Eur. Phys. J. Special Topics **227**, 1789–1816 (2019)
© EDP Sciences, Springer-Verlag GmbH Germany, part of Springer Nature, 2019
https://doi.org/10.1140/epjst/e2019-800186-9

THE EUROPEAN PHYSICAL JOURNAL SPECIAL TOPICS

Regular Article

ESPResSo 4.0 – an extensible software package for simulating soft matter systems

Florian Weik[1,a], Rudolf Weeber[1], Kai Szuttor[1], Konrad Breitsprecher[1], Joost de Graaf[2], Michael Kuron[1], Jonas Landsgesell[1], Henri Menke[1,3], David Sean[1], and Christian Holm[1,b]

[1] Institut für Computerphysik, Universität Stuttgart, Allmandring 3, 70569 Stuttgart, Germany
[2] Institute for Theoretical Physics, Center for Extreme Matter and Emergent Phenomena, Utrecht University, Princetonplein 5, 3584 CC Utrecht, The Netherlands
[3] Department of Physics, University of Otago, PO Box 56, Dunedin 9054, New Zealand

Received 15 October 2018 / Received in final form 16 November 2018
Published online 8 March 2019

Abstract. ESPResSo is an extensible simulation package for research on soft matter. This versatile molecular dynamics program was originally developed for coarse-grained simulations of charged systems [H.J. Limbach et al., Comput. Phys. Commun. **174**, 704 (2006)]. The scope of the software has since broadened considerably: ESPResSo can now be used to simulate systems with length scales spanning from the molecular to the colloidal. Examples include, self-propelled particles in active matter, membranes in biological systems, and the aggregation of soot particles in process engineering. ESPResSo also includes solvers for hydrodynamic and electrokinetic problems, both on the continuum and on the explicit particle level. Since our last description of version 3.1 [A. Arnold et al., Meshfree methods for partial differential equations VI, Lect. Notes Comput. Sci. Eng. **89**, 1 (2013)], the software has undergone considerable restructuring. The biggest change is the replacement of the Tcl scripting interface with a much more powerful Python interface. In addition, many new simulation methods have been implemented. In this article, we highlight the changes and improvements made to the interface and code, as well as the new simulation techniques that enable a user of ESPResSo 4.0 to simulate physics that is at the forefront of soft matter research.

1 Introduction

Soft matter [1–3] is an established field of research that concerns itself with systems ranging from the nanometer scale up to tens of micrometers, existing on the interface between physics, chemistry, and biology. These systems are classified as "soft" because the relevant energy scale of interactions is typically comparable to the thermal energy, making them easy to deform through the application of external forces and fields. The broadness of the field is exemplified by the subjects covered

[a] e-mail: fweik@icp.uni-stuttgart.de
[b] e-mail: holm@icp.uni-stuttgart.de

therein: polymer science [4,5], colloidal science [6,7], liquid crystals [8,9], biological physics [10,11], food science [12], material science [13], to name but a few.

Statistical techniques may be employed to describe soft matter systems, as thermal fluctuations and thus Brownian motion play an important role and allow the system to readily explore phase space. The field has thus proven itself to be a fruitful playground for shaping the community's understanding of statistical physics concepts [14,15], as well as emergent non-equilibrium phenomena [16–18]. The delicate interplay of energy and entropy that is found in many colloidal suspensions leads to a plethora of complex structural properties that are difficult to grasp with pen and paper. Indeed, the soft matter community owes a great deal to computer simulations for its current level of understanding.

The use of computers in soft matter traces its root to first molecular-dynamics (MD) simulations of phase separation in a system of hard spheres [19,20]. From these humble beginnings, computer simulation grew out to a new field that bridges experimental and theoretical efforts. These methods came to their own in soft matter, as the typical size and nature of the interactions found therein, lend themselves well to coarse graining. This is a process by which many of the microscopic freedoms that govern a system may be captured in terms of effective interactions on a larger scale. The use of effective interactions enables the simulation of emergent effects using simple models at a fraction of the computation effort that is required to account for the full microscopic details. This has played a decisive role in elucidating, e.g., the guiding principles for phase behavior of complex liquids [21–24], the structure of ferrofluids [25–28], the structure of polyelectrolytes [29–32], and the swelling of hydrogels and ferrogels [33–37].

The MD simulation package ESPResSo is one of the most broad, efficient, and well-suited platforms for the study of coarse-grained models of soft matter. Only LAMMPS[1] [38] and ESPResSo++[2] [39] approach ESPResSo in terms of the scope of their simulation methods and features. Here, we should emphasize that while ESPResSo is capable of simulating molecular fluids, its scope does not cover atomisticly resolved systems. There are a number of better-suited, open-source softwares available for atomistic simulations, such as GROMACS[3] [40–42], NAMD[4] [43–45], and AMBER[5] [46,47], to name just a few of the most commonly used ones.

The aim of this paper is to present the improvements made in ESPResSo 4.0 compared to our previous release ESPResSo 3.1 [48]. We have also taken the opportunity to introduce readers to our software package and encourage them to consider ESPResSo, both for use in soft matter research, and as a platform for method and algorithm development. A strong community is key to maintaining and extending a strong and reliable software for research. We will provide examples of cutting-edge physics that can be simulated using ESPResSo 4.0, which will hopefully create interest in tyring out ESPResSo and potentially joining our thriving user and developer community.

In brief, this paper covers the following subjects. We start by giving a broad introduction to ESPResSo, discussing its history, goals, and community in Section 2. This will set the background to the developments made for our ESPResSo 4.0 release. Next, we focus on the new interface (Sect. 3), the interaction with external packages (Sect. 4), user interaction with the software in terms of visualization and output (Sects. 5 and 6, respectively), and changes to the new code development procedures

[1] `https://lammps.sandia.gov`
[2] `http://www.espresso-pp.de/`
[3] `http://www.gromacs.org`
[4] `https://www.ks.uiuc.edu/Research/namd/`
[5] `http://ambermd.org`

and practices (Sect. 7) that we adopted with ESPResSo 4.0. Finally, we introduce the standout physical methods and models that were recently introduced within ESPResSo: energy minimization routines (Sect. 9), cluster analysis (Sect. 10), chemical reactions (Sect. 11), polarizable molecules (Sect. 12), and self-propelled particles (Sect. 13). We end with a summary and outlook in Section 14.

2 Extensible simulation package for research on soft matter systems

2.1 The idea behind ESPResSo

ESPResSo development began in 2001. Its name is an acronym for Extensible Simulation Package for Research on Soft matter systems, which encapsulates the goal of the software project:

On the one hand, the original development team wished to create a highly parallel molecular dynamics program that could deal efficiently with charged bead-spring models of polymers. At that time, we had learned to master electrostatic interactions in periodic geometries [49,50] and had also developed other fast electrostatic solvers for partially periodic geometries [51–57]. The resulting software saw its first use in the study of polyelectrolyte solutions and charged colloidal suspensions.

On the other hand, the ESPResSo framework was designed to be “extensible”. That is, its core architecture was created to facilitate the testing and integration of new algorithms. ESPResSo has seen many additions since 2001, living up to this extensible character [48]. This includes support for the simulation of hydrodynamic interactions via an efficient GPU implementation of a lattice Boltzmann solver, rigid-body dynamics, a scheme for the study of irreversible agglomeration, and electrostatic solvers that allow for dielectric inclusions or even locally varying dielectric permittivities [48,58–62].

From the start, ESPResSo has a two-fold architecture, which roughly separates the developer from the user and promotes ease of use: (i) Massively parallel simulation routines were implemented in the C programming language using the Message Passing Interface (MPI). These routines scale well up to hundreds of processors on large computing clusters. They also facilitate the addition of new algorithms, even by novice developers. (ii) The simulation core exposes a scripting interface to the user, such that they did not have to directly interact with the core code. This interface was based on the Tcl language–the most suitable choice at that time to provide easy-to-use scripting. This gave the user programmatic control over the simulation protocol, and enabled run-time analysis and visualization in a straightforward manner.

2.2 The motivation for ESPResSo 4.0

Every simulation package has to cope with changing computer architectures and programming models, ensure continued ease of development, and cater to users’ expectations concerning interaction with the software. ESPResSo 4.0 is our answer to the changes in the field of scientific computing that have taken place over the past five years.

Today, the Python language has become the de-facto standard in scientific computing. There exists a vast ecosystem of third-party Python packages for numerical algorithms, visualization, and statistical analysis. This motivated us to port the scripting interface from Tcl to Python for the latest version of ESPResSo, as will be described in Section 3. We have also updated our online visualization to make the most of ESPResSo 4.0's new Python interface, see Section 5. Post-processing and visualization of large data sets is facilitated by the introduction of the HDF5 output format and parallellized output routines, see Section 6. Finally, the ESPResSo core has been converted to the C++ programming language, which has had added benefits both in terms of software development practices and in terms of performance, see Section 7.

To stay at the forefront of the field, it is not only necessary to commit to the user and development experience, it is also critical to keep up with the ever shifting interests of the user base. ESPResSo has done so in the past [48] and continues this effort with the ESPResSo 4.0 release. Our simulation package now covers an even wider range of physical systems of relevance to modern soft matter research. This includes non-equilibrium phenomena in active matter, which ESPResSo 4.0 captures using model self-propelled particles [63–66], see Section 13. Going beyond soft matter, particle-based methods have attracted interest from the engineering community. Specifically, this community utilizes MD to model large-scale physical processes such as soot aggregation [67,68] and air filtration [69], with the aim to improve these processes. Section 10 describes the opportunities for these kinds of studies in our latest release. However, as will become clear shortly, ESPResSo 4.0 is capable of much more.

2.3 The user and developer community

The ESPResSo package was set up as an open software project, committed to open source development under the GNU General Public License (GPL), a practice that we continue to adhere to. Users of ESPResSo are supported through mailing lists[6] and workshops, such as the annual ESPResSo summer schools, which have between 20 and 40 participants. These workshops are often run as CECAM[7] tutorials, jointly funded through grants from the German Research Foundation through the SFB 716 and the cluster of excellence SimTech (EXC 310). The mailing lists and workshops also allow the developer team to announce new features implemented in the code base of ESPResSo to the global research community.

ESPResSo is used by scientists all over the world, which is documented by more than 431 citations on Web of Science since 2006 and 608 citations on Google Scholar, as of this writing. Several research groups have contributed algorithms to the core of ESPResSo, which include: (i) Continuum flow solvers (including ones on GPUs) and various methods to couple them to MD [70–76]. (ii) Advanced algorithms for rare-event sampling and the reconstruction of free-energy landscapes [77,78]. (iii) Monte-Carlo methods [79]. (iv) An interface to Python analysis package MDAnalysis. (v) Extensions to the support for anisotropic particles. (vi) Methods to calculate dipolar interactions on the GPU. Significant effort has been made to accommodate the various sources of these contributions and ensure code quality of ESPResSo 4.0. This includes putting in place an extended test infrastructure and increasing the test coverage. Furthermore, all changes to the software, including those made by the core team, undergo peer review before they are merged, see Section 7.

The latest version of ESPResSo and its documentation can be found on the website http://espressomd.org. Ongoing development is organized through the social code hosting platform GitHub[8], which facilitates contribution handling through its collaborative features.

6 espressomd-users@nongnu.org
7 https://www.cecam.org/
8 https://github.com/espressomd/espresso/

3 The Python interface

In recent years, Python has become increasingly popular in the scientific community as a scripting language, in areas such as machine learning (PyTorch, scikit-learn, Keras) [80–82], symbolic mathematics (SymPy, SAGE) [83,84], data visualization (Matplotlib, Mayavi) [85,86], and numerical mathematics (NumPy, SciPy, pandas) [87,88]. In ESPResSo 4.0, we have therefore decided to switch the interface scripting language from Tcl to Python. This allows for the quick generation of clear and concise scripts that can harness the power of third party numerical tools. The ESPResSo core functionality can be accessed like any other Python module by importing the `espressomd` module. The whole architecture is built around the `system` object which encapsulates the physical state of the high-performance simulation core.

ESPResSo is a particle-based molecular dynamics simulator, i.e., all fundamental operations in the simulation setup revolve around particles. Inspired by Python's list datatype and the NumPy package [87], we introduced slicing operations on the list of particles. Many functions also accept lists and automatically broadcast themselves to the individual list elements. This often reduces the number of lines of code significantly, e.g., placing several particles at random positions in the simulation box can now be done in a single line:

```
import numpy as np

system.part.add(pos=np.random.random((50, 3)) * system.box_l)

print(system.part[:].pos)
```

In Python, functions can be passed to other functions as arguments. ESPResSo 4.0 makes use of this, e.g., for particle selection using a user-specified criterion:

```
charged_particles = system.part.select(lambda p: p.q != 0.0)
```

Because the NumPy package [87] is very popular in numerical computing, including in undergraduate courses, we designed the interface for maximal compatibility. This allows the user to directly pass on values returned from ESPResSo functions to a variety of NumPy routines, e.g., to compute the mean, standard deviation, or histograms.

We further make use of the object-oriented programming capabilities of Python. Instead of commands having an immediate effect, the user instantiates objects of a class and adds these objects to the system instance. For example, when the user creates an object for the electrostatic solver, it is not immediately active, but has to be added to the list of actors first. This new design aims to reduce dependencies between different parts of the user-provided simulation script and eliminate issues related to the order of commands.

4 Integration with external packages

4.1 Using ESPResSo with other Python packages

As we already mentioned earlier, the ESPResSo Python interface was designed with NumPy in mind and most functions return datatypes which are compatible with the routines offered by NumPy. These routines are well-maintained and supported by a large community with numerous online resources and introductory material. One example are various statistics routines. These include mean, variance and median calculations, but also `numpy.histogram()`. As an example, a probability distribution of inter-particle distances can be used to obtain effective potentials via Boltzmann inversion [89]. Furthermore, the confidence interval estimators from `scipy.stats.t.interval()` can be used to control the amount of sampling in a simulation. Random number generation from `numpy.random` and `scipy.stats` can be used, e.g., for setting up a randomized starting configuration. Linear algebra methods from `numpy.linalg` allow for easily calculating the principal axes of and moments of inertia of a set of particles. Lastly, new interpolated bonded and pair potentials in ESPResSo can be added by providing NumPy arrays for the energies and forces.

It is also possible to create Jupyter notebooks [90] using ESPResSo. These interactive worksheets are popular for teaching and provide a great way to nicely present results and the generating simulation code alongside. For example the introductory tutorials for ESPResSo 4.0 are presented in this fashion.

ESPResSo 4.0 provides an interface to the MDAnalysis Python package, which provides a large number of analysis routines for particle-based data. This includes the calculation of radial distribution functions, density profiles, and the persistence length of polymer chains.

4.2 Algorithms for Coulomb and dipolar interactions from the ScaFaCoS library

The treatment of Coulomb and dipolar interactions has always been one of the strong points of ESPResSo. Electrostatic interactions are inherently important in many soft matter systems: e.g. colloidal suspensions are often stabilized by means of electrostatic repulsion, the structure of polyelectrolytes depends on the concentration of charged salt atoms, and electric fields are used for the analysis and separation of particles in electrophoresis applications.

Being long-ranged, electrostatic interactions have to be treated by specialized algorithms in systems with periodic and mixed boundary conditions. ESPResSo already contains implementations of the particle-particle-particle-mesh (P^3M) algorithm [91] for fully periodic systems, as well as the electrostatic layer correction (ELC) [53] and MMM2D [51] scheme for 2D-periodic slit-pore geometries. However, more algorithms have been developed, designed for particular applications. Many of these are available in the ScaFaCoS library [92] (Scalable Fast Coulomb Solvers). ESPResSo 4.0 provides an interface to this library, exposing all provided electrostatic solvers. Moreover, an extension to the ScaFaCoS library adds solvers for dipolar interactions with an $\mathcal{O}(N \log N)$ scaling for periodic, mixed, and open boundary conditions. With ESPResSo 4.0, this version of the library can be used as a solver for magnetostatic interactions [93,94]. The dipolar P^2NFFT solver for open boundaries is of particular relevance to research in magnetic soft matter: properties of such material can depend on the sample shape due to the demagnetization field [95,96].

5 Online visualization

A graphical representation of the system is of great value when it comes to presenting results of a simulation study. Also, it is common practice to visualize intermediate results while setting up the simulation. This saves time, as possible pitfalls related to positions, directions or orientations can be rectified immediately when discovered via visualization. Usually, this is done by writing out trajectories or volumetric data and utilizing external tools like Paraview [97] or VMD [98]. ESPResSo with Python is

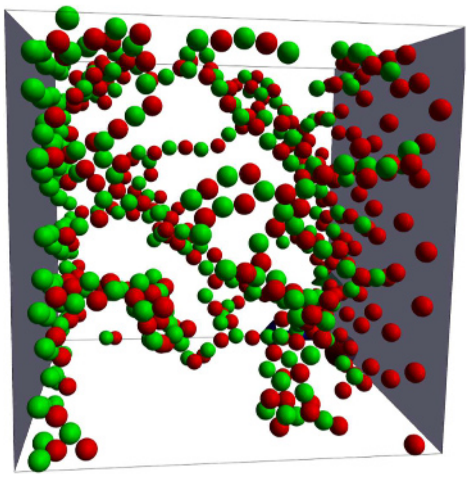


Fig. 1. Example of the OpenGL visualizer of ESPResSo for charged particles in a plate capacitor. Positive (red) and negative (green) ions experience a constant-potential boundary condition.

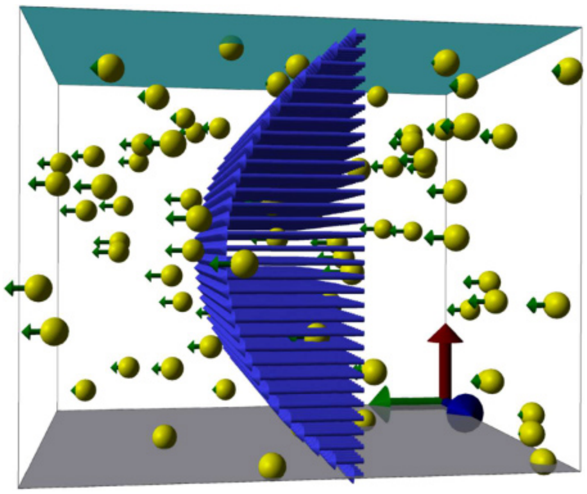


Fig. 2. Example of the OpenGL visualizer of ESPResSo for particles advected in a Poiseuille flow between two walls (Green arrows: particle velocity; blue arrows: fluid velocity).

well suited for this kind of workflow. Furthermore, ESPResSo features two options for visualizing simulations while they are running, i.e., live or on-line visualization. The first one uses Mayavi [86], a Python framework for "3D scientific data visualization and plotting" to visualize particles. Mayavi has a user-friendly graphical interface to manipulate the visual appearance and to interact with the resulting representation. Second, ESPResSo 4.0 comes with a 3D rendering engine that uses PyOpenGL [99, 100]. Both visualizers are not primarily intended to produce print-quality renderings, but rather to give a quick impression of the system setup and the equilibration process. Two exemplary snapshots are shown in Figures 1 and 2; the corresponding Python scripts are available in the distribution package of ESPResSo 4.0.

Our OpenGL visualizer has been crafted specifically for use with ESPResSo and is capable of not only visualizing particles, but also several specific features like constraints, detailed particle properties, the cell system, the domain decomposition across processors, or fluid flows computed with the lattice Boltzmann method [101] (Fig. 1). It has a large number of options to adjust colors, materials, lighting, camera perspective, and many more. Vectorial particle properties can be visualized by 3D arrows on the particles to display forces, velocities, etc. The OpenGL visualizer can also be used for offline visualization to display completed simulations.

A unique feature is the ability to interact with the running simulation: Particles can be queried for their properties in real-time by simply clicking on them in the 3D window. Most of the system information like active interactions, actors and global properties can be viewed in the visualization. In the Python script, the user can assign callback functions to keyboard input, so that all exposed system and particle properties (e.g., the temperature of the Langevin thermostat or the external force on a particle group) can be manipulated in real-time for testing and demonstration purposes. This allows to use ESPResSo also as an educational tool for a broad range of skill levels from basic physics to advanced simulation methods.

6 Parallel output

Writing trajectories of a large set of particles to disk can be a bottleneck of parallel MD simulations. For serial I/O the costs of writing the trajectory include the communication to gather the data on the node that actually performs the write. A second aspect of the performance cost is the amount of data that has to be written to disk. Thus, in ESPResSo 4.0 we added the possibility to write binary files in parallel using the H5MD specification [102] which utilizes the HDF5 file format [103]. H5MD files are intended to be self-contained and therefore enhance the portability of simulation output. This more readily allows one to use a wide class of analysis programs. The implementation is based on a wrapper around the HDF5 library. In typical ESPResSo simulations, the I/O performance is improved by using parallel output for particle numbers greater than approximately 10^4. Writing in this binary format is significantly faster than more traditional ASCII text. The H5MD files written by ESPResSo 4.0 may conveniently be read into Python using the H5py module [104]. Simulation trajectories may also be visualized in VMD [98] by using its H5MD plugin. The H5MD output file contains a field that holds the source script that was used for its creation. Thus the origin of the raw simulation data can always be traced back to the ESPResSo Python script, which helps tremendously in reproducing data.

For convenience, we still support ESPResSo's VTF format which outputs system and particle properties as ASCII text. However, the use of Python as a scripting language opens the door to a large number of alternatives. Many aspects of a simulation, as well as simulation parameters can, for instance, be stored in a structured format using Python's `pickle` module.

7 Software engineering and testing

As outlined in the previous section, the scripting interface is powered by a high-performance simulation core. The core of previous versions of ESPResSo was implemented in the C programming language. It was only natural to also upgrade the core in a manner that mirrors the new object-oriented approach of the interface. Therefore, ESPResSo 4.0 has switched from C to C++. C++ is a modern object-oriented programming language that allows for a more compact, readable code style. Just like Python, C++ benefits from an active community and a rich ecosystem of third-party support libraries, e.g., the Boost libraries [105]. Thus, functionality that was previously implemented in the simulation core can now be provided externally. This reduces the code complexity and consequently the maintenance effort.

ESPResSo is designed with high-performance computing in mind. On high-performance compute clusters, the environment is often times very heterogeneous. For example, external libraries provided by different operating systems can vary noticeably, and only very recent C++ compilers fully support modern standards like C++11. To ensure that ESPResSo can be built and run on a large variety of different setups, it turned out to be necessary to comprehensively test it on different combinations of operating systems and compiler versions. ESPResSo has always included a suite of test cases that could automatically run small simulations to formally verify the physical correctness of the implementation by comparing to established results. In ESPResSo 4.0, not only have these tests been extended to cover more code, but also to specifically test the correctness of certain core aspects, a method called unit testing.

7.1 Improvements in the simulation core

The ESPResSo code base has been developed continuously for more than 15 years by a host of contributors of various backgrounds and styles. Thanks to the substantial overlap of language features between C and C++, it was possible to switch the language with minimal effort. Even though C and C++ share a common heritage, the programming paradigms are radically different and we are gradually transforming the old C code into more modern C++ code.

New language and library features of C++, especially ones which became available with the C++11 standard, are employed for more concise and expressive design of our software. ESPResSo is continuously used in day-to-day research activities and mainly developed by interested domain researchers; hence a gradual modernization strategy was necessary. Among the many algorithms whose code has been refactored, one central method is the traversal of particle pairs used for the evaluation of short-range interactions. In ESPResSo 4.0, the traversal has been separated from the kernel that evaluates the interactions for each particle pair. Repetition of the complicated routine for the pair energy, virial, and force calculation in the three types of particle cell systems supported by ESPResSo is hence avoided. This has been achieved by replacing several loops over the particles by a single function that takes the kernels as template parameters. This allows us to test key routines in isolation, independent of any interaction potentials. Added benefits to this decoupling are, improved performance and the ease by which new methods may be introduced by (future) collaborators.

7.2 Testing

ESPResSo contains a test suite to verify the correctness of the code. Hitherto, these tests were run by the maintainers before accepting code contributions, but features without tests were not covered. A formal measurement of the code coverage was introduced because it proved hard to detect which features are untested. In preparation for this release, we systematically improved coverage of previously untested areas of code. It is our experience that adding tests to existing code not only ensures correctness, but also improves the user experience by streamlining interfaces and avoiding inconsistencies and confusion.

For example, ESPResSo's GPU and CPU lattice Boltzmann (LB) implementations had few systematic tests. However, this is an important method for many users of ESPResSo, and central for the correctness of many soft matter simulations [106]. We therefore introduced multiple new tests in this area that compare the flow fields computed by the LB method, as well as its coupling to particles, against stationary and time-dependent analytical solutions and reference data. These tests include direct verification of momentum and mass conservation, as well as validation of the statistical properties of the fluctuating variant of the LB and particle coupling. This helped us to improve the existing code as well as its interface.

In addition to the improved integration tests, this release includes some unit tests. Unit tests help identify issues on the level of individual functions which make up the algorithms and give more precise information where an issue occurred. This further allows changing the building blocks of an algorithm or their reuse across different simulation methods.

7.3 Continuous integration

As mentioned earlier, the development of ESPResSo is organized through GitHub. Contributors can submit patches to fix a bug or introduce new features by means of a pull request. Whenever such a pull request is filed, the aforementioned set of unit and integration tests is automatically run and the outcome is communicated to GitHub where it is displayed in an informative fashion. This method of automatically running the test suite for changes is called continuous integration. Many online services exist which are free to use for open source projects like ESPResSo. Unfortunately, their free service plans are too limited for the vast variety of setups we need to test. Therefore we mirror the ESPResSo repository to a private instance of GitLab[9]. This instance manages a fleet of dedicated in-house computers and distributes jobs for each configuration to test. On top of the automated testing, other developers are tasked with peer review of the patches to ensure a consistent integration into the code base.

We also try to cover as many combinations of operating systems, compilers, Python versions, and with and without GPU processors as possible. Our use of the Clang compiler [107] is particularly noteworthy as it is able to perform static code analysis to detect a large number of common programming errors and it is able to generate code to automatically recognize undefined behavior during runtime, thus often detecting bugs that are hard to discover for a human software developer.

8 External fields

ESPResSo 4.0 implements a new extensible framework for coupling particles to external fields, i.e., fields that are not caused by particle interactions. An “external field” is composed of a field source and a coupling, which can be combined as needed. An important use case for this feature are particles advected in a prescribed flow field under the assumption that they do not back-couple to that flow field. An example are soot particles on the nanometer scale in combustion exhausts [68,108]. Here the field source would be a flow field specified on a regular grid which is interpolated to the particle positions. The coupling term would be a viscous friction, combining the velocity difference between the particle and the flow field and a friction constant to a force. Further use cases for external fields are complex boundary shapes as they are found in super-capacitor electrodes as for example described by [109]. In this application, electrodes with non-trivial geometry are considered. The electric field caused

[9] GitLab is a free, self-hosted platform similar to GitHub, see `https://gitlab.com`.

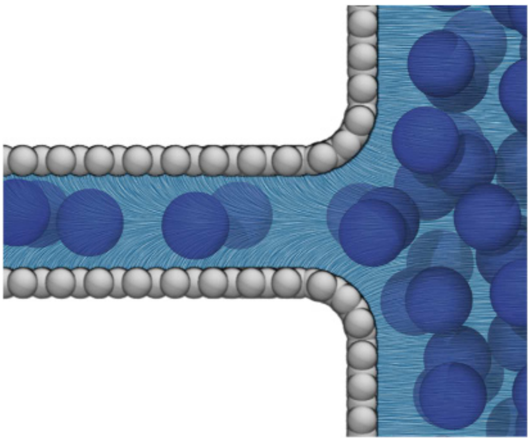


Fig. 3. Electric Field lines in a slit pore filled with an ionic liquid.

by the electrodes is pre-computed by solving Poisson's equation for the system, and the resulting field is then superimposed on the inter-particles forces in the MD simulations as an interpolated external electric field. A snapshot of particles and electric field lines in such a system is shown in Figure 3. The feature can also be used for Lennard–Jones particles in front of convex surfaces. Such a particle can interact with an extended part of the surface. This cannot be described by ESPResSo's constraint feature, which only accounts for an interaction along the surface normal vector.

Let us now look at the two components of external fields in detail. The field source is described as a global scalar or vectorial field. Either, user-provided data are interpolated from a regular grid onto the particle positions, a constant value throughout the full domain is used, or an affine map A according to

$$\boldsymbol{u}(\boldsymbol{r}) = A\boldsymbol{r} + \boldsymbol{b}, \tag{1}$$

with a user-provided matrix A and shift b is evaluated.

The coupling maps a particle property and the value of the field source at the particle position to a force or potential. In the latter case the force is calculated by invoking the coupling with the gradient of the field. For example a gravitational field can be defined as a constant scalar field source coupling to the particle mass.

Presently there are several possibilities for the particle coupling, including but not limited to a particle's scalar mass or charge, or the particle's velocity vector $\boldsymbol{v}$ using a viscous coupling, e.g.,

$$\boldsymbol{F}_i = -\gamma(\boldsymbol{v}_i - \boldsymbol{u}). \tag{2}$$

As a simple example we consider a two-dimensional system comprising particles with excluded-volume interactions in a square simulation box of side length 2π. We combine interpolated flow field data with a viscous coupling. The flow field is a discretized static Taylor–Green vortex, specified by

$$\begin{aligned} u_x &= \cos(x)\sin(y), \\ u_y &= -\sin(x)\cos(y), \end{aligned} \tag{3}$$

where $\boldsymbol{u}$ is the flow velocity. It has the vorticity

$$\omega \equiv \nabla \times \boldsymbol{u} = 2\left|\cos(x)\cos(y)\right|. \tag{4}$$

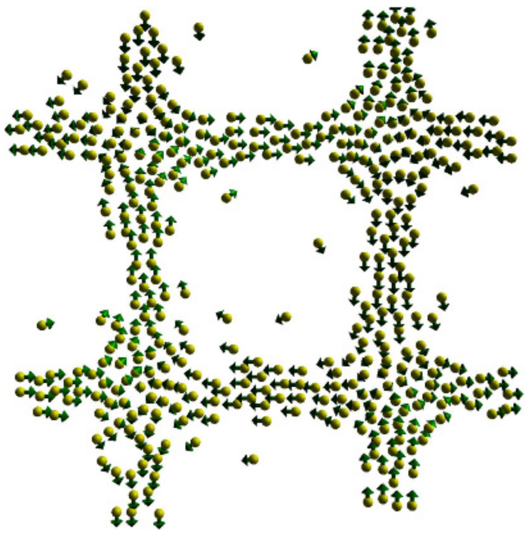


Fig. 4. Example for the use of external fields in ESPResSo 4.0. We show the stationary state of particles in a static Taylor–Green vortex flow. The green arrows denote the particle velocity and direction.

In Figure 4 we show a snapshot of a simulation where the particles were initially placed at random positions distributed equally over the simulation volume. They are driven out of the areas with high vorticity ω by inertial forces and accumulate elsewhere. A complete simulation script can be found in the supplementary information.

9 Energy minimization

Since bulk molecular dynamics are typically set up with random particle positions, initial configurations often have a lot of overlap between neighboring particles. This leads to problems with numerical stability due to very high energies. To remove this overlap there are two common methods. Limiting the forces between particles to a maximum value and performing an energy minimization step after the initial setup. While it was possible to cap the the forces in previous version of ESPResSo, now also energy minimization by the steepest descent method is available. It allows relaxing the position as well as the orientation of the particles by running the following iterative scheme:

$$\Delta_i^{x,y,z} = \operatorname{sgn}(F_i^{x,y,z}) \min(\gamma F_i^{x,y,z}, \Delta_{\max}) \tag{5}$$
$$x_i^{x,y,z} = x_i^{x,y,z} + \Delta_i^{x,y,z}, \tag{6}$$

while $\max_i |F_i| \geq F_{\max}$. Here the $x_i^{x,y,z}$ and $F_i^{x,y,z}$ are the components of the position and force of the ith particle. That is the update rule and limits are applied by component. γ and $F_{\max}$ are relaxation parameters provided by the user. The orientation of the particle is relaxed in a similar fashion. Steepest descent energy minimization is available as an alternative to the integrator in ESPResSo.

In soft matter simulations the solvent is often treated on a coarse-grained level using a lattice Boltzmann fluid, which interacts with the particles of the system via so-called point coupling [106]. This type of coupling limits the maximal coupling strength for one particle which can lead to unrealistic transport properties for larger objects

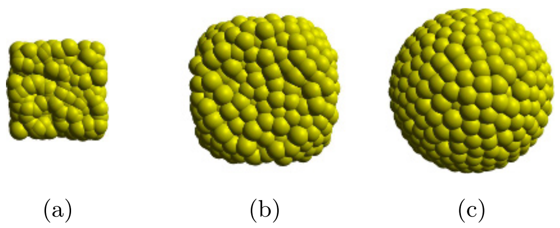


Fig. 5. (a) Initial configuration of the particles filling the raspberry. (b) Configuration after 100 steps of energy relaxation. (c) Configuration after 1000 steps.

like colloids. One way to overcome this problem is to use so-called raspberry models [110], rigid bodies of multiple particles, which couple to the fluid in multiple points to reduce interpolation artifacts and can achieve higher friction with the fluid. This can help to obtain better transport coefficients [63,64]. To set up such a raspberry particle, its volume is filled with particles in a homogeneous manner. A common way to do this is to put the particles into a spherical constraint, give them a repulsive interaction and relax the energy. This can now be easily implemented in ESPResSo. As an example, Figure 5 shows the initial and relaxed configuration of 1400 particles in a spherical constraint with a soft-core Weeks–Chandler–Andersen interaction. Initially the particles are placed randomly in a cube contained by the sphere, then the energy minimization is performed. For the simulation script and detailed parameters please refer to the supplementary information.

10 Cluster analysis

An analysis of the spatial clustering of particles is an important tool in many fields of soft matter science. Some examples include the nucleation of crystals [111,112], clustering of magnetic nanoparticles in magnetic soft matter systems [113–116], and the irreversible agglomeration of soot particles in combustion processes [67,68,117]. Clusters are typically defined by means of a criterion for a given pair of particles which takes the form of an equivalence relation. That is, if particles a and b are "neighbors" and particles b and c are "neighbors", all three particles belong to a cluster. The pair criterion is chosen based on the application. For crystallization studies, a local bond order parameter [118] above a certain threshold on both particles is a common choice. For magnetic systems, a combination of distance and either dipole configuration or pair energies are used [114,115]. For the agglomeration of soot particles, the bonds created by the collision algorithms are used [68].

While it is possible to perform cluster analysis in the post-processing state of a simulation, this requires the storage of a significant number of snapshots of the simulation. Due to the potentially large amount of storage space this might take, as well as the extra time it takes to write the snapshot to disk, this is often not a good approach. ESPResSo 4.0 therefore introduces an online cluster analysis which can be run from within the simulation. The implementation is loosely based on the Hoshen–Kopelman scheme [119], but does not use a lattice. The procedure is as follows: All pairs of particles are examined. If they are "neighbors" as determined by a user-specified pair criterion, there are several possible cases.

– If neither particle belongs to a cluster, a new clusters is created and both particles are marked as members.
– If one particle is part of a cluster and the other is not, the free particle is added to the cluster.

– If the two particles belong to different clusters, a note is made that the two clusters are one and the same and need to be merged later.

The clusters are labeled by numeric IDs, assigned in ascending order. After all pairs of particles are examined, the clusters marked as identical in the previous step are merged. Cluster IDs are traversed in descending order and merged clusters receive the lower of the two cluster IDs. In this way, the merging can be done in a single pass. ESPResSo 4.0 currently contains pair criteria based on inter-particle distance, pairwise short-range energy, and the presence of a bonded interaction. Further criteria can be added easily. The cluster analysis is organized around a `ClusterStructure` object, which provided the methods to run analysis as well as access to the clusters found. Furthermore, some analysis routines are provided on a per-cluster level, e.g., for a cluster's radius of gyration, fractal dimension, or inertia tensor. Lastly, direct access to the particles making up a cluster is provided.

As an example, let us consider the simulation of a ferrofluid monolayer, i.e, a two-dimensional suspension of soft spheres carrying a magnetic dipole at their center. The example is loosely based on reference [114]. Magnetic particles tend to form chain and ring-like clusters due to the non-isotropic nature of the dipole-dipole potential. For simplicity, we define particles as "neighbors" if the distance between their centers is less than 1.3σ, where σ denotes the particle diameter in the purely repulsive Lennard–Jones potential [119]. The sample system contains 1000 magnetic particles at an area fraction of $\phi = 0.1$. The relative strength of the dipolar interactions to the thermal energy is

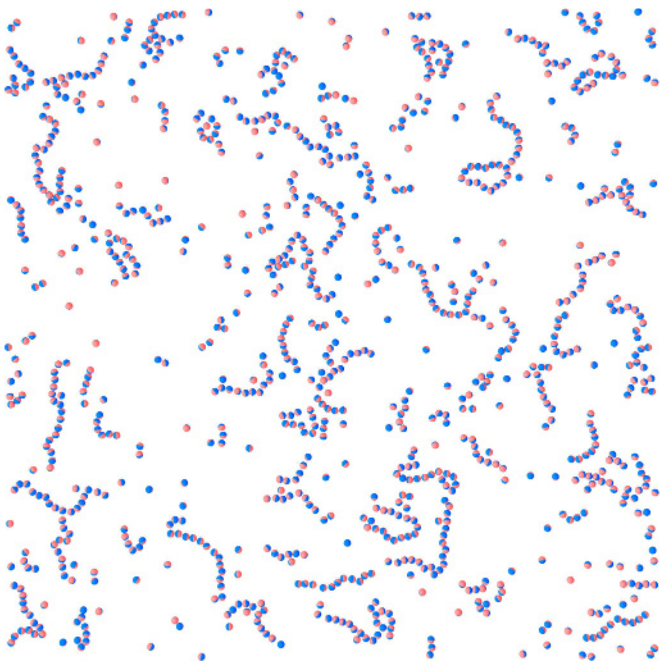
$$\lambda = \frac{\mu_0 m^2}{4\pi\sigma^3 k_\mathrm{B} T}, \tag{7}$$

where μ_0 denotes the vacuum permittivity and m the particles' dipole moment. The dipolar interactions are calculated by means of the dipolar P^{3}M method [120] and the dipolar layer correction [121]. While the particle positions are confined to a plane, the magnetic dipoles can rotate freely in three dimensions. This is called a quasi-2D system.

In Figure 6, a snapshot from the simulation along with a plot of the cluster composition of the suspension averaged over 1000 snapshots are shown. The full simulation script is provided in the supplementary information. Please note that the script is meant as an example. For a scientific study, larger systems, higher accuracies for the dipolar interactions, and longer sampling times are needed.

11 Particle-based reactions

Chemical reactions can take place in many systems, for example water can autodissociate in positively charged protons and negatively charged OH groups, or there are other groups that can dissociate only partially in solvents like water, like the carboxylic group COOH. These groups can be found, for example, in weak polyelectrolytes, charge stabilized colloids, or proteins [122], and their charging state depends on the pH-value [123] of the solution. Therefore, the presence of weak groups in charged polymers (polyelectrolytes) promotes phenomena like protonation-configuration coupling [124,125]. Protonation-configuration coupling means that the configuration of a protein or polymer depends on the protonation state of the titratable groups within the polymer. However, the protonation state itself depends on the configuration. This tight coupling introduces the need to investigate this phenomenon through computer simulations. Reactions can also give rise to a net attraction of particles, which are on average charge neutral [126,127]. Here, the net charge of the



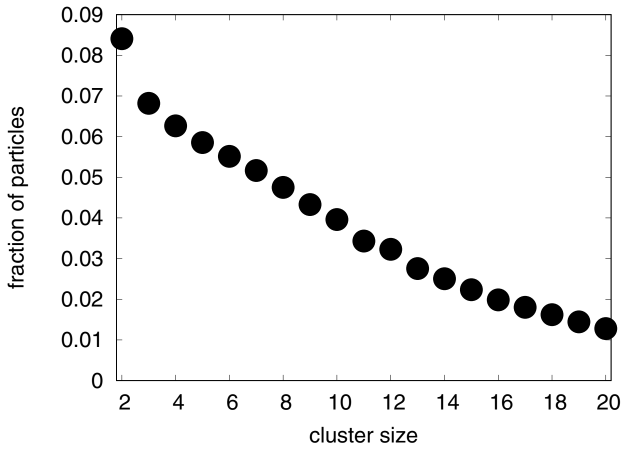


Fig. 6. Top: snapshot of the ferrofluid monolayer. Bottom: composition of the suspension plotted as fraction of particles being part of a cluster of a given size.

particles fluctuates around its mean, thus allowing a particle to be temporarily positive or negative. This fluctuation effect then induces a net attraction. Investigating physical phenomena like the above with the correct statistics, requires introducing particle based reaction schemes in ESPResSo 4.0 which are described in the following.

A typical acidic reaction is shown in Figure 7: an acid particle (HA) may release a proton (H^+), or vice versa, the deprotonated form of the acid (A^-) may take up a proton. Such protonation reactions change the electrostatic charge of the particles taking part in the reaction. Therefore, ESPResSo with its various electrostatic solvers [128], is well-suited to investigate electrostatic effects involved in reactions.

The first scheme that allowed for acid-base reactions in thermodynamic equilibrium using Molecular Dynamics and Monte Carlo simulations appeared in the

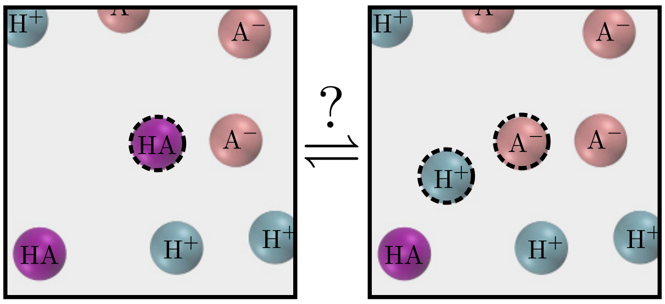


Fig. 7. The protonated/deprotonated states of a Monte-Carlo reaction step. Target particles are circled with a dotted line.

1990s [129] in the form of the constant-pH ensemble. Soon after, the reaction ensemble [130,131] was introduced which allows for the simulation of arbitrary chemical reactions. In both reaction schemes, particle properties (such as the charge) need to be changed and particles need to be created or deleted based on probabilistic criteria. The reaction ensemble and the constant-pH method only differ in the probabilistic criteria [132] and both methods are implemented in ESPResSo 4.0.

In the reaction ensemble, arbitrary reactions can be implemented in the simulation:

$$\sum_{i=1}^{z} \nu_i s_i = 0, \tag{8}$$

where z chemical species of type s_i with stoichiometric coefficients ν_i are reacting [133]. The acceptance probability in the reaction ensemble for a reaction from state r to l is given by [134]

$$\mathrm{acc}^{\mathrm{RE},\xi}(l|r) = \min\left\{1, (V\Gamma)^{\bar{\nu}\xi} \prod_{i=1}^{z} \left[\frac{N_i^r!}{(N_i^r + \xi\nu_i)!}\right] \exp(-\beta \Delta E_{\mathrm{pot},r\to l})\right\}, \tag{9}$$

where N_i^r is the number of particles prior to a reaction and ξ the "extent" of the reaction which is selected randomly with $\xi \pm 1$ and $\beta = 1/(k_{\mathrm{B}}T)$ proportional to the inverse of the temperature. Further parameters are the concentration-dependent reaction constant $\Gamma = c^{\circ\bar{\nu}} \exp(-\beta \Delta G^\circ)$ where c° is the reference concentration for which the change in the free enthalpy ΔG° is tabulated, the potential energy difference with $\Delta E_{\mathrm{pot},r\to l} = E_{\mathrm{pot},r} - E_{\mathrm{pot},l}$, the volume of the system V and the total change in the number of molecules $\bar{\nu} = \sum_i \nu_i$ due to the reaction. The corresponding protonation and deprotonation reactions are usually performed after a fixed number of MD simulation steps with constant particle numbers [134].

In addition to the above standard reaction ensemble algorithm, we implemented a rare event sampling method on top of the reaction ensemble [135], called the Wang–Landau [136] reaction ensemble algorithm. The Wang–Landau scheme is a so-called flat histogramme technique that samples all states with equal probability. This modification of the algorithm might prove useful when investigating systems with metastable states and rare transitions between them.

In the other chemical equilibrium sampling method which is present in literature–the constant-pH ensemble–the acceptance probability for a reaction $\mathrm{HA} \rightleftharpoons \mathrm{A}^- + \mathrm{H}^+$ is given by:

$$\mathrm{acc}^{\mathrm{cpH},\xi}(l|r) = \min\big(1, \exp\big[-\beta(\Delta E_{\mathrm{pot}} \pm (\ln(10)/\beta)\,(\mathrm{pH} - pK_a))\big]\big), \tag{10}$$

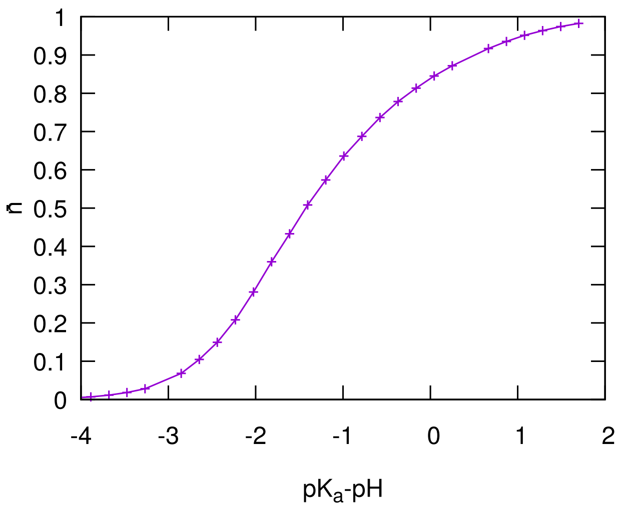


Fig. 8. The titration curve of a linear weak polyacid obtained with the constant pH method. For increasing pH the acid is less associated. The data are taken from reference [132] and simulation parameters are described there in more detail.

where pH is the pH of an implicitly imposed proton reservoir, $pK_a = -\log_{10}\exp(-\beta\Delta G^{\circ})$ and where $\pm$ is used in the case of a association/dissociation. Additionally the dissociation proposal probability is proportional to the number of dissociable groups HA, while the association proposal probability is proportional to the number of deprotonated groups A^-.

As a showcase for our ability to simulate chemcial reactions in ESPResSo 4.0 we present in Figure 8 the titration curve of a linear weak polyelectrolyte. A typical titration curve which is obtained from experiment shows the degree of dissociation as a function of $pK_a - pH$. The degree of association $\overline{n}$ measures how many particles N are in the associated state HA or in the dissociated state A^-: $\overline{n} = \frac{N(\mathrm{HA})}{N_{\mathrm{HA}}+N_{\mathrm{A}^-}}$. At low pH, a weak acid is mostly associated ($\overline{n} \approx 1$) while at higher pH a weak acid becomes less and less associated ($\overline{n} \approx 0$).

The reaction ensemble implementation in ESPResSo also allows for simulations in the grand-canonical ensemble. This is, because the grand canonical simulation scheme [137] can be represented as a reaction $\emptyset \rightleftharpoons \mathrm{A}$ with $\Gamma = c_A^b \exp(\beta\mu_A^{\mathrm{ex},b})$, where c_A^b is the bulk concentration of the species A and $\mu_A^{\mathrm{ex},b}$ the excess chemical potential of the species in the bulk. The needed excess chemical potential can be obtained via the Widoms insertion method [137]. We also implemented this method in ESPResSo 4.0. It resembles a reaction which is constantly rejected while observing potential energy changes due to the insertion of particles. The excess chemical potential in a homogeneous system is then given by [137]

$$\mu^{\mathrm{ex}} = -k_{\mathrm{B}}T\ln\Big(\Big\langle e^{-\beta(E_{\mathrm{pot}}(N+1)-E_{\mathrm{pot}}(N))}\Big\rangle\Big). \tag{11}$$

12 Including explicit dipolar polarization with Drude oscillators

A particle's electron density is redistributed by the local electric field, leading to a nonadditive molecular interaction. Especially the study of ionic liquids (molten salts) polarization plays a large role. Often, this is effectively included in the force-field

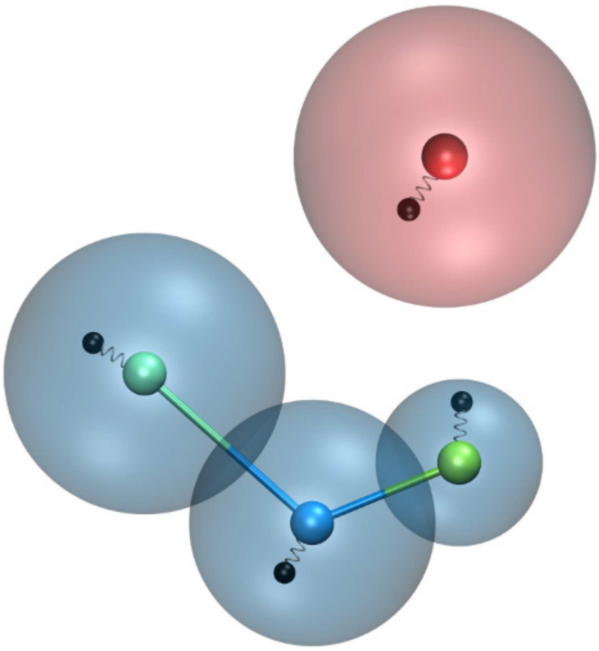


Fig. 9. Sketch of the polarizable coarse grained model for the ionic liquid BMIM PF_6.

(e.g. via reduced charges) to match continuum properties [138–142]. In systems where polarization effects are supposed to be inhomogeneous, e.g. at interfaces, incorporating the dynamic nature of polarization can refine the physical picture. Also simulations of bulk systems can benefit from explicit polarization. For example, the experimental values for the static dielectric constant and the self-diffusion coefficient of water were accurately reproduced using a polarizable model [143]. Thermalized cold Drude oscillators [144], available in ESPResSo 4.0, can be used to model this dynamic particle polarization. The basic idea is to add a "charge-on-a-spring" (Drude charge) to a particle (Drude core) that mimics an electron cloud which can be elongated to create a dynamically inducible dipole (see Fig. 9) [145]. The energetic minimum of the Drude charge can be obtained self-consistently, which requires several iterations of the system's electrostatics solver and is usually considered computationally expensive [146]. However, with thermalized cold Drude oscillators, the distance between Drude charge and core is coupled to a thermostat, so that it fluctuates around the self-consistent field solution. This thermostat is kept at a lower temperature compared to the global temperature to minimize the heat flow into the system. A second thermostat is applied on the center of mass of the Drude charge and core system to maintain the global temperature. The downside of this approach is that usually a smaller time step has to be used to resolve the high frequency oscillations of the spring to get a stable system [147]. In ESPResSo, the basic ingredients to simulate such a system are split into three bonds; their combined usage creates polarizable compounds:

1. A *harmonic bond* between charge and core.
2. For the cold thermostat, a Langevin-type *thermalized distance bond* is used for the Drude-core distance.
3. A *subtract* P^3M *short-range bond* to cancel unwanted electrostatic interaction.

The system-wide thermostat has to be applied to the center of mass and not to the core particle directly. Therefore, the particles have to be excluded from the global thermostat, which is possible in ESPResSo by setting the temperature and friction coefficient of the Drude complex to zero. It thus remains possible to use a global Langevin thermostat for non-polarizable particles. As the Drude charge should not alter the charge or mass of the Drude complex, both properties have to be subtracted from the core when adding the Drude particle. In the following convention, we assume that the Drude charge is always negative. It is calculated via the spring constant k and polarizability α (in units of inverse volume) with $q_d = \sqrt{-k\alpha}$. For polarizable molecules (i.e., connected particles, coarse grained models etc.) with partial charges on the molecule sites, the Drude charges will have electrostatic interaction with other cores of the molecule. Often, this is unwanted, as it might be already part of the force-field (via partial charges or parametrization of the covalent bonds). Without any further measures, the elongation of the Drude particles will be greatly affected by the partial charges of the molecule that are in close proximity. To prevent this, one has to cancel the interaction of the Drude charge q_d with the partial charges of the cores q_{partial} within the molecule. This can be done with the *subtracts P^3M short-range bond*, which is also used to cancel the electrostatics between Drude core $\leftrightarrow q_{\text{core}}$ and Drude charge q_d. This ensures that only the dipolar interaction inside the molecule remains. The error of this approximation increases with the share of the long-range part of the electrostatic interaction. In most cases, this error is negligible comparing the distance of the charges and the real-space cutoff of the P^{3}M electrostatics solver. In ESPResSo, helper methods assist setting up this exclusion. In combination with particle polarizability, the *Thole correction* [148] is often used to correct for overestimation of induced dipoles at short distances. Ultimately, it alters the short-range electrostatics of P^{3}M to result in a damped Coulomb interaction potential

$$V(r) = \frac{q_i q_j}{r}\left[1 - e^{-sr}\left(1 + \frac{sr}{2}\right)\right]. \tag{12}$$

The Thole scaling coefficient s is related to the polarizabilities α_k and Thole damping parameters a_k of the interacting species via

$$s = \frac{(a_i + a_j)/2}{(\alpha_i \alpha_j)^{1/6}}. \tag{13}$$

Note that for the Drude oscillators, the Thole correction should be applied only for the dipole part $\pm q_d$ added by the Drude charge and not on the total core charge, which can be different for polarizable ions. Also note that the Thole correction acts between all dipoles, intra- and intermolecular. Again, the accuracy is related to the P^{3}M accuracy and the split between short-range and long-range electrostatics interaction. ESPResSo assists with the bookkeeping of mixed scaling coefficients and has convenient methods to set up all necessary Thole interactions.

13 Active particles

In ESPResSo 4.0 it is possible to simulate active systems, wherein individual particles constantly transduce (internal) energy to perform work in the form of motion. The combination of this self-propulsion with particle interactions gives rise to unique out-of-equilibrium behaviors, of which living systems offer a wealth of examples: mammals, fish and birds [149–153], as well as microorganisms [154–161]. The last decade and a half have seen the rapid development of man-made counterparts to microorganisms, following the first experimental realizations of “chemical swimmers” by [162] and [163], respectively. These artificial swimmers reduce the complexity of living matter, but still display signature features of being out of equilibrium, such as the ability to perform work [164] and motility-induced phase separation [165,166].

Significant progress has been made both theoretically and computationally in understanding the individual and collective behavior of swimmers using simple models. Arguably the most famous of these is the Vicsek model [167], followed closely by the active Brownian model [168–170]. In our developments, we have drawn inspiration from the Active Brownian Model (ABM) and created a variant thereof: The Active Langevin Model (ALM), which we will describe here. This ALM can also be coupled to the ESPResSo (GPU) lattice Boltzmann (LB) fluid dynamics solver [48,70] to account for the characteristic dipolar flow field that is associated with self-propulsion by microorganisms [171,172] as well as chemical swimmers [173]. Our implementation [65] is similar to the sub-lattice approach introduced by Nash et al. [174,175]. We refer to this extension as the hydrodynamic ALM (HALM) and we will briefly touch upon it here.

The ALM equation of motion for the translation of the ith swimmer's center of mass $\boldsymbol{r}_i$ is given by:

$$M\frac{\partial^2}{\partial t^2}\boldsymbol{r}_i = -\underline{\boldsymbol{\Gamma}}_t\frac{\partial}{\partial t}\boldsymbol{r}_i + f\hat{\boldsymbol{u}}_i - \sum_{j\neq i}\boldsymbol{\nabla}V(r_{ij},\boldsymbol{Q}_i,\boldsymbol{Q}_j) + \boldsymbol{\xi}_{i,t}(t). \tag{14}$$

Here, we assume that the swimmer is fully shape anisotropic and we have introduced the following quantities: the mass M, the translational diffusion tensor $\underline{\boldsymbol{\Gamma}}_t$, a (co-rotating) unit vector that gives the direction of self-propelled motion $\hat{\boldsymbol{u}}_i$, the self-propulsion force f, and swimmer–swimmer interaction potential V. The latter depends on the separation $r_{ij} \equiv |\boldsymbol{r}_i - \boldsymbol{r}_j|$ and orientation of the swimmers, as specified by quaterions $\boldsymbol{Q}_i$. Thermal noise is introduced via $\boldsymbol{\xi}_{i,t}(t)$, which satisfies $\langle\boldsymbol{\xi}_{i,t}(t)\rangle = \boldsymbol{0}$ and $\langle\boldsymbol{\xi}_{i,t}(t) \otimes \boldsymbol{\xi}_{j,t}(t')\rangle = 6k_{\mathrm{B}}T\underline{\boldsymbol{\Gamma}}_t\delta_{ij}\delta(t-t')$. The equation of motion for quaternions is lengthy and well-described in reference [176], it is therefore not reproduced here.

Figure 10a shows a representation of an ALM swimmer. Each particle is assigned a direction of self-propulsion $\hat{\boldsymbol{u}}_i$, along which it experiences a constant force $\boldsymbol{F}_{\mathrm{ALM}} = f\hat{\boldsymbol{u}}$. This self-propulsion force balances against the friction exerted by the implicit solvent, $\boldsymbol{F}_{\mathrm{fric}} = \underline{\boldsymbol{\Gamma}}_t\boldsymbol{U}$, leading to persistent directed motion with swim speed $U = |\boldsymbol{U}|$. Individual swimmers can have different propulsion forces and/or friction matrices in ESPResSo 4.0, providing users with tremendous flexibility in creating mixtures with a range of mobilities.

A swimmer's direction of self-propulsion may be changed by rotational Brownian motion or by torques, either from external fields or collisions. The former leads to enhanced diffusion [163] as shown in Figure 10b. ALM in ESPResSo 4.0 currently does not allow for the coupling of translational and rotational degrees of freedom, as required for the simulation of L-shaped [177] and chiral swimmers [178]; future releases will bring such functionality. Lastly, note that ALM is not fully damped (Fig. 10b), i.e., there is an inertial component to the dynamics, which is uncommon in simulating (large) colloidal particles. However, ALM approaches the fully damped ABM when the friction coefficient is chosen to be large. In this limit, the time step should be suitably small to ensure that the algorithm produces the correct physics.

HALM introduces coupling between ALM and an LB fluid to account for hydrodynamic interactions between swimmers, particles, and obstacles. In developing HALM we have drawn inspiration from the work of Nash et al. [174,175], who formulated a similar coupling to an LB fluid for the ABM. A directional self-propulsion force is applied to the particle, similar to the way this is done in ALM. In LB, a single point particle gains an effective radius [179] or rather an inherent friction by coupling to the grid. This friction counter-balances the propulsive force leading to a constant

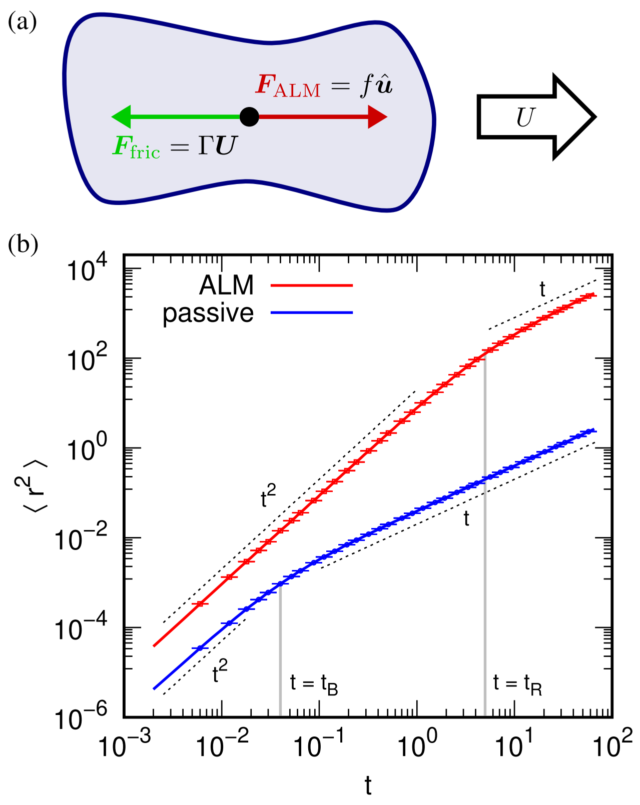


Fig. 10. The Active Langevin Model (ALM). (a) A two-dimensional (2D) schematic of a shape anisotropic ALM self-propelled particle. The balance of self-propulsion force $f\hat{\boldsymbol{u}}$ and fluid friction $\underline{\boldsymbol{\Gamma}}_t\boldsymbol{U}$, leads to a constant swim speed U. (b) The mean-squared displacement $\langle r^2 \rangle$ as a function of time t of a passive Langevin particle (blue) and ALM swimmer (red), with error bars indicating the standard error. In the passive case, the ballistic regime $\langle r^2 \rangle \propto t^2$ transitions to a diffusive regime $\langle r^2 \rangle \propto t$ around the ballistic-to-diffusion crossover time t_{B}, as indicated by the grey vertical line. The ballistic regime is stretched by the activity and enhanced diffusion sets on a time scale associated with rotational diffusion t_{R}.

swim speed. However, this also leads to force onto the LB fluid, while self-propulsion in a fluid should be force free. Therefore, a counter-force is applied to the LB fluid to ensure that momentum is not transferred into it globally, see Figure 11. This force/counter-force pair induces a flow field with the typical leading-order dipolar contribution that characterizes self-propulsion. Figure 11 illustrates two combinations of the force pair leading to the well-known puller and pusher dipolar flow fields, representing, e.g., algae [171,180] and spermatozoa [181], respectively. We refer to reference [65] for full details of the implementation.

The reason for introducing ALM and HALM here, rather than relying on Brownian dynamics, is related to the way particles in ESPResSo couple to the LB fluid. Using these algorithms in tandem, the effect of the hydrodynamic interactions between swimmers may be disentangled, in much the same way as was done for passive particles [182]. Careful tuning of the relevant hydrodynamic parameters and the self-propulsion forces is key to reproducing low-Reynolds number hydrodynamic solutions, we refer to reference [183] for an in-depth discussion. These authors have verified that the Nash et al. implementation [174,175] and HALM have the same near-field flow characteristics, long-range hydrodynamic retardation effects, and par-

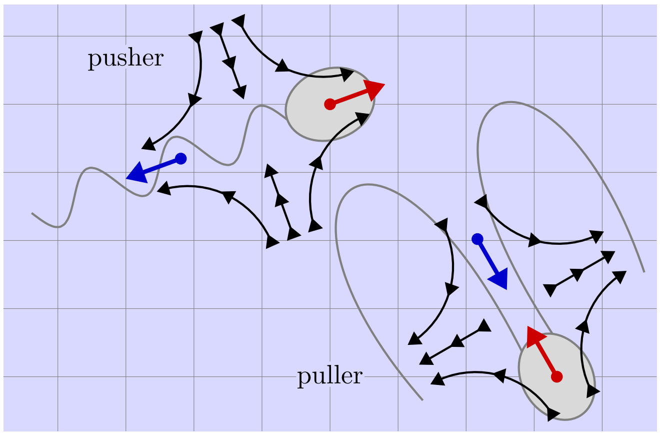


Fig. 11. Sketches of two types of force-free hydrodynamic ALM (HALM) swimmers. Pusher and puller swimmers result from a change in the placement of the counter force (blue arrow) with respect to the swimmer's center, on which a propulsive force (red arrow) is applied. These forces are resolved sub-lattice (gray mesh) and interpolated to achieve the flow fields, as sketched using black arrows. The inherent friction of the swimmer's body with the lattice Boltzmann fluid leads to a persistent motion with a well-defined swim speed. The similarity of this figure to Figure 4.1 from R. Nash's Ph.D. thesis [175] is intentional and emphasizes the inspiration for our algorithm.

ticle dynamics [184]. Thus, the inertial component to HALM does not significantly affect the dynamics of our model.

ALM and HALM can both be used in conjunction with the raspberry method of creating shape-anisotropic particles [63,64,110,185]. This enables, for example, the study of the effect of polarization and roughness on motility-induced clustering [186]. HALM combined with raspberry particles leads to hydrodynamic multipole moments beyond the leading dipole moment for the self-propulsion of shape-anisotropic particles in fluids [65]. This property has been exploited to determine the effect of these hydrodynamic moments on the motion of these swimmers in confining geometries [66], as well as their interaction with passive particles in their surrounding [65,183]. The use of raspberry particles in combination with HALM removes most of the lattice artifacts that point-particle HALM suffers from [65]. We therefore recommend employing this combination within ESPResSo to properly account for any rigid body dynamics in a fluid, including rotational coupling due to shape anisotropies.

14 Conclusions

In this work, we outlined the major revisions that ESPResSo 4.0 has undergone, and the benefits that the user will experience from these changes. These include the reimplementation of the user interface in Python and an overhaul of the core architecture to modern software development paradigms. We have also reported new features and presented them accompanied by specific use cases. This enables the simulation of systems at the forefront of soft matter research, including active matter and catalytic reactions.

We foresee a bright future for ESPResSo as a software package and as a research project. Together with our collaborators, we will further improve the documentation and usability of this platform. In addition, algorithmic improvements are planned, which include: (i) Adaptive grid refinement to the electrokinetics code. (ii) A load-balancing scheme for efficient simulation of heterogeneous systems. (iii) Lees-Edwards boundary conditions for rheological measurements. We cordially invite new users to try out ESPResSo as a simulation tool for their research and participate in its continued development.

The current status of the package and the latest tutorials and documentation can be found on the project website http://espressomd.org, or on GitHub https://github.com/espressomd/.

We would like to acknowledge more than 100 researchers who contributed over the last fifteen years to the ESPResSo software, and whose names can be found on our website http://espressomd.org or in the AUTHORS file distributed with ESPResSo. We are also grateful to our colleagues of the SFB 716 for numerous suggestions for extending the capabilities of ESPResSo and in helping us to improve our software. As part of a collaboration [108], Milena Smiljanic contributed code to the cluster analysis framework, particularly to the analysis routines on the single cluster level. CH, KS, and FW gratefully acknowledge funding by the German Science Foundation (DFG) through the collaborative research center SFB 716 within TP C5, the SimTech cluster of excellence (EXC 310), and grants HO 1108/25-1, HO 1108/26-1, AR 593/7-1, HO 1108/28-1. MK, JdG, and CH thank the DFG for funding through the SPP 1726 Microswimmers–From Single Particle Motion to Collective Behavior. JdG further acknowledges funding from an NWO Rubicon Grant (#680501210) and a Marie Skłodowska-Curie Intra European Fellowship (G.A. No. 654916) within Horizon 2020. The simulations were partially performed on the bwUniCluster funded by the Ministry of Science, Research and Arts and the Universities of the State of Baden-Württemberg, Germany, within the framework program bwHPC.

Author contribution statement

Supervision: CH; Funding Acquisition, CH; Resources, CH; Conceptualization: CH, RW, and FW. Writing: All authors contributed to the preparation of the manuscript. Software and Validation: FW, RW, and KS are core developers of ESPResSo. The main code contributors of individual features discussed in this article are the following. Test infrastructure: KS, MK. Visualization: KB, MK. Particle-based reactions: JL. Drude oscillators: KB. External fields: FW. Steepest descent energy minimization: FW. Cluster analysis: RW. Active particles: JdG, HM. H5MD parallel output: KS. VTF output: DS. A full list of code contributions can be found at https://github.com/espressomd/espresso/graphs/contributors.

References

1. P.G. de Gennes, Rev. Mod. Phys. **64**, 645 (1992)
2. M. Doi, *Soft matter physics* (Oxford University Press, Hardcover, 2013)
3. J.-L. Barrat, J.-P. Hansen, *Basic concepts fo simple and complex liquids* (Cambridge University Press, Cambridge, 2003)
4. M. Doi, S.F. Edwards, in *The theory of polymer dynamics* (Oxford University Press, 1988), Vol. 73
5. M. Rubinstein, R.H. Colby, *Polymer physics* (Oxford University Press, Oxford, UK, 2003)
6. E.J. Verwey, J.T.G. Overbeek, *Theory of the stability of lyophobic colloids* (Elsevier, Amsterdam, 1948)
7. H. Löwen, J. Phys.: Condens. Matter **13**, R415 (2001)
8. S. Chandrasekhar, *Liquid crystals* (Cambridge University Press, Cambridge, 1992)
9. I.V. Hamley, *Introduction to soft matter* (Wiley, Chichester, 2003)
10. P. Nelson, *Biological physics – energy, information, life* (Freeman, New York, 2004)

11. I. Levental, P.C. Georges, P.A. Janmey, Soft Matter **3**, 299 (2007)
12. J. Ubbink, A. Burbidge, R. Mezzenga, Soft Matter **4**, 1569 (2008)
13. S. Polarz, M. Antonietti, Chem. Commun. **22**, 2593 (2002)
14. K. Kroy, E. Frey, Ann. Phys. **14**, 20 (2005)
15. D. Frenkel, Science **296**, 65 (2002)
16. U. Seifert, Rep. Progr. Phys. **75**, 126001 (2012)
17. M.E. Cates, Rep. Progr. Phys. **75**, 042601 (2012)
18. É. Fodor, M. Marchetti, Physica A **504**, 106 (2018)
19. B.J. Alder, T.E. Wainwright, J. Chem. Phys. **27**, 1208 (1957)
20. B. Alder, T. Wainwright, Phys. Rev. **127**, 359 (1962)
21. P. Pusey, W.C. Poon, S. Ilett, P. Bartlett, J. Phys.: Condens. Matter **6**, A29 (1994)
22. M.A. Bates, D. Frenkel, J. Chem. Phys. **109**, 6193 (1998)
23. R. van Roij, M. Dijkstra, J.-P. Hansen, Phys. Rev. E **59**, 2010 (1999)
24. M.E. Leunissen, C.G. Christova, A.P. Hynninen, C.P. Royall, A.I. Campbell, A. Imhof, M. Dijkstra, R. van Roij, A. van Blaaderen, Nature **437**, 235 (2005)
25. P.J. Camp, J.C. Shelley, G.N. Patey, Phys. Rev. Lett. **84**, 115 (2000)
26. Z. Wang, C. Holm, H.W. Müller, Phys. Rev. E **66**, 021405 (2002)
27. S.H.L. Klapp, M. Schoen, J. Mol. Liq. **109**, 55 (2004)
28. M. Klinkigt, R. Weeber, S. Kantorovich, C. Holm, Soft Matter **9**, 3535 (2013)
29. M.J. Stevens K. Kremer, Phys. Rev. Lett. **71**, 2228 (1993)
30. A.V. Dobrynin, M. Rubinstein, S.P. Obukhov, Macromolecules **29**, 2974 (1996)
31. U. Micka, C. Holm, K. Kremer, Langmuir **15**, 4033 (1999)
32. H.J. Limbach, C. Holm, K. Kremer, Europhys. Lett. **60**, 566 (2002)
33. S. Schneider, P. Linse, Eur. Phys. J. E **8**, 457 (2002)
34. Q. Yan, J.J. de Pablo, Phys. Rev. Lett. **91**, 018301 (2003)
35. B.A. Mann, R. Everaers, C. Holm, K. Kremer, Europhys. Lett. **67**, 786 (2004)
36. R. Weeber, S. Kantorovich, C. Holm, Soft Matter **8**, 9923 (2012)
37. R. Weeber, M. Hermes, A.M. Schmidt, C. Holm, J. Phys.: Condens. Matter **30**, 063002 (2018)
38. S.J. Plimpton, J. Comput. Phys. **117**, 1 (1995)
39. H.V. Guzman, H. Kobayashi, N. Tretyakov, A.C. Fogarty, K. Kreis, J. Krajniak, C. Junghans, K. Kremer, T. Stuehn, arXiv:1806.10841 (2018)
40. H.J.C. Berendsen, D. van der Spoel, R. van Drunen, Comput. Phys. Commun. **91**, 43 (1995)
41. D. van der Spoel, E. Lindahl, B. Hess, G. Groenhof, A.E. Mark, H.J.C. Berendsen, J. Comput. Chem. **26**, 1701 (2005)
42. S. Pronk, S. Páll, R. Schulz, P. Larsson, P. Bjelkmar, R. Apostolov, M.R. Shirts, J.C. Smith, P.M. Kasson, D. van der Spoel, B. Hess, E. Lindahl, Bioinformatics **29**, 845 (2013)
43. M. Nelson, W. Humphrey, A. Gursoy, A. Dalke, L. Kale, R.D. Skeel, K. Schulten, Int. J. Supercomput. Appl. **10**, 251 (1996)
44. J.C. Phillips, R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R.D. Skeel, L. Kale, K. Schulten, J. Comput. Chem. **26**, 1781 (2005)
45. J.C. Phillips, J.E. Stone, K.L. Vandivort, T.G. Armstrong, J.M. Wozniak, M. Wilde, K. Schulten, in *Proceedings of the 1st first workshop for high performance technical computing in dynamic languages* (IEEE Press, New York, 2014), p. 6
46. D.A. Pearlman, D.A. Case, J.W. Caldwell, W.R. Ross, I.T.E. Cheatham, S. DeBolt, D. Ferguson, G. Seibel, P. Kollman, Comput. Phys. Commun. **91**, 1 (1995)
47. D.A. Case, T.E. Cheatham, T. Darden, H. Gohlke, R. Luo, K.M. Merz, A. Onufriev, C. Simmerling, B. Wang, R. Woods, J. Comput. Chem. **26**, 1668 (2005)
48. A. Arnold, O. Lenz, S. Kesselheim, R. Weeber, F. Fahrenberger, D. Röhm, P. Košovan, C. Holm, in *Meshfree methods for partial differential equations VI*, Lecture Notes in Computational Science and Engineering, edited by M. Griebel, M.A. Schweitzer (Springer, Berlin Heidelberg, 2013), Vol. 89, pp. 1–23
49. M. Deserno, C. Holm, J. Chem. Phys. **109**, 7678 (1998)
50. M. Deserno, C. Holm, J. Chem. Phys. **109**, 7694 (1998)

51. A. Arnold, C. Holm, Comput. Phys. Commun. **148**, 327 (2002)
52. A. Arnold, C. Holm, Chem. Phys. Lett. **354**, 324 (2002)
53. A. Arnold, J. de Joannis, C. Holm, J. Chem. Phys. **117**, 2496 (2002)
54. A. Arnold, J. de Joannis, C. Holm, J. Chem. Phys. **117**, 2503 (2002)
55. A. Arnold, C. Holm, in *Advanced computer simulation approaches for soft matter sciences II*, Advances in polymer sciences, edited by C. Holm, K. Kremer (Springer, Berlin, 2005), Vol. II, pp. 59–109
56. A. Arnold, C. Holm, J. Chem. Phys. **123**, 144103 (2005)
57. A. Arnold, B.A. Mann, C. Holm, in *Computer simulations in Condensed Matter: from Materials to Chemical Biology*, Lecture Notes in Physics, edited by M. Ferrario, G. Ciccotti, K. Binder (Springer, Berlin, Germany, 2006), Vol. 1, pp. 193–222
58. S. Tyagi, A. Arnold, C. Holm, J. Chem. Phys. **127**, 154723 (2007)
59. S. Tyagi, A. Arnold, C. Holm, J. Chem. Phys. **129**, 204102 (2008)
60. C. Tyagi, M. Süzen, M. Sega, M. Barbosa, S.S. Kantorovich, C. Holm, J. Chem. Phys. **132**, 154112 (2010)
61. A. Arnold, K. Breitsprecher, F. Fahrenberger, S. Kesselheim, O. Lenz, C. Holm, Entropy **15**, 4569 (2013)
62. F. Fahrenberger C. Holm, Phys. Rev. E **90**, 063304 (2014)
63. L.P. Fischer, T. Peter, C. Holm, J. de Graaf, J. Chem. Phys. **143**, 084107 (2015)
64. J. de Graaf, T. Peter, L.P. Fischer, C. Holm, J. Chem. Phys. **143**, 084108 (2015)
65. J. de Graaf, H. Menke, A.J. Mathijssen, M. Fabritius, C. Holm, T.N. Shendruk, J. Chem. Phys. **144**, 134106 (2016)
66. J. de Graaf, A.J. Mathijssen, M. Fabritius, H. Menke, C. Holm, T.N. Shendruk, Soft Matter **12**, 4704 (2016)
67. G. Inci, A. Arnold, A. Kronenburg, R. Weeber, Aerosol Sci. Technol. **48**, 842 (2014)
68. G. Inci, A. Kronenburg, R. Weeber, D. Pflüger, Flow, Turbul. Combust. **98**, 1065 (2017)
69. C. Schober, D. Keerl, M.J. Lehmann, M. Mehl, in *Proceedings of the VII international conference on coupled problems in science and engineering*, edited by M. Papadrakakis, E. Oñate, B. Schrefler (International Center for Numerical Methods in Engineering, 2016)
70. D. Röhm, A. Arnold, Eur. Phys. J. Special Topics **210**, 89 (2012)
71. I. Cimrák, M. Gusenbauer, T. Schrefl, Comput. Math. Appl. **64**, 278 (2012)
72. I. Cimrak, M. Gusenbauer, I. Jančigová, Comput. Phys. Commun. **185**, 900 (2014)
73. I. Cimrák, I. Jancigová, K. Bachratá, H. Bachratỳ, in *III International conference on particle-based methods–fundamentals and applications particles* (2013), Vol. 2013, pp. 133–144
74. G. Rempfer, G.B. Davies, C. Holm, J. de Graaf, J. Chem. Phys. **145**, 044901 (2016)
75. A. Guckenberger, M.P. Schraml, P.G. Chen, M. Leonetti, S. Gekle, Comput. Phys. Commun. **207**, 1 (2016)
76. C. Bächer, L. Schrack, S. Gekle, Phys. Rev. Fluids **2**, 013102 (2017)
77. K. Kratzer, A. Arnold, R.J. Allen, J. Chem. Phys. **138**, 164112 (2013)
78. K. Kratzer, J.T. Berryman, A. Taudt, J. Zeman, A. Arnold, Comput. Phys. Commun. **185**, 1875 (2014)
79. S. Samin, Y. Tsori, C. Holm, Phys. Rev. E **87**, 052128 (2013)
80. A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, A. Lerer, *Neural information processing systems* (2017)
81. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, J. Mach. Learn. Res. **12**, 2825 (2011)
82. F. Chollet, *Deep learning with Python* (Manning Publications Co., Greenwich, 2017)
83. A. Meurer, C.P. Smith, M. Paprocki, O. Čertík, S.B. Kirpichev, M. Rocklin, A. Kumar, S. Ivanov, J.K. Moore, S. Singh, T. Rathnayake, S. Vig, B.E. Granger, R.P. Muller, F. Bonazzi, H. Gupta, S. Vats, F. Johansson, F. Pedregosa, M.J. Curry, A.R. Terrel, V. Roučka, A. Saboo, I. Fernando, S. Kulal, R. Cimrman, A. Scopatz, PeerJ Comput. Sci. **3**, e103 (2017)

84. W. Stein, D. Joyner, ACM SIGSAM Bull. **39**, 61 (2005)
85. J.D. Hunter, Comput. Sci. Eng. **9**, 90 (2007)
86. P. Ramachandran, G. Varoquaux, Comput. Sci. Eng. **13**, 40 (2011)
87. E. Jones, T. Oliphant, P. Peterson, et al., SciPy: open source scientific tools for Python, 2001–2019
88. W. McKinney, in *Proceedings of the 9th Python in science conference*, edited by S. van der Walt, J. Millman (2010), pp. 51–56
89. D. Reith, M. Pütz, F. Müller-Plathe, J. Comput. Chem. **24**, 1624 (2003)
90. T. Kluyver, B. Ragan-Kelley, F. Pérez, B.E. Granger, M. Bussonnier, J. Frederic, K. Kelley, J.B. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, C. Willing, in *Positioning and power in academic publishing: players, agents and agendas*, edited by F. Loiyides, B. Schmidt (IOS Press, London, 2016), pp. 87–90
91. R.W. Hockney, J.W. Eastwood, *Computer simulation using particles* (IOP, London, 1988)
92. A. Arnold, F. Fahrenberger, C. Holm, O. Lenz, M. Bolten, H. Dachsel, R. Halver, I. Kabadshow, F. Gähler, F. Heber, J. Iseringhausen, M. Hofmann, M. Pippig, D. Potts, G. Sutmann, Phys. Rev. E **88**, 063308 (2013)
93. F. Nestler, Appl. Numer. Math. **105**, 25 (2016)
94. R. Weeber, F. Nestler, F. Weik, M. Pippig, D. Potts, C. Holm, arXiv:1808.10341 (2018)
95. Y.L. Raikher, O.V. Stolbov, JMMM **258/259**, 477 (2003)
96. K. Morozov, M. Shliomis, H. Yamaguchi, Phys. Rev. E **79**, 040801 (2009)
97. U. Ayachit, *The ParaView guide: a parallel visualization application* (Kitware, Inc., USA, 2015)
98. W. Humphrey, A. Dalke, K. Schulten, J. Mol. Graph. **14**, 33 (1996)
99. M. Fletcher, R. Liebscher, PyOpenGL–the Python OpenGL binding, 2005
100. D. Shreiner, *OpenGL reference manual: the official reference document to OpenGL, version 1.2*, 3rd edn. (Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999)
101. D. Röhm, *Lattice Boltzmann simulations on GPUs, diplomarbeit* (University of Stuttgart, Germany, 2011)
102. P. de Buyl, P.H. Colberg, F. Höfling, Comput. Phys. Commun. **185**, 1546 (2014)
103. The HDF Group, *Hierarchical data format, version 5*, http://www.hdfgroup.org/HDF5/
104. A. Collette, *Python and HDF5* (O'Reilly, 2013)
105. Boost C++ libraries, https://www.boost.org (1998–2019)
106. B. Dünweg, A.J.C. Ladd, in *Advanced computer simulation approaches for soft matter sciences III*, Advances in Polymer Science (Springer-Verlag Berlin, Berlin, Germany, 2009), Vol. 221, pp. 89–166
107. Clang: a C language family frontend for LLVM, https://clang.llvm.org/
108. M. Smiljanic, R. Weeber, C. Holm, A. Kronenburg, Eur. Phys. J. Special Topics. Submitted
109. K. Breitsprecher, C. Holm, S. Kondrat, ACS Nano **12**, 9733 (2018)
110. V. Lobaskin, B. Dünweg, New J. Phys. **6**, 54 (2004)
111. M. Radu, T. Schilling, Europhys. Lett. **105**, 26001 (2014)
112. K. Kratzer, A. Arnold, Soft Matter **11**, 2174 (2015)
113. K. Butter, P.H.H. Bomans, P.M. Frederik, G.J. Vroege, A.P. Philipse, Nat. Mater. **2**, 88 (2003)
114. J.J. Cerdà, S. Kantorovich, C. Holm, J. Phys.: Condens. Matter **20**, 204125 (2008)
115. R. Weeber, M. Klinkigt, S. Kantorovich, C. Holm, J. Chem. Phys. **139**, 214901 (2013)
116. J.G. Donaldson, S.S. Kantorovich, Nanoscale **7**, 3217 (2015)
117. A. Attili, F. Bisetti, M.E. Mueller, H. Pitsch, Combust. Flame **161**, 1849 (2014)
118. W. Lechner, C. Dellago, J. Chem. Phys. **129**, 114707 (2008)
119. J.D. Weeks, D. Chandler, H.C. Andersen, J. Chem. Phys. **54**, 5237 (1971)
120. J.J. Cerdà, V. Ballenegger, O. Lenz, C. Holm, J. Chem. Phys. **129**, 234104 (2008)
121. A. Bródka, Chem. Phys. Lett. **400**, 62 (2004)

122. W. Richtering, Smart Coll. Mater. **133**, 9 (2006)
123. J.M. Berg, *Biochemistry*, 8th edn. (Freeman, New York, NY, USA, 2015)
124. M. Castelnovo, P. Sens, J.-F. Joanny, Eur. Phys. J. E **1**, 115 (2000)
125. C. Shi, J.A. Wallace, J.K. Shen, Biophys. J. **102**, 1590 (2012)
126. M. Lund, B. Jönsson, Biochemistry **44**, 5722 (2005)
127. M. Lund, B. Jönsson, Quart. Rev. Biophys. **46**, 265 (2013)
128. H.J. Limbach, A. Arnold, B.A. Mann, C. Holm, Comput. Phys. Commun. **174**, 704 (2006)
129. C.E. Reed, W.F. Reed, J. Chem. Phys. **96**, 1609 (1992)
130. E.R. Smith, J. Stat. Phys. **77**, 449 (1994)
131. J.K. Johnson, A.Z. Panagiotopoulos, K.E. Gubbins, Mol. Phys. **81**, 717 (1994)
132. J. Landsgesell, C. Holm, J. Smiatek, Eur. Phys. J. Special Topics **226**, 725 (2017)
133. P. W. Atkins, J. de Paula, *Physical chemistry* (Oxford Univ. Press, Oxford, UK, 2010)
134. C. Heath Turner, J.K. Brennan, M. Lisal, W.R. Smith, J. Karl Johnson, K.E. Gubbins, Mol. Simul. **34**, 119 (2008)
135. J. Landsgesell, C. Holm, J. Smiatek, J. Chem. Theory Comput. **13**, 852 (2017)
136. F. Wang, D.P. Landau, Phys. Rev. E **64**, 056101 (2001)
137. D. Frenkel, B. Smit, *Understanding molecular simulation*, 1st edn. (Academic Press, San Diego, 1996)
138. I. Leontyev, A. Stuchebrukhov, Phys. Chem. Chem. Phys. **13**, 2613 (2011)
139. J. Schmidt, C. Krekeler, F. Dommert, Y. Zhao, R. Berger, L. Delle Site, C. Holm, J. Phys. Chem. B **114**, 6150 (2010)
140. F. Dommert, K. Wendler, R. Berger, L.D. Site, C. Holm, Chem. Phys. Chem. **13**, 1625 (2012)
141. F. Dommert, C. Holm, Phys. Chem. Chem. Phys. **15**, 2037 (2013)
142. M. Kohagen, P.E. Mason, P. Jungwirth, J. Phys. Chem. B **120**, 1454 (2016)
143. G. Lamoureux, E. Harder, I. Vorobyov, B. Roux, A. MacKerell, Chem. Phys. Lett. **418**, 245 (2006)
144. G. Lamoureux, B. Roux, J. Chem. Phys. **119**, 3025 (2003)
145. J.R. Bordin, R. Podgornik, C. Holm, Eur. Phys. J. Special Topics **225**, 1693 (2016)
146. H. Yu, T. Hansson, W.F. van Gunsteren, J. Chem. Phys. **118**, 221 (2003)
147. P. Mitchell, D. Fincham, J. Phys.: Condens. Matter **5**, 1031 (1993)
148. B.T. Thole, Chem. Phys. **59**, 341 (1981)
149. D. Helbing, I. Farkas, T. Vicsek, Nature **407**, 487 (2000)
150. M. Ballerini, N. Cabibbo, R. Candelier, A. Cavagna, E. Cisbani, I. Giardina, V. Lecomte, A. Orlandi, G. Parisi, A. Procaccini, M. Viale, V. Zdravkovic, Proc. Natl. Acad. Sci. **105**, 1232 (2008)
151. Y. Katz, K. Tunstrøm, C. Ioannou, C. Huepe, I. Couzin, Proc. Natl. Acad. Sci. **108**, 18720 (2011)
152. J. Zhang, W. Klingsch, A. Schadschneider, A. Seyfried, in *Traffic and granular flow '11*, edited by V.V. Kozlov, A.P. Buslaev, A.S. Bugaev, M.V. Yashina, A. Schadschneider, M. Schreckenberg (Springer, Berlin, Heidelberg, 2013), p. 241
153. J. Silverberg, M. Bierbaum, J. Sethna, I. Cohen, Phys. Rev. Lett. **110**, 228701 (2013)
154. D. Woolley, Reproduction **126**, 259 (2003)
155. I. Riedel, K. Kruse, J. Howard, Science **309**, 300 (2005)
156. A. Sokolov, I.S. Aranson, J.O. Kessler, R.E. Goldstein, Phys. Rev. Lett. **98**, 158102 (2007)
157. M. Polin, I. Tuval, K. Drescher, J. Gollub, R. Goldstein, Science **325**, 487 (2009)
158. V. Geyer, F. Jülicher, J. Howard, B. Friedrich, Proc. Natl. Acad. Sci. **110**, 18058 (2013)
159. R. Ma, G. Klindt, I. Riedel-Kruse, F. Jülicher, B. Friedrich, Phys. Rev. Lett. **113**, 048101 (2014)
160. M. Reufer, R. Besseling, J. Schwarz-Linek, V. Martinez, A. Morozov, J. Arlt, D. Trubitsyn, F. Ward, W. Poon, Biophys. J. **106**, 37 (2014)
161. J. Schwarz-Linek, J. Arlt, A. Jepson, A. Dawson, T. Vissers, D. Miroli, T. Pilizota, V.A. Martinez, W.C. Poon, Coll. Surf. B: Biointerfaces **137**, 2 (2016)

162. W.F. Paxton, K.C. Kistler, C.C. Olmeda, A. Sen, S.K. St. Angelo, Y. Cao, T.E. Mallouk, P.E. Lammert, V.H. Crespi, J. Am. Chem. Soc. **126**, 13424 (2004)
163. J.R. Howse, R.A. Jones, A.J. Ryan, T. Gough, R. Vafabakhsh, R. Golestanian, Phys. Rev. Lett. **99**, 048102 (2007)
164. C. Maggi, J. Simmchen, F. Saglimbeni, J. Katuri, M. Dipalo, F. De Angelis, S. Sanchez, R. Di Leonardo, Small **12**, 446 (2016)
165. I. Theurkauff, C. Cottin-Bizonne, J. Palacci, C. Ybert, L. Bocquet, Phys. Rev. Lett. **108**, 268303 (2012)
166. J. Palacci, S. Sacanna, A.P. Steinberg, D.J. Pine, P.M. Chaikin, Science **339**, 936 (2013)
167. T. Vicsek, A. Czirók, E. Ben-Jacob, I. Cohen, O. Shochet, Phys. Rev. Lett. **75**, 1226 (1995)
168. W. Ebeling, F. Schweitzer, B. Tilch, BioSystems **49**, 17 (1999)
169. J. Stenhammar, A. Tiribocchi, R. Allen, D. Marenduzzo, M. Cates, Phys. Rev. Lett. **111**, 145702 (2013)
170. X. Zheng, B. Ten Hagen, A. Kaiser, M. Wu, H. Cui, Z. Silber-Li, H. Löwen, Phys. Rev. E **88**, 032304 (2013)
171. K. Drescher, R. Goldstein, N. Michel, M. Polin, I. Tuval, Phys. Rev. Lett. **105**, 168101 (2010)
172. K. Drescher, J. Dunkel, L. Cisneros, S. Ganguly, R. Goldstein, Proc. Natl. Acad. Sci. **108**, 10940 (2011)
173. A.I. Campbell, S.J. Ebbens, P. Illien, R. Golestanian, arXiv:1802.04600 (2018)
174. R. Nash, R. Adhikari, M. Cates, Phys. Rev. E **77**, 026709 (2008)
175. R.W. Nash, Efficient lattice Boltzmann simulations of self-propelled particles with singular forces, Ph.D. thesis, The University of Edinburgh, 2010
176. N.S. Martys, R.D. Mountain, Phys. Rev. E **59**, 3733 (1999)
177. F. Kümmel, B. ten Hagen, R. Wittkowski, I. Buttinoni, R. Eichhorn, G. Volpe, H. Löwen, C. Bechinger, Phys. Rev. Lett. **110**, 198302 (2013)
178. H. Wensink, V. Kantsler, R. Goldstein, J. Dunkel, Phys. Rev. E **89**, 010302 (2014)
179. P. Ahlrichs, B. Dünweg, J. Chem. Phys. **111**, 8225 (1999)
180. E. Harris, *The chlamydomonas sourcebook: a comprehensive guide to biology and laboratory use* (Elsevier Science, Amsterdam, 2013)
181. V. Kantsler, J. Dunkel, M. Polin, R.E. Goldstein, Proc. Natl. Acad. Sci. **110**, 1187 (2013)
182. D. Röhm, S. Kesselheim, A. Arnold, Soft Matt. **10**, 5503 (2014)
183. J. de Graaf, J. Stenhammar, Phys. Rev. E **95**, 023302 (2017)
184. J. de Graaf, J. Stenhammar, Pers. Commun. (2017)
185. A. Chatterji, J. Horbach, J. Chem. Phys. **122**, 184903 (2005)
186. S.E. Ilse, C. Holm, J. de Graaf, J. Chem. Phys. **145**, 134904 (2016)