to, *e.g.*, Langevin thermostat).

On top of the highlighted methods for multiscale molecular simulations available in ESPResSo++, this new release also introduces the implementation details of the hierarchical strategy for the equilibration of dense polymer melts. The hierarchical equilibration strategy comprises a recursive coarse-graining algorithm with its corresponding sequential back-mapping [41].

The contents of this publication are focused on the introduction of the new or updated methods and algorithms within the second release of ESPResSo++. The adaptive simulation schemes are described in Section 2 while the Lattice Boltzmann method is presented in Section 3. The hierarchical strategy for the equilibration of polymer melts is described in Section 4. In Section 5 we present the deployment of the HeSpaDDA algorithm. Section 6 provides information on how to contribute to the development of ESPResSo++. Finally, Section 7 reports on the integration of ESPResSo++ with other useful packages.

Regarding the development of ESPResSo++, we want to highlight its user-friendly environment due to the Python interface used for the simulation scripts, and thus higher degrees of freedom to interact with other scientific software parts of the Python community, e.g. `NumPy` [42], `SciPy` [43], `scikit-learn` [44], `Pandas` [45] and `PyEMMA` [46]. Those wishing to get started with the package should visit our webpage [47] or directly go to our GitHub repository [48]. Directions for downloading and building ESPResSo++ are given in both references. Finally for more details of the methods, algorithms or general code of ESPResSo++ please make use of our documentation [49] and previous publication [6].

2. Adaptive resolution simulations

2.1. Introduction

Heterogeneous systems containing a wide range of length- and timescales can be challenging to model using molecular simulation. This is because high-resolution, chemically detailed models are needed to describe certain processes or regions of interest; however, such models are also computationally expensive, and using them to model the entire system can be prohibitive. One approach to tackle this problem is the use of multi-resolution simulation techniques, in which more expensive, typically atomistic, and cheaper, typically coarse-grained models are used within the same simulation box, allowing one to reach longer overall length- and timescales [24,50–53]. In such techniques, a region in space is defined in which molecules are modeled using atomistic detail, while coarse-grained models are used elsewhere (see examples in Fig. 1). The Adaptive Resolution Simulation (AdResS) methodology deals with the coupling between atomistic (AT) and coarse-grained (CG) models [12,24,34,40,53–55]. In this methodology, solvent particles can freely diffuse between AT and CG regions, smoothly changing their resolution as they cross a hybrid or transition region. The AdResS approach can be useful for modeling a wide variety of different systems, such as simple solutes in dilute solution and complex biomolecular systems [19,32,33,35,56–65].

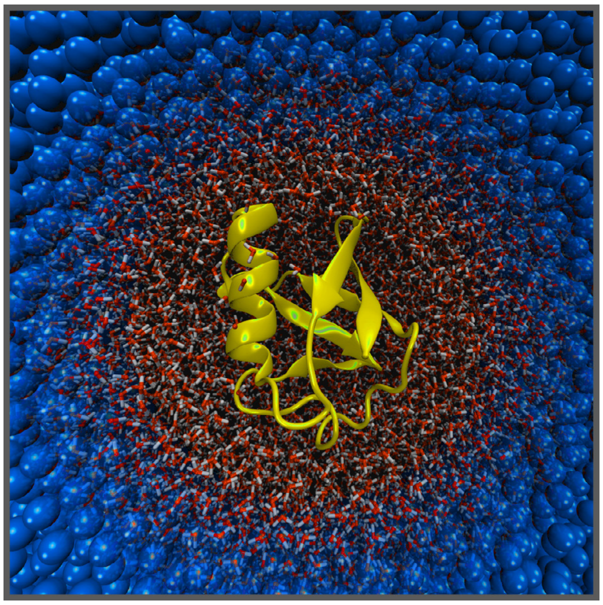


Fig. 1. AdResS simulation of an atomistic protein and its atomistic hydration shell, coupled to a coarse-grained particle reservoir via a transition region [32].

ESPResSo++ provides full support of the AdResS methodology, and the ESPResSo++ AdResS implementation has been used to simulate systems ranging from homogeneous fluids to biomolecules in solution [19,32–34,37,40,65,66]. In the AdResS approach, the coupling of AT and CG models can take place via an interpolation on the level of either forces or energies. Force-interpolation AdResS was included in release 1.0 of ESPResSo++. Energy-interpolation (known as Hamiltonian- or H-AdResS), as well as the latest features are presented in this article.

2.2. Force interpolation

In AdResS, a typically small part of the system, the AT region, is described on the AT level and coupled via a hybrid (HY) transition region to the CG region, where a coarser, computationally more efficient model is used. The interpolation is achieved via a resolution function $\lambda(\mathbf{R}_\alpha)$, a smooth function of the center of mass position $\mathbf{R}_\alpha$ of molecule α. For each molecule, its instantaneous resolution value $\lambda_\alpha = \lambda(\mathbf{R}_\alpha)$ is calculated based on the distance of the molecule from the center of the AT region. It is 1 if the molecule resides within the AT region and smoothly changes via the HY region to 0 in the CG region (see, for example, Ref. [34]).

In the force interpolation scheme, the original AdResS technique [24,54], two different non-bonded force fields are coupled as

$$\mathbf{F}_{\alpha|\beta} = \lambda(\mathbf{R}_\alpha)\lambda(\mathbf{R}_\beta)\mathbf{F}^{AT}_{\alpha|\beta} + \left(1 - \lambda(\mathbf{R}_\alpha)\lambda(\mathbf{R}_\beta)\right)\mathbf{F}^{CG}_{\alpha|\beta}, \quad (1)$$

where $\mathbf{F}_{\alpha|\beta}$ is the total force between the molecules α and β and $\mathbf{F}^{AT}_{\alpha|\beta}$ defines the atomistic force-field, which is decomposed into atomistic forces between the individual atoms of the molecules α and β. Finally, $\mathbf{F}^{CG}_{\alpha|\beta}$ is the CG force between the molecules, typically evaluated between their centers of mass. Note that in addition to the non-bonded interactions usually also intramolecular bond and angle potentials are present. As these are computationally significantly easier to evaluate, they are typically not subject to any interpolation and therefore not further discussed here.

2.3. Potential energy interpolation

Alternatively, the atomistic and the CG models can also be interpolated on the level of potential energies. In the Hamiltonian adaptive resolution simulation approach (H-AdResS) [12,67,68], the Hamiltonian of the overall system is defined as

$$H = \sum_{\alpha} \sum_{i \in \alpha} \frac{\mathbf{p}_{\alpha i}^2}{2m_{\alpha i}} + \sum_{\alpha} \left\{ \lambda(\mathbf{R}_\alpha) V_\alpha^{\mathrm{AT}} + (1 - \lambda(\mathbf{R}_\alpha))\, V_\alpha^{\mathrm{CG}} \right\}, \qquad (2)$$

where $m_{\alpha i}$ and $\mathbf{p}_{\alpha i}$ are, respectively, the mass and the momentum of atom i of molecule α. The single-molecule potentials V_α^{AT} and V_α^{CG} are the sums of all non-bonded intermolecular interaction potentials corresponding to the AT and the CG model acting on molecule α. We again omitted additional intramolecular interactions. To ensure a smooth transition of the molecules between the low- and high-resolution regions, one typically chooses a non-linear interpolation function λ, such as a squared cosine [19,34]. Furthermore, the CG potential is often parametrized to have an excluded volume similar to the AT potential. This prevents overlapping particles and diverging forces when molecules travel between the regions. Alternatively, a force-capping mechanism can be applied if necessary [33]. The intramolecular atomistic degrees of freedom, which are subject to bonded and angular interactions, are either frozen when a molecule resides in the CG region or they are simply integrated everywhere in the full system including the CG region, since they are typically computationally very efficient to evaluate.

The crucial difference between the force-based and the potential energy-based adaptive resolution scheme is an additional force term, dubbed drift force, arising in the forces corresponding to the Hamiltonian in Eq. (2) (for details, see [12]). Both schemes have characteristic advantages and disadvantages and which approach is better suited for implementing the adaptive coupling depends on the application. On the one hand, the force interpolation technique exactly preserves Newton's third law, but it does not allow a Hamiltonian formulation, does not conserve the energy [69], and it therefore requires thermostatting for stable simulations [69–74]. The method can be used for the calculation of expectations in the canonical ensemble, which relies on thermostatting anyway, and even for the estimation of dynamical quantities in the AT region when only the CG region is thermostated to dissipate the excess energy [19,32].

H-AdResS, on the other hand, also allows microcanonical simulations and other approaches that require the existence of a well-defined Hamiltonian. However, it violates Newton's third law in the hybrid region and it features an additional undesired force that must be explicitly taken care of. H-AdResS can be applied when the exact dynamics, i.e. the exact preservation of Newton's third law, is only relevant in the AT region. Importantly, the additional force term that stems from the application of the position-derivative on the resolution function $\lambda(\mathbf{R}_\alpha)$ in Eq. (2) and that reflects the Helmholtz free energy difference between the AT and the CG models is only a minor hindrance. Its effect, pushing particles from one region to the other, can be efficiently canceled out on average using so-called free energy corrections (also see Section 2.4). The H-AdResS methodology is particularly advantageous when one is interested in energy-conserving calculations or other simulations that rely on a Hamiltonian formulation, such as path integral-based techniques (see Section 2.8) [12,36,37].

Both adaptive resolution schemes have been successfully applied for simulations of various different systems, including complex liquids, solutes in dilute solutions, large biomolecules, polymers and quantum systems [19,32,33,35–37,56–65,75–77]. The increase in computational efficiency compared to fully high-resolution simulations varies and depends strongly on the size and properties of the system, the employed AT and CG models, as well as the parallelization and domain-decomposition scheme (also see Section 5). In previous applications, the intermolecular force calculations, the part of the simulation that is actually modified in AdResS, were speeded up by factor of up to $\approx$ 9, while overall reported simulation speed-ups are of the order $\approx$ 2–3 [19,32,33,36,37].

2.4. Free energy corrections and the thermodynamic force

Typical CG models have significantly different pressures compared to the AT reference systems [78–81]. In adaptive resolution simulations, this leads to a pressure gradient between the AT and the CG subsystems, which, in addition to the drift force in H-AdResS, pushes particles across the HY transition region.

Therefore, a correction field must be applied in the HY region to enforce a flat density profile along the direction of resolution change. This compensation force counteracts the pressure gradient and cancels the drift force in H-AdResS. An appropriate correction can be derived, for example, via Kirkwood thermodynamic integration [12,82]. This is known as free energy correction and in particular useful in H-AdResS. An alternative approach, frequently used in the force interpolation method, is to construct a correction force directly from the distorted density profile obtained without any correction and then refine it in an iterative fashion. This approach is known as the thermodynamic force [83].

ESPResSo++ allows the straightforward inclusion of such correction forces and also includes routines that can be used to calculate density profiles, pressures and energies, required for deriving these corrections.

2.5. Self-adjusting adaptive resolution simulations

Many complex systems, such as proteins, membranes and interfaces, do not feature regular spherical or planar geometries. Furthermore, they undergo large-scale conformational changes during simulation. Therefore, recently a scheme was derived that, within the framework of forced-based AdResS, allows AT regions of any arbitrary shape. Additionally, the AT region can change its geometry during the simulation to follow, for example, a folding peptide [19]. This is established by associating several spherical AT regions with many atoms of a macromolecule, such that their overlap defines an envelope around the extended object. When it deforms, this shell adapts accordingly.

This scheme is available in ESPResSo++ within the force interpolation approach and in combination with the thermodynamic force.

2.6. Multiple time stepping in adaptive resolution simulations

Since CG potentials are typically significantly softer than AT force fields, the corresponding equations of motions can be solved using a larger time step. This suggests the use of multiple time stepping (MTS) techniques in adaptive resolution simulations, in which both AT and CG potentials are present simultaneously. A RESPA-based MTS approach [84,85] is now available in ESPResSo++, which enables different time steps for updates of the CG and the AT forces.

2.7. Thermodynamic integration

As explained above in Section 2.3, no global Hamiltonian can be defined in the force-interpolation version of AdResS. Nevertheless, the potential-energy-based Thermodynamic Integration (TI) approach to free energy calculations can be combined with force-interpolation AdResS, as recently shown using simulations of amino acid solvation in ESPResSo++ [66]. This is because AdResS allows the sampling of local configurations which are equivalent to those of fully atomistic simulations.

The TI implementation in ESPResSo++ can also be used to perform standard fully atomistic free energy calculations, such as to calculate solvation free energies or ligand binding affinities, among other examples.

2.8. Path integral-based adaptive resolution simulations

The path integral (PI) formalism can be used in molecular simulations to account for the quantum mechanical delocalization of light nuclei [85,86]. It is frequently used, for example, when modeling hydrogen-rich chemical and biological systems, such as proteins or DNA [87–91]. In the PI methodology, quantum particles are mapped onto classical ring polymers, which represent delocalized wave functions. This renders the PI approach computationally highly expensive (for a detailed introduction see, for example, [85]).

In practice, the quantum mechanical description is often only necessary in a small subregion of the overall simulation. Recently, a PI-based adaptive resolution scheme was developed that allows to include the PI description only locally and to use efficient classical Newtonian mechanics in the rest of the system [36,37]. In this approach, the ring polymers are forced to collapse to classical, point-like particles in the classical region via a mechanism similar to the potential energy interpolation outlined above. In the PI formalism the strength of the spring constants of the ring polymers representing the quantum particles depends on the particles' masses. A light particle has weaker springs between the beads of the ring polymers, resulting in more extended ring polymers, which correspond to more delocalized particles. Similarly, heavy particles have very stiff springs leading to more collapsed ring polymers corresponding to more classical behavior. In the PI-AdResS scheme, one works in normal mode coordinates and interpolates between light and much heavier masses for the degrees of freedom that correspond to the ring polymers' internal modes and ring vibrations, using a resolution function just like in the other adaptive resolution schemes described previously. In that way, the ring polymers behave quantum-mechanically in the high-resolution region and effectively classically in the low-resolution region. Importantly, when the ring polymers are collapsed to point-like particles in the classical, low-resolution subregion, the potential energy calculation becomes significantly more efficient, since one does not need to consider the different contributions from different ring polymer beads anymore.

The method is based on an overall Hamiltonian description and it is consistent with a bottom-up PI quantization procedure. It allows for the calculation of both quantum statistical as well as approximate quantum dynamical quantities in the quantum subregion using ring polymer or centroid molecular dynamics. The methodology is implemented in the ESPResSo++ package and it also makes use of multiple time stepping. It has been used for the calculation of the structural and dynamical properties of quantum-mechanically modeled water as well as liquid hydrogen. For a more technical discussion of the scheme see Refs. [36,37].

3. Lattice Boltzmann

Introduction

The Lattice Boltzmann (LB) method in ESPResSo++ was designed for efficient simulations of phase-separating semidiluted polymer solutions. These solutions are characterized by: (i) a low volume fraction $\phi < 5\%$ of polymeric material with respect to the system's volume V and (ii) long polymer chains that start to overlap. These requirements are satisfied for spatially large systems with only a few very long chains. Since it is not computationally feasible to treat the solvent as explicit particles (their number would be much greater than several millions) we rely on the lattice-based LB methodology in the solvent treatment [92–96]. The polymer chains are modeled by molecular dynamics (MD).

The hybrid LB/MD method is used to study the phase-separation of the polymer solution upon the change of the solvent quality. Under good solvent conditions the chains are extended coils, as the interactions between their monomers and the solvent are favorable. This situation is shown in Fig. 2a. A quench into poor solvent regime (Fig. 2b) initiates the collapse of the polymers and sets out a slow coarsening (Fig. 2c), i.e. agglomeration of individually collapsed chains into multichain polymeric droplets. The quenched system evolves on multiple time and length scales and demonstrates rich dynamical properties.

Implementation details

The LB technique can be viewed as a version of coarse-graining of the solvent fluid on a lattice. At every lattice site $\vec{r}$ and time t the fluid is modeled by a set of single-particle distribution functions or populations $f_i(\vec{r}, t)$. The sites are connected in one LB timestep, Δt_{LB}, by the finite set of velocities, $\vec{c}_i$. Mass and momentum density are given by $\rho = \sum_i f_i$ and $\vec{j}_i = \sum_i f_i \vec{c}_i$, respectively . In ESPResSo++ we employ a popular three-dimensional D3Q19 model with 19 velocity vectors $\vec{c}_i$ [93].

The LB step is divided into collision and streaming parts. At first, the populations collide according to the kinetic rules given by a collision operator. As the operator we use the multiple-relaxation times scheme [97] that allows a straightforward introduction of thermal fluctuations [13] relevant to soft matter research. In the streaming phase the post-collisional populations are propagated to the neighboring sites according to velocity vectors $\vec{c}_i$ and the LB step is finished.

The coupling between the LB fluid and MD particles is done in a dissipative fashion [28] as sketched in Fig. 3a. The force $\vec{F}$ exerted by a solvent onto an MD particle located at position $\vec{R}$ and moving with velocity $\vec{v}$, is given by $\vec{F} = -\zeta[\vec{v} - \vec{u}(\vec{R})] + \vec{F}_{\mathrm{rand}}$, where $\vec{F}_{\mathrm{rand}}$ is the random force due to thermal motion, and the first term is the viscous friction with an amplitude ζ. This term accounts for the velocity of the MD particle $\vec{R}$ with respect to the velocity of the fluid at the position of the particle $\vec{u}(\vec{R})$. The latter is interpolated from the fluid velocities $\vec{u}_i$ at the neighboring lattice sites.

To conserve total momentum of the LB/MD system a counter-force $-\vec{F}$ should act from the MD particle onto the LB fluid. We recast this force in terms of the momentum change $-\vec{F} = \Delta\vec{j}/\Delta t$, where Δt is the MD timestep. The momentum change $\Delta\vec{j}$ of the LB fluid is distributed to the neighboring lattice sites.

A time-costly LB step is done only after several MD steps [28], so we use $\Delta t_{\mathrm{LB}}/\Delta t = 5$ or 10. In this approach, the forces $\vec{F}$ onto MD particles are calculated in every MD step, while the concomitant momentum changes $\Delta\vec{j}$ at the LB sites are accumulated in memory. Firstly, they update the fluid velocities in every MD step: $\vec{u}_i \to \vec{u}_i + \Delta\vec{j}_i/\rho_i$, where ρ_i and $\vec{j}_i$ are the mass and momentum density of the LB fluid at the site i. Secondly, the accumulated momentum changes are applied at the LB collision step via correction of the collision operator. This algorithm is shown in Fig. 3b. For a detailed description of the method we address the reader to Ref. [98].

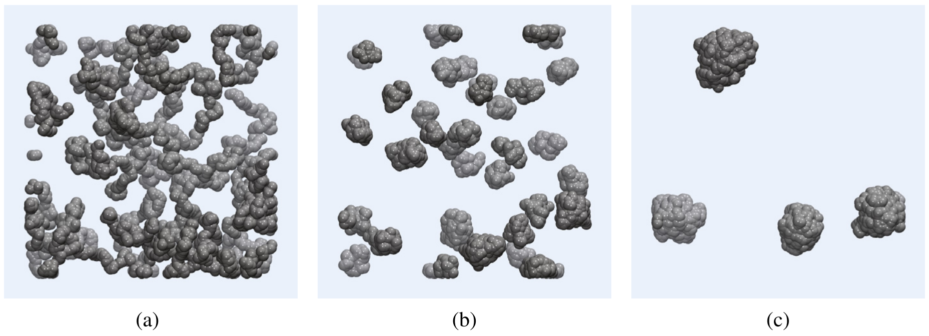


Fig. 2. (a) Good solvent, (b) Poor solvent and (c) Coarsening.

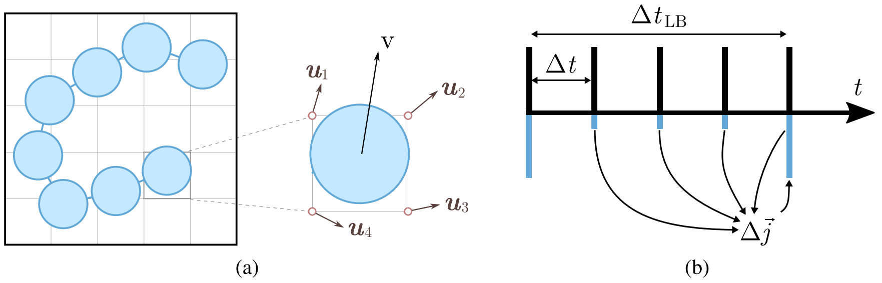


Fig. 3. (a) Schematic representation of the LB-to-MD coupling. The two-dimensional plot is used for simplification. The implemented method is three-dimensional. (b) Time scales separation between the LB and MD time scales.

Efficiency

The LB method of ESPResSo++ employs a regular lattice. Along with the extreme locality of the algorithm (only neighboring sites are connected) it profits from a straightforward but efficient parallelization strategy realized by MPI (message-passing interface). The hybrid LB/MD approach conserves hydrodynamics and is more feasible for large simulations than explicit solvent treatment. Moreover, the timestep separation between MD and LB realized in ESPResSo++ facilitates further speed-up, as a time-intensive LB update is done only every several MD steps.

4. Hierarchical equilibration strategy for polymer melts

Introduction

To study the properties of polymer melts by numerical simulations, we have to prepare equilibrated configurations. However, the relaxation time for polymer melts increases, according to reptation theory, with the third power of the molecular weight [99–101]. In fact, equilibrated configurations of high molecular weight polymer melts cannot be obtained by brute-force calculation in a realistic time, i.e. the CPU time for 1000 polymers consisting of 2000 monomers is roughly estimated as about 4.0×10^6 hours on a single processor (2.2 GHz) on the basis of reptation theory [99–101] and actual measured CPU time per one particle per step. Hence, an effective method for decreasing the equilibration time is required. The hierarchical equilibration strategy pioneered in Ref. [41,102] is a particularly suitable way to do this.

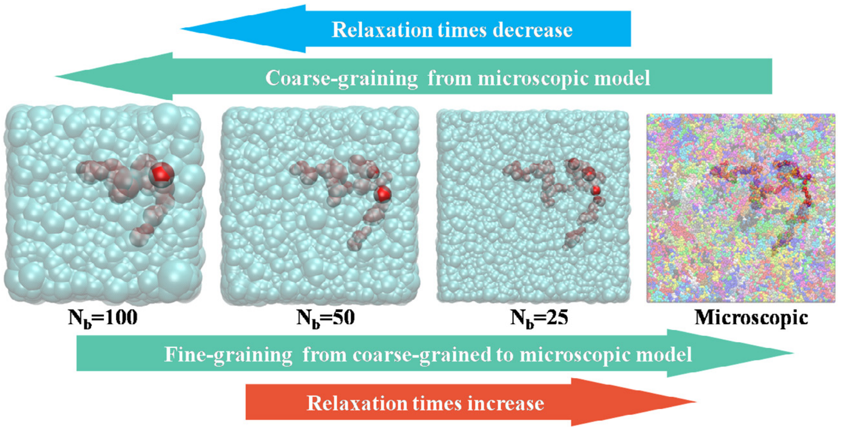
The hierarchical equilibration strategy consists of recursive coarse-graining and sequential back-mapping [41]. At first, a polymer chain, originally consisting of M monomers, is replaced by a coarse-grained (CG) chain consisting of M/N_b softblobs, mapping from each subchain with N_b monomers, represented as the model developed by Vettorel [103]. In this model, the relaxation time does not increase in accordance with the reptation theory but rather Rouse theory, since the CG chains can pass through each other. The degree of freedom of the system is N_b times less than that of the microscopic model. Hence, the relaxation time of CG chain configuration is drastically decreased. After equilibrating a configuration at a very coarse resolution, each CG polymer chain is replaced with a more fine-grained (FG) chain. In this back-mapping procedure, a CG blob is divided into several FG blobs. The center of mass (COM) of the FG blobs coincides with the position of the CG blob's center and is kept fixed during the relaxation of the local conformation of the FG monomers within the CG blob. Consequently, a microscopic equilibrated configuration can be reproduced by sequential back-mapping.

The required functions of this strategy have been implemented into ESPResSo++ (see Figs. 4 and 5).

To demonstrate that the system is equilibrated, we evaluate the mean square internal distance (MSID), defined as $\langle R_{ij}/s_{ij}\rangle$ where R_{ij} is the distance between i and jth monomer on a chain and s_{ij} is the contour length between two monomers. The MSID is the slowest physical property to equilibrate. Hence, other properties already have their equilibrated value, when the MSID reaches the equilibrated form. Auhl's work [104] already obtained the equilibrated MSID form. We use their result as a reference MSID. Fig. 6a shows the MSID equilibrated by the hierarchical strategy for $M = 500$, 1000 and 2000. All of them converge to the same line as Auhl's result [104]. For confirmation, we also present the pair correlation function $g(r)$ for $M = 500$ obtained from brute force calculation and our hierarchical strategy shown in Fig. 6b. They are in quite good agreement with each other. Thus, we conclude that our hierarchical strategy can fully equilibrate the polymer melts system.

Efficiency

The efficiency of the hierarchical strategy for polymer melts can be estimated by a comparison with the brute-force calculation. The CPU time for brute-force calculations τ_{brute} is described as $\tau_{\mathrm{brute}} \sim N \times M^3 \times (M/N_e)\,\tau_{\mathrm{mon}}$, where N_e stands for the number of monomers between entanglement. This value is obtained for the product of the number of monomers, $N \times M$, and the reptation time, $M^2 \times (M/N_e)\tau_{\mathrm{mon}}$, [100].



Relaxation times decrease
Coarse-graining from microscopic model
N_b=100
N_b=50
N_b=25
Microscopic
Fine-graining from coarse-grained to microscopic model
Relaxation times increase

Fig. 4. The schematic representation of hierarchical equilibration strategy. To decrease relaxation time, the microscopic configurations are mapped to a coarse-grained soft blob model. After equilibrating configurations at a very coarse resolution, the microscopic resolution is reproduced by sequential back-mapping.

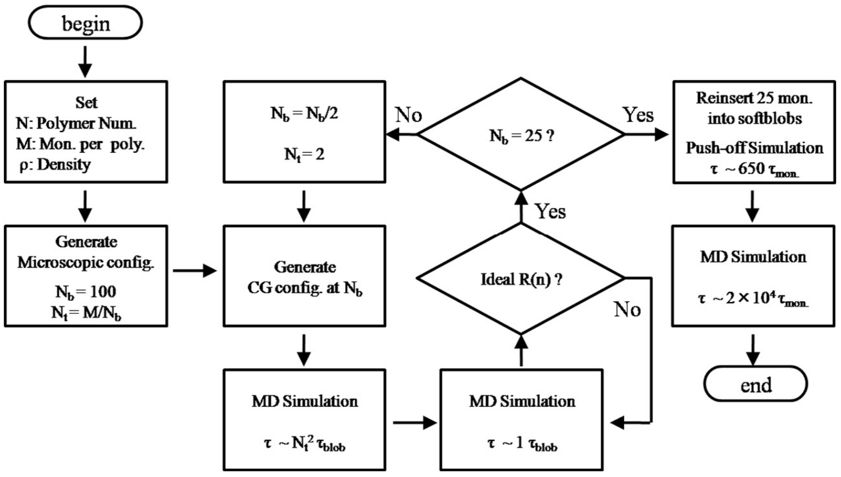


Fig. 5. The flowchart of the hierarchical equilibration strategy.

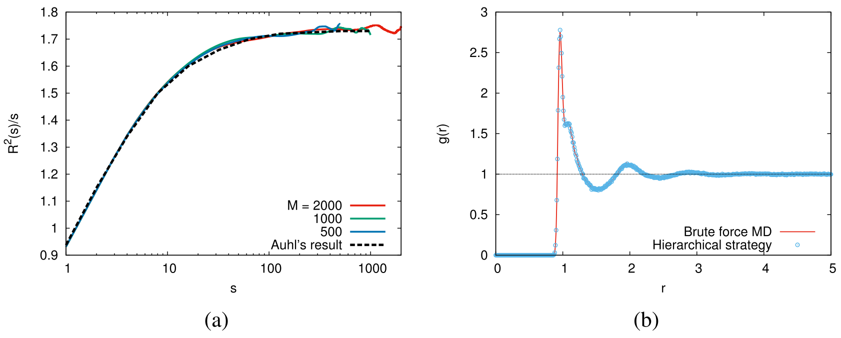


Fig. 6. (a) The mean square internal distances with various polymerization degree M. M = 500 (blue), 1000 (green) and 2000 (red). The dashed line stands for Auhl's result [104]. (b) Pair correlation function g(r) with the polymerization degree M = 500 taken from the hierarchical equilibration strategy (symbol) and the brute force molecular dynamics simulations.

The computational time for the hierarchical strategy τ_{hier} is defined as the summation of the computational time at various resolutions. Thus, we should estimate the relaxation time at various resolutions. The relaxation time τ_{100} for the softblob at $N_b = 100$ can be estimated as $\tau_{100} \sim N \times (M/100)^3 \tau_{\text{blob}}$. The relaxation time τ_{50} for the softblob at $N_b = 50$ is roughly estimated as $\tau_{50} \sim N \times (M/50) \times 64\tau_{\text{blob}}$. In a similar way, the relaxation time τ_{25} for the softblob at $N_b = 25$ can be estimated as $\tau_{25} \sim N \times (M/25) \times 64\tau_{\text{blob}}$. Practically, the calculation time τ_{push} for the push-off procedure and τ_{micro} for the microscopic model have been defined as $650NM\tau_{\text{mon}}$ and about $2NM \times 10^4\tau_{mon}$ respectively [105]. At all MD simulations except the push-off procedure, the increment time dt is defined as the unit time divided by 200. Only at the push-off procedure, dt is defined as the unit time divided by 10^4. Thus, τ_{push} should be multiplied by 50 when we estimate the computational effort. Additionally, the CPU time for a step per particle per processor (2.2 GHz) is about 4.0×10^{-7} s for the microscopic model and about 4.0×10^{-5} s for softblob models. Thus, τ_{blob} should be multiplied by 100 for estimating the efficiency. Hence, the computational time τ_{hier} is estimated as $\tau_{\text{hier}} \sim 100 \times \tau_{100} + 100 \times \tau_{50} + 100 \times \tau_{25} + 50 \times \tau_{\text{push}} + \tau_{\text{micro}}$.

As a consequence, we can estimate the efficiency of the hierarchical strategy by the ratio of τ_{hier} and τ_{brute} represented as

$$\frac{\tau_{\text{hier}}}{\tau_{\text{brute}}} \sim 10^{-4}\frac{N_e}{M} + 4.2884 \times 10^4 \frac{N_e}{M^3}. \tag{3}$$

For example, after substituting $M = 2000$ and $N_e = 100$ to $\tau_{\text{hier}}/\tau_{\text{brute}}$, we obtain the concrete value of the ratio

$$\frac{\tau_{\text{hier}}}{\tau_{\text{brute}}} \approx \frac{1}{1.9 \times 10^3}. \tag{4}$$

Please note that the efficiency of the hierarchical strategy increases with increasing M.

5. Heterogeneous spatial domain decomposition algorithm

Introduction

Simulating heterogeneous molecular systems on supercomputers requires the conception and development of efficient parallelization techniques or Domain Decomposition (DD) schemes [15,16,106]. In the first release of ESPResSo++ the Domain Decomposition scheme was a combination of the Linked-Cell-List algorithm (LCL) with a homogeneous-spatial Domain Decomposition. Such schemes are applied to traditional molecular simulations, for instance in dense homogeneous polymer melt systems [41,105,107]. While traditional molecular simulations are performed with the same resolution for all molecules in the simulation box; in heterogeneous systems [14], we tackle different resolutions (densities). Spatially the simulation box is typically comprised by subregions with different resolutions, namely, for multiscale simulations the coarse-grained and the atomistic/hybrid subregions (see Fig. 7(a)). In terms of computational costs, the most expensive regions are the ones containing atomistic details (higher resolution) followed by the regions using coarse-grained models [30,31] or ideal gas [33] which are significantly cheaper.

The challenges faced by domain decomposition algorithms arise from two factors, namely, the interactions per subdomain and the communication between subdomains. An example of the inter-domains communication constraint is the imbalanced amount of data communicated between the fully atomistic and hybrid regions in comparison to the CG regions (see Fig. 7(c)). In addition, the interactions per domain will be imbalanced if a homogeneous grid decomposes equally the system as shown in Fig. 7(b). This is mainly because the distribution of interactions per subdomain could differ in orders of magnitude depending on the spatial heterogeneous mapping ratio as defined in Eq. (5).

$$R_{SH}^{res} = \frac{N_{HR}^{res}}{N_{LR}^{res}} \tag{5}$$

where N_{HR}^{res} is the number of entities in the high-resolution region that corresponds to one entity in the low-resolution one N_{LR}^{res}. For example mapping the atomistic water molecule to the coarse-grained model can usually result in a $R_{SH}^{res} = 3$ [14] (see also Fig. 8(a)).

To tackle the aforementioned limitations (illustrated in Fig. 7(b) and (c)), the updated release of ESPResSo++ includes an implementation of the Heterogeneous Spatial Domain Decomposition Algorithm, for short HeSpaDDA [14]. In a nutshell, HeSpaDDA will make use of *a priori* knowledge of the system setup, meaning the region that is computationally less expensive. This inherent load-imbalance could come from different resolutions or different densities. The algorithm will then propose a non-uniform domain layout, *i.e.* domains of different size and their distribution amongst compute instances. This can lead to significant speedups for systems of the aforementioned type over standard algorithms, *e.g.* spatial Domain Decomposition [16] or spatial and force based DD [106].

Algorithm description

The proper allocation of processors in heterogeneous molecular simulations is vital for the intrinsic computational scaling and performance of the production run. Moreover, the whole simulation performance is constrained to the initial domain decomposition and hence the correspondence of the number of processors to different resolution regions of the initial given configuration. An example of such a heterogeneous initial domain decomposition is provided by a multiscale simulation of water, where the system is decomposed in the *x*-axis by 8 processors and the homogeneous and HeSpaDDA cases are depicted one next to the other (see Fig. 8(b) and (c)). Also the algorithm flowchart can be found in Fig. 8(d).

Once the processors allocation has been built, an initial cells distribution per subdomain is created to find the maximum number of cells to be used per region. Such a distribution can be done either symmetrically and non-symmetrically. The symmetric cells distribution is triggered if the heterogeneous regions can be decomposed as mirrors within half of the box and the non-symmetric distribution occurs if the heterogeneous regions cannot be mirrored within the simulation box. Within those functions, control statements check if the number of processors to be used are even or odd, as well as the number of cells in each dimension. In case there are still non-distributed cells the symmetric and non-symmetric functions will call a pseudo-random weighted cells distribution for the remaining ones. As a final step the algorithm verifies if the performed DD is scalable, and suggests a possible number of cores to perform the heterogeneous simulation (a Python function named cherry picked processors *cherrypickTotalProcs*). A detailed description of the processors allocation and cells distribution algorithms are provided in a previous article [14] and all python scripts can be found in the referenced code [48].

Implementation in ESPResSo++

The implementation of HeSpaDDA in ESPResSo++ has involved the creation of new data structures, for the number of cells inhomogeneously distributed in each subdomain as depicted in Fig. 8(C). Such data structures are linked to an iterative algorithm that allocates processors to the simulation box according to the resolution of regions *i.e.* fully atomistic, coarse-grained, among others. The processors allocation algorithm flowchart is described in Fig. 8(d).

6. Development workflow

Since the last release [6] we have moved to GitHub hosting and hence from Mercurial to Git https://git-scm.com/ as a version control system. We have also set a new development workflow, *fork-and-branch*, which is commonly used on the GitHub platform [108].

This approach requires two things from the developers: fork the repository and use pull requests to ask for changes in the code base. Basically, every new feature is developed on a branch of the developer's fork repository. Once a feature reaches completion, a merge request is sent to the default branch via GitHub's pull request mechanism. We use *master* branch as the default development branch. The pull request is then reviewed by one of the ESPResSo++ core developers. Usually, minor improvements like e.g. adding tests or documentation are requested from the feature developer. Once all the newly added and existing tests pass, the feature is merged into the master branch.

This whole workflow is supported by continuous integration (CI) tests [109], meaning before the pull request is accepted, it has to fulfill three conditions: properly build, pass all unit-tests and do not decrease code coverage.

The build process is pursued under three Linux distributions: Ubuntu (latest and long-term-support), Fedora, and OpenSUSE.

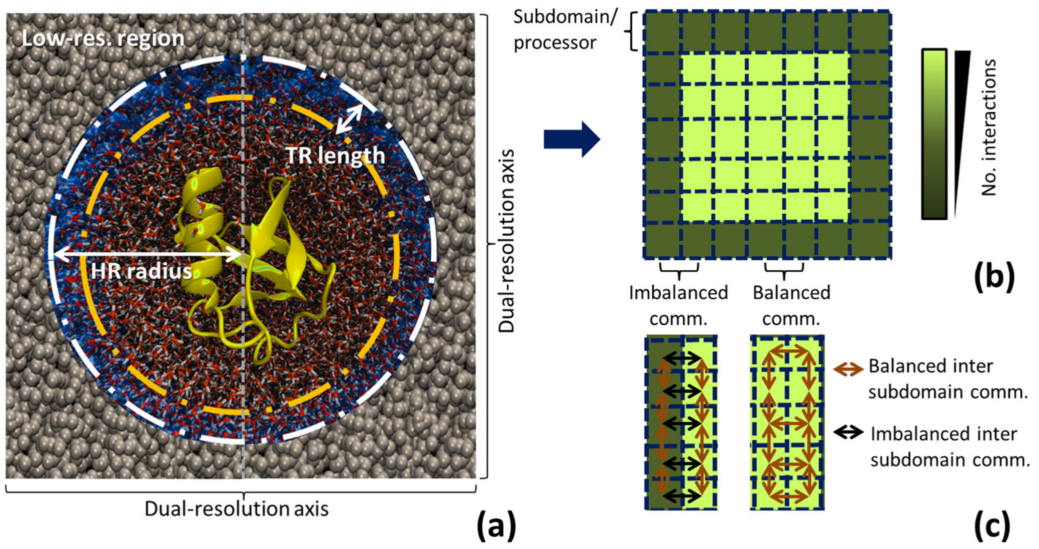


Fig. 7. AdResS simulation of an atomistic protein and its atomistic hydration shell, coupled to a coarse-grained particle reservoir via a transition region [32]. (a) Illustrates all details of the multiscale system subregions. The low-resolution is in gray while the high-resolution is marked by its radius and a white circle. The transition or hybrid region is also marked between the white and orange circles. (b) Scheme of interactions load (computationally exhaustive) as for the subdomains homogeneous distribution of the protein system described above and (c) shows the communication schemes derived from an imbalanced load distribution . (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

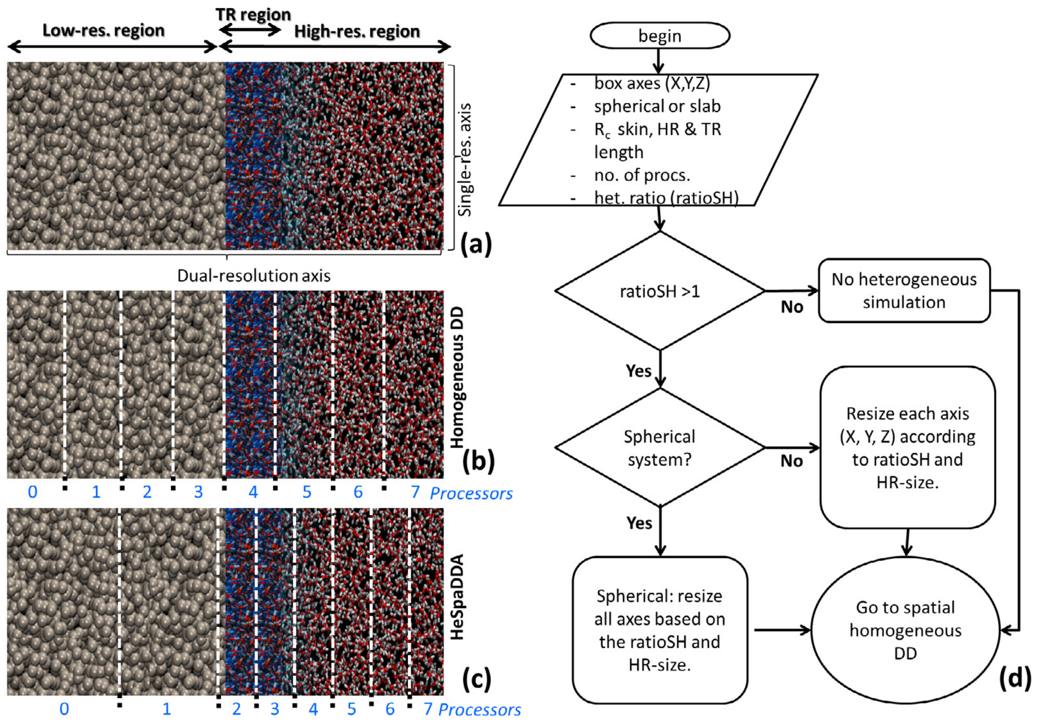

TR region
Low-res. region
High-res. region
Single-res. axis
(a)
Dual-resolution axis
Homogeneous DD
(b)
0 1 2 3 4 5 6 7 Processors
HeSpaDDA
(c)
0 1 2 3 4 5 6 7 Processors
begin
- box axes (X,Y,Z)
- spherical or slab
- R_c skin, HR & TR length
- no. of procs.
- het. ratio (ratioSH)
ratioSH >1
No
No heterogeneous simulation
Yes
Spherical system?
No
Resize each axis (X, Y, Z) according to ratioSH and HR-size.
Yes
Spherical: resize all axes based on the ratioSH and HR-size.
Go to spatial homogeneous DD
(d)

Fig. 8. AdResS simulation of water (a) coarse-grained (low-res. region), Hybrid (TR-region) and the Atomistic regions are illustrated. The two latter make up the high-res. region. (b) depicts the processor allocation for the homogeneous one-dimensional domain decomposition, while (c) for HeSpaDDA. (d) shows the processors allocation flowchart of HeSpaDDA.

Moreover, every change is checked against two different compilers, gcc [110] (versions 4.7, 4.8, 4.9) and clang [111] as well as the internal and external Boost library [112].

As for the tests, we use two types, one that tests particular features, so-called unit-tests. The second set, the regression tests, run against existing reference data. CI gives the developers the advantage that even before the actual code review within the pull request it is easy to see if any changes broke existing tests.

The last condition is the code coverage, which describes the percentage of code lines that were tested with unit-tests. For that, we use the external tool (https://codecov.io/). The current approximate code coverage, calculated by that tool, is 44%.

If all conditions are matched then the pull request can be accepted. In addition, we use CI to build different pieces of documentation including the website, the Doxygen [113] documentation of the code and a pdf of the user's guide. The newly generated documentation is automatically deployed to http://espressopp.github.io/. This documentation is available to the developer to support the usage of the most recent development version. This continuous deployment allows to not waste any time using the possibly outdated documentation of the last release. Moreover, the latest master version is deployed and released to Docker Hub for users who prefer to test ESPResSo++ without building it themselves.

The release versions of ESPResSo++ follow the idea of semantic versioning [114] and the releases workflow ties into it very easily. In a nutshell, fixes and small new features that do not change the application interface can go into "stable" and hence only trigger a minor release, while big refactors that introduce a new feature or break backward compatibility of the application interface go into a major release. After each major release, a *stable* branch is created and git tags on that branch mark the individual minor releases. Bugs and hotfixes are merged (via the same pull request workflow) into the stable branch directly. If necessary, the stable branch is merged into the master branch occasionally to include all fixes into the master branch as well. Here, we present version 2.0 of the package.

Moreover, in the spirit of reproducibility, we have deposited the release on the zenodo.org repository [115].

7. Integration with other packages

The ESPResSo++ can easily be used with other software packages, primarily due to its internal design that allows it to work as nothing more than a Python module. In this way, ESPResSo++ can be called from any other Python code, even from Jupyter [116] during an interactive session. This allows using ESPResSo++ as an educational tool during hands-on sessions.

The recent release of ESPResSo++ brings support for the new file format H5MD [117] that uses HDF5 [118] storage. The H5MD file format is suitable to hold information about the simulation details in a self-descriptive, binary and portable format. Along with the particle positions, with this file format it is possible to store information of box size, particle types, mass, partial charges, velocity, forces and other properties like software version, integrator time-step and a seed of random number generator. Together with the information about particles, the H5MD can store information about the connectivity, in a static or dynamic manner. The static information comprises bonds, angles and dihedrals which are stored at the beginning of the simulation and they would not be updated during the run. Moreover, a dynamic information storage can track the changes in the bonds, angles and dihedrals also during the simulation. This can be very important, e.g., when we use ESPReSo++ to perform chemical reactions [20,21]. By this, it is possible to share not only results of the simulation but also certain details that allow reproducing these results. In addition, HDF5 storage natively supports parallel input–output (I/O) operations which allows performing efficient parallel simulations.

Apart from the H5MD file format, ESPResSo++ has trajectory writers to GROMACS [119], XTC, XYZ and PDB file formats. The simulation details can also be stored in LAMMPS [16] file format. When it comes to reading, the GROMACS topology and trajectory file formats are supported, hence it is possible to run directly a simulation from those input files. This is also true of LAMMPS input files. Because of the variety of file formats that are supported by ESPResSo++, the integration with existing packages is very easy, for example, VOTCA 1.3 [11] package can cooperate with ESPResSo++ out-of-the-box.

8. Examples and documentation

All features and their implementation are described in detail in the ESPResSo++ documentation [49]. Furthermore, ESPResSo++ comes with many example scripts [120] and tutorials that cover all methodologies and demonstrate to the user how to set up various types of simulation systems in practice.

9. Conclusions

In this article we have presented several state-of-the-art multiscale molecular simulation methods that have recently been implemented in the ESPResSo++ package. Specifically, we discussed the latest advancements of adaptive resolution simulations as well as another multiscale scheme for coupling lattice Boltzmann and molecular dynamics techniques. Furthermore, a hierarchical strategy for equilibrating polymer melts and an accompanying domain decomposition scheme, which have also been implemented into ESPResSo++, were presented. The deployment of those methods and algorithms shows the flexibility and extensibility offered by the software package for method development of advanced molecular simulations techniques.

From the software development viewpoint, we are providing scientists from the material and biomolecular scientific communities a computational tool that can be used out-of-the-box from simple Python scripts and allows exploring diverse molecular systems in polymer research, membranes, proteins, crystallization processes, evaporation, among others. This update is also very useful for the prototyping of new theoretical concepts and molecular simulation method development since it includes proven functionalities like: (i) extension of new simulation methods on top of the presented ones, as shown in Section 2, (ii) the development of new algorithms in Sections 4 and 5, or (iii) the combination of multiple scales and multiple methods like the development shown in Section 3.

We have covered many possible applications of the recent methods and algorithms included within the updated ESPResSo++, in particular for the areas of soft matter science, as illustrated within each section of this article. Selected applications of the new ESPResSo++ release have been published and are also referenced within this release communication. We note that these improvements make the simulation of many different systems feasible, from short proteins to huge polymer melts, passing through advanced path integral multiscale systems.

On top of the aspects described above, the ESPResSo++ package is open source (published under the GNU General Public License (GPL) version 3) and hence offers the molecular simulation community the possibility of extending the package and/or adapting methods to their research interests. Moreover a friendly developers environment, including recent developers software tools, mailing lists, repository management, an improved documentation and even parsing of input files from other MD packages, aims to smooth the transition from such packages to ESPResSo++.

Acknowledgments

T. Stuehn, N. Tretyakov and H.V. Guzman acknowledge financial support under the project SFB-TRR146 of the Deutsche Forschungsgemeinschaft. H. Kobayashi acknowledges the European Union's Horizon 2020 research and innovation program under the grant agreement No. 676531 (project E-CAM). A.C. Fogarty acknowledges research funding through the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2 013)/ERC grant agreement no. 340906-MOLPROCOMP. J. Krajniak acknowledges "Strategic Initiative Materials" in Flanders (SIM) under the InterPoCo program and VSC (Flemish Supercomputer Center; Hercules Foundation and the Flemish Government — department EWI). This work has been partially authored by an employee of Los Alamos National Security, LLC, operator of the Los Alamos National Laboratory under Contract No. DE-AC52-06NA25396 with the U.S. Department of Energy. The United States Government retains and the publisher, by accepting this work for publication, acknowledges that the United States Government retains a nonexclusive, paid-up, irrevocable, world-wide license to publish or reproduce this work, or allow others to do so for United States Government purposes. C.J. thanks Los Alamos National Laboratory for a Director's Postdoctoral fellowship supporting the early stage of this work. Assigned: LA-UR-18-24195.

References

[1] K. Kremer, F. Müller-Plathe, Mol. Simul. 28 (8–9) (2002) 729–750.
[2] N. Attig, K. Binder, H. Grubmüller, K. Kremer (Eds.), Computational Soft Matter: From Synthetic Polymers to Proteins, in: NIC Lecture Notes, vol. 23, Forschungszentrum Jülich, 2004.
[3] G.A. Voth (Ed.), Coarse-Graining of Condensed Phase and Biomolecular Systems, CRC Press, Boca Raton, Florida, 2008.
[4] C. Holm, K. Kremer (Eds.), Advanced Computer Simulation Approaches for Soft Matter Science I-III, in: Series: Advances in Polymer Science, vol. 173, 185, 221, Springer, Berlin, 2005-2009.
[5] C. Peter, K. Kremer, Faraday Discuss. 144 (2005) 9–24.
[6] J.D. Halverson, et al., Comput. Phys. Comm. 184 (4) (2013) 1129.
[7] A.J. Liu, et al., Soft Matter 11 (2015) 2326–2332.
[8] D. Rapaport, Comput. Phys. Comm. 62 (2) (1991) 198–216.
[9] D.E. Shaw, et al., Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, 2009, pp. 1–11.
[10] H.V. Guzman, Beilstein J. Nanotechnol. 8 (2017) 968–974.
[11] S.Y. Mashayak, et al., PLoS One 10 (7) (2015) 1–20.
[12] R. Potestio, et al., Phys. Rev. Lett. 110 (2013) 108301.
[13] B. Dünweg, U.D. Schiller, A.J.C. Ladd, Phys. Rev. E 76 (2007) 036704.
[14] H.V. Guzman, C. Junghans, K. Kremer, T. Stuehn, Phys. Rev. E 96 (2017) 053311.
[15] B. Hess, C. Kutzner, D. van der Spoel, E. Lindahl, J. Chem. Theory Comput. 4 (2008) 435–447.
[16] S. Plimpton, J. Comput. Phys. 117 (1995).
[17] J.C. Phillips, et al., J. Comput. Chem. 26 (2005) 1781–1802.
[18] H.J. Limbach, A. Arnold, B.A. Mann, C. Holm, Comput. Phys. Comm. 174(9) (2006) 707–727.
[19] K. Kreis, R. Potestio, K. Kremer, A.C. Fogarty, J. Chem. Theory Comput. 12 (8) (2016) 4067–4081.
[20] P. de Buyl, E. Nies, J. Chem. Phys. 142 (13) (2015) 134102.
[21] J. Krajniak, Z. Zhang, S. Pandiyan, E. Nies, G. Samaey, J. Comput. Chem. 39 (22) (2018) 1764–1778.
[22] J. Krajniak, S. Pandiyan, E. Nies, G. Samaey, J. Chem. Theory Comput. 12 (11) (2016) 5549–5562.
[23] J. Krajniak, Z. Zhang, n.S. Pandiya, E. Nies, G. Samaey, J. Comput. Chem. 39 (11) (2017) 648–664.
[24] M. Praprotnik, L.D. Site, K. Kremer, Annu. Rev. Phys. Chem. 59 (2008) 545.
[25] M. Praprotnik, C. Junghans, L. Delle Site, K. Kremer, Comput. Phys. Comm. 179 (2008) 51–60.
[26] S. Poblete, M. Praprotnik, K. Kremer, L.D. Site, J. Chem. Phys. 132 (11) (2010) 114101.
[27] M. Praprotnik, S. Poblete, K. Kremer, J. Stat. Phys. (2011) 946–966.
[28] P. Ahlrichs, B. Dünweg, J. Chem. Phys. 111 (17) (1999) 8225–8239.
[29] M. Praprotnik, L.D. Site, K. Kremer, J. Chem. Phys. 123 (22) (2005) 224106–224114.
[30] M. Praprotnik, S. Matysiak, L.D. Site, K. Kremer, C. Clementi, J. Phys.: Condens. Matter 19 (29) (2007) 292201.
[31] M. Praprotnik, S. Matysiak, L.D. Site, K. Kremer, C. Clementi, J. Phys.: Condens. Matter 21 (49) (2009) 499801.
[32] A.C. Fogarty, R. Potestio, K. Kremer, J. Chem. Phys. 142 (19) (2015) 195101.
[33] K. Kreis, A.C. Fogarty, K. Kremer, R. Potestio, Eur. Phys. J. Spec. Top. 224 (12) (2015) 2289.
[34] K. Kreis, R. Potestio, J. Chem. Phys. 145 (4) (2016) 044104.
[35] J. Zavadlav, R. Podgornik, M. Praprotnik, J. Chem. Theory Comput. 11 (10) (2015) 5035.
[36] K. Kreis, M.E. Tuckerman, D. Donadio, K. Kremer, R. Potestio, J. Chem. Theory Comput. 12 (7) (2016) 3030–3039.
[37] K. Kreis, K. Kremer, R. Potestio, M.E. Tuckerman, J. Chem. Phys. 147 (24) (2017) 244104.
[38] R. Delgado-Buscalioni, K. Kremer, M. Praprotnik, J. Chem. Phys. 128 (2008) 114110.
[39] R. Delgado-Buscalioni, K. Kremer, M. Praprotnik, J. Chem. Phys. 131 (24) (2009) 244107.
[40] K. Kreis, D. Donadio, K. Kremer, R. Potestio, Europhys. Lett. 108 (3) (2014) 30007.
[41] G. Zhang, L.A. Moreira, T. Stuehn, K.C. Daoulas, K. Kremer, ACS Macro Lett. 3 (2) (2014) 198–203.
[42] T.E. Oliphant, A Guide to NumPy, Trelgol Publishing, 2006.
[43] E. Jones, T.E. Oliphant, P. Peterson, et al., SciPy: Open source scientific tools for Python, 2001-, [Online; accessed <today>].
[44] F. Pedregosa, G. Varoquaux, et al., J. Mach. Learn. Res. 12 (2011) 2825–2830.
[45] W. Mckinney, in: G. Varoquaux, S. van der Walt, J. Millman (Eds.), Proceedings of the 9th Python in Science Conference, Pasadena, CA USA, 2010, pp. 51–56.
[46] M.K. Scherer, et al., J. Chem. Theory Comput. 11 (2015) 5525–5542.
[47] ESPResSo++ Website, ESPResSo++, 2010-2018, http://www.espresso-pp.de.
[48] ESPResSo++ Developers, ESPResSo++ GitHub repository, 2014-2018, https://github.com/espressopp/espressopp.
[49] ESPResSo++ Developers, ESPResSo++ website, 2010-2018, http://www.espresso-pp.de/Documentation/index.html.
[50] W. E, B. Engquist, Z. Huang, Phys. Rev. B 67 (2003) 092101.
[51] D. Roehm, et al., Comput. Phys. Comm. 192 (2015) 138–147.
[52] K. Meier, et al., Angew. Chem. Int. Ed. 52 (10) (2013) 2820–2834.
[53] L. Delle Site, M. Praprotnik, Phys. Rep. 693 (2017) 1.
[54] M. Praprotnik, L. Delle Site, K. Kremer, J. Chem. Phys. 123 (22) (2005) 224106.
[55] R. Potestio, C. Peter, K. Kremer, Entropy, 16 (8) (2014) 4199.
[56] B.P. Lambeth, C. Junghans, K. Kremer, C. Clementi, L. Delle Site, J. Chem. Phys. 133 (22) (2010) 221101.
[57] S. Fritsch, C. Junghans, K. Kremer, J. Chem. Theory Comput. 8 (2) (2012) 398.
[58] D. Mukherji, N.F.A. van der Vegt, K. Kremer, L. Delle Site, J. Chem. Theory Comput. 8 (2) (2012) 375.
[59] D. Mukherji, K. Kremer, Macromolecules 46 (22) (2013) 9158.
[60] H. Wang, C. Hartmann, C. Schütte, L. Delle Site, Phys. Rev. X 3 (1) (2013) 011018.
[61] J. Zavadlav, M.N. Melo, S.J. Marrink, M. Praprotnik, J. Chem. Phys. 140 (5) (2014) 054114.
[62] J.H. Peters, R. Klein, L. Delle Site, Phys. Rev. E 94 (2) (2016) 023309.
[63] J. Sablić, M. Praprotnik, R. Delgado-Buscalioni, Soft Matter 12 (2016) 2416–2439.
[64] P.A. Netz, R. Potestio, K. Kremer, J. Chem. Phys. 145 (23) (2016) 234101, http://dx.doi.org/10.1063/1.4972014.
[65] A.C. Fogarty, R. Potestio, K. Kremer, Proteins: Struct. Funct. Bioinform. 84 (12) (2016) 1902–1913.
[66] R. Fiorentini, K. Kremer, R. Potestio, A.C. Fogarty, J. Chem. Phys. 146 (24) (2017) 244113, http://dx.doi.org/10.1063/1.4989486.
[67] R. Potestio, et al., Phys. Rev. Lett. 111 (6) (2013) 060601.
[68] P. Español, et al., J. Chem. Phys. 142 (6) (2015) 064115.
[69] L. Delle Site, Phys. Rev. E 76 (4) (2007) 047701.
[70] M. Praprotnik, L. Delle Site, K. Kremer, Phys. Rev. E 73 (6) (2006) 066701.
[71] M. Praprotnik, K. Kremer, L.D. Site, Phys. Rev. E 75 (1) (2007).
[72] M. Praprotnik, K. Kremer, L.D. Site, J. Phys. A 40 (15) (2007) F281–F288.
[73] S. Poblete, M. Praprotnik, K. Kremer, L. Delle Site, J. Chem. Phys. 132 (11) (2010) 114101.
[74] M. Praprotnik, S. Poblete, L. Delle Site, K. Kremer, Phys. Rev. Lett. 107 (2011) 099801.
[75] R. Potestio, L. Delle Site, J. Chem. Phys. 136 (5) (2012) 054101.
[76] A.B. Poma, L.D. Site, Phys. Rev. Lett. 104 (2010) 250201.
[77] A.B. Poma, L. Delle Site, Phys. Chem. Chem. Phys. 13 (2011) 10510.
[78] F.H. Stillinger, H. Sakai, S. Torquato, J. Chem. Phys. 117 (1) (2002) 288.
[79] A.A. Louis, J. Phys. Condens. Matter 14 (40) (2002) 9187.
[80] H. Wang, C. Junghans, K. Kremer, Eur. Phys. J. E 28 (2) (2009) 221.
[81] G. D'Adamo, A. Pelissetto, C. Pierleoni, J. Chem. Phys. 138 (23) (2013) 234107.
[82] J.G. Kirkwood, J. Chem. Phys. 3 (5) (1935) 300.
[83] S. Fritsch, S. Poblete, C. Junghans, G. Ciccotti, L.D. Site, K. Kremer, Phys. Rev. Lett. 108 (170602) (2012).
[84] M.E. Tuckerman, B.J. Berne, G.J. Martyna, J. Chem. Phys. 97 (3) (1992) 1990–2001.
[85] M.E. Tuckerman, Statistical Mechanics: Theory and Molecular Simulation, Oxford University Press, New York, 2010.
[86] R. Feynman, A.R. Hibbs, Quantum Mechanics and Path Integrals, McGraw-Hill, New York, 1965.
[87] A. Pérez, M.E. Tuckerman, H.P. Hjalmarson, O.A. Von Lilienfeld, J. Am. Chem. Soc. 132 (33) (2010) 11510–11515.
[88] X.Z. Li, M.I.J. Probert, A. Alavi, A. Michaelides, Phys. Rev. Lett. 104 (6) (2010) 066102.
[89] Y. Nagata, R.E. Pool, E.H.G. Backus, M. Bonn, Phys. Rev. Lett. 109 (22) (2012) 226101.
[90] B. Pamuk, et al., Phys. Rev. Lett. 108 (19) (2012) 193003.
[91] L. Wang, S.D. Fried, S.G. Boxer, T.E. Markland, Proc. Natl. Acad. Sci. USA 111 (52) (2014) 18454–18459.
[92] R. Benzi, S. Succi, M. Vergassola, Phys. Rep. 222 (3) (1992) 145197.
[93] Y.H. Qian, D. D'Humires, P. Lallemand, Europhys. Lett. 17 (6) (1992) 479.
[94] S. Succi, The Lattice Boltzmann Equation for Fluid Dynamics and Beyond, Clarendon Press, Oxford, 2001.
[95] B. Dünweg, A.J. Ladd, in: C. Holm, K. Kremer (Eds.), Advanced Computer Simulation Approaches for Soft Matter Sciences III, in: Advances in Polymer Science, vol. 221, Springer Berlin Heidelberg, 2009, pp. 89–166.
[96] C.K. Aidun, J.R. Clausen, Annu. Rev. Fluid Mech. 42 (1) (2010) 439472.
[97] D. d'Humires, I. Ginzburg, M. Krafczyk, P. Lallemand, L.S. Luo, Philos. Trans.: Math. Phys. Eng. Sci. 360 (1792) (2002) 437–451.
[98] N. Tretyakov, B. Dnweg, Comput. Phys. Comm. 216 (2017) 102108.
[99] P.G. de Gennes, J. Chem. Phys. 55 (2) (1971) 572–579.

[100] M. Doi, S. Edwards, J. Chem. Soc., Faraday Transactions 2: Mol. Chem. Phys. 74 (1978) 1789–1801.
[101] M. Doi, S.F. Edwards, The Theory of Polymer Dynamics, Vol. 73, Oxford university press, 1988.
[102] G. Zhang, T. Stuehn, K.C. Daoulas, K. Kremer, J. Chem. Phys. 142 (2015) 221102.
[103] T. Vettorel, G. Besold, K. Kremer, Soft Matter 6 (10) (2010) 2282–2292.
[104] R. Auhl, R. Everaers, G.S. Grest, K. Kremer, S.J. Plimpton, J. Chem. Phys. 119 (24) (2003) 12718–12728.
[105] L.A. Moreira, G. Zhang, F. Müller, T. Stuehn, K. Kremer, Macromol. Theory Simul. 24 (5) (2015) 419–431.
[106] D. Shaw, J. Comput. Chem. 26 (13) (2005) 1318–1328.
[107] K. Kremer, G.S. Grest, J. Chem. Phys. 92 (8) (1990) 5057–5086.
[108] S. Lowe, Using the Fork-and-Branch Git Workflow, 2015, (last accessed: 05.05.2018).
[109] G. Booch, Object Oriented Design: With Applications, The Benjamin/Cummings Series in Ada and Software Engineering, Benjamin/Cummings Pub., 1991.
[110] R.M. Stallman, Free Software Foundation, 2016.
[111] C. Lattner, The BSD Conference, 2008, pp. 1–2.
[112] B. Karlsson, Beyond the C++ Standard Library: An Introduction to Boost, Pearson Education, 2005.
[113] 2011, Doxygen: Documentation generation system.
[114] P.W. Tom, Semantic versioning v2.0.0.
[115] O. Lenz, et al., espressopp/espressopp: ESPResSo++ 2.0, 2018.
[116] F. Pérez, B.E. Granger, Comput. Sci. Eng. 9 (3) (2007) 21–29.
[117] P. de Buyl, P.H. Colberg, F. Hoefling, Comput. Phys. Comm. 185 (6) (2014) 1546–1553.
[118] The HDF Group, Hierarchical Data Format, version 5, 1997-2016, http://www.hdfgroup.org/HDF5/.
[119] B. Hess, C. Kutzner, D. Van Der Spoel, E. Lindahl, J. Chem. Theory Comput. 4 (3) (2008) 435.
[120] ESPResSo++ Developers, ESPResSo++ Examples repository, 2014-2018, https://github.com/espressopp/espressopp/tree/master/examples.