ESPResSo 4.0 – An Extensible Software Package for Simulating Soft Matter Systems

Florian Weik,[1,*] 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,†]

[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, P.O. Box 56, Dunedin 9054, New Zealand*

(Dated: October 15, 2018)

ESPResSo is an extensible simulation package for research on soft matter systems. It is a versatile molecular dynamics program, originally developed for coarse-grained simulations of charged soft matter systems [Limbach *et al.*, Comput. Phys. Commun. **174**, 704 (2006)]. The scope of the software has since broadened considerably, and ESPResSo is now 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, as well as aggregation of soot particles in process engineering. ESPResSo includes solvers for hydrodynamics and electrokinetics, both on the continuum and explicit particle level. Since our last description of version 3.1 [Arnold *et al.*, Meshfree Methods for Partial Differential Equations VI, Lecture Notes in Computational Science and Engineering **89** (2013)], ESPResSo underwent a 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 major changes, improvements, and new simulation scenarios.

I. INTRODUCTION

Soft matter [1–3] has been established as an interesting research field for systems ranging from the nanometer scale up to tens of micrometers. A statistical approach may be utilized, as the relevant energy scale of interactions is often comparable to the thermal energy, making Brownian motion and fluctuations an important ingredient. Soft matter touches upon aspects of polymer science [4, 5], colloidal science [6, 7], liquid crystals [8, 9], biological physics [10, 11], food science [12], and material science [13]. Moreover, soft matter science has been a fruitful playground for shaping a better understanding of statistical physics concepts [14, 15] and emergent non-equilibrium phenomena [16–18]. The delicate interplay of energy and entropy 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. This traces its root all the way back to first computer simulations of hard sphere phase separation [19, 20], which utilized statistical physics concepts to great effect.

From these humble beginnings, the use of computer simulations to describe complex systems grew out to a new field that bridges experimental and theoretical efforts. In soft matter, these methods came to their own, as the size and nature of the interactions implies that many of the microscopic freedoms that govern a system may be coarse-grained out. This 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. Such coarse-grained models have played a decisive role in elucidating the guiding principles, *i.e.*, 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 simulation package ESPResSo is particularly well suited for the study of coarse-grained models for soft matter. Here, we would like to stress that we do not aim to study atomistically resolved systems. For such tasks, there are a number of better suited open source softwares available, such as GROMACS[1] [38–40], NAMD[2] [41–43], AMBER[3] [44, 45], to name just a few of the most commonly used ones. LAMPPS[4] [46] and (to a lesser degree) DL_POLY[5] [47] approach ESPResSo the closest in terms of the scope of their simulation methods and features.

The development of ESPResSo began in 2001. The name is an acronym for **E**xtensible **S**imulation **P**ackage for **Res**earch on **So**ft matter systems, which encapsulates the goal of the software project. That is, the original team wished to develop a highly parallel molecular dynamics program that could deal efficiently with charged bead-spring models of polymers for studying polyelectrolyte solutions and charged colloidal suspensions. However, this framework should also facilitate the testing and integration of new algorithms that would enable it to keep up

* fweik@icp.uni-stuttgart.de
† holm@icp.uni-stuttgart.de

[1] http://www.gromacs.org
[2] https://www.ks.uiuc.edu/Research/namd/
[3] http://ambermd.org
[4] https://lammps.sandia.gov
[5] https://www.scd.stfc.ac.uk/Pages/DL_POLY.aspx

with the ever shifting interests of the field. At the time of inception, we had learned to master electrostatic interactions in periodic geometries with great control [48, 49] and had also developed other fast electrostatic solvers for partially periodic geometries [50–56]. ESPResSo has seen the incorporation of many more algorithms since then, allowing it to truly live up to its "extensible" character [57]. However, it is not enough to keep up with newly developed simulation algorithms. A simulation package also has to cope with changing computer architectures, programming models, and users' expectations.

From the start, ESPResSo has had a twofold architecture. Massively parallel simulation routines were implemented in the C programming language using the Message Passing Interface (MPI) to be scalable up to hundreds of processors on large computing clusters. The simulation core exposed a scripting interface to the user, based on the Tcl language which was the most suitable choice at that time. The combination of fast simulation routines with an easy-to-use scripting interface is very useful as it allows programmatic control of the simulation protocol with simultaneous analysis and visualization. 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.

ESPResSo has always been strong in treating Coulomb and dipolar interactions under various boundary conditions. Over time it was extended to support 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 [57–62]. In ESPResSo 4.0 support for using the ScaFaCoS[6] library has been added. It contains several advanced electrostatic and dipolar solvers for various partially periodic dimensions [63, 64]. In recent years, the interest into particle-based methods, especially molecular dynamics simulations, has grown in the engineering community. Their interest lie with modeling and improving large scale physical processes such as soot aggregation [65, 66] and air filtration [67]. Furthermore, functionality to study non-equilibrium phenomena in active matter through the use of self-propelled particles [68–71] has been added to ESPResSo 4.0.

The ESPResSo package has been set up as an open software project from its very beginning, committed to open source development under the GNU General Public License (GPL). Users of ESPResSo are supported through mailing lists[7] and workshops, such as the annual ESPResSo summer schools with 20–40 participants. These workshops are often run as CECAM[8] tutorials, jointly funded through grants from the German Science Foundation through the SFB 716 and the cluster of excellence SimTech (EXC 310). These platforms allow the developer team to announce new features implemented in the code base of ESPResSo to the global research community. The software is used by scientists all over the world, which is documented by more than 425 citations on Web of Science since 2006 as of this writing (604 citations on Google Scholar).

Several research groups have contributed algorithms to the core of ESPResSo. These contributions include: (i) Continuum flow solvers (CPU+GPU) and various methods to couple them with MD [72–78]. (ii) Advanced algorithms for rare-event sampling and the reconstruction of free-energy landscapes [79, 80]. (iii) Monte-Carlo methods [81]. (iv) An interface to MDAnalysis. (v) Extensions to the support for anisotropic particles. (vi) Methods to calculate dipolar interactions on the GPU. Significant effort has been spent 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. The latest version of ESPResSo and its documentation can be found on the website https://www.espressomd.org. Ongoing development is organized through the social code hosting platform GitHub[9], which facilitates contribution handling through its collaborative features.

The aim of this paper is to present the improvements provided by ESPResSo 4.0 compared to the previous version ESPResSo 3.1. In addition, through this article, we would like to encourage users to consider ESPResSo, both for use in soft matter research, and as a platform for method and algorithm development. A thriving user and developer community is key to maintaining and extending a strong and reliable software for research on a wide variety of soft matter systems.

II. 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) [82–84], symbolic mathematics (SymPy, SAGE) [85, 86], data visualization (matplotlib, Mayavi) [87, 88], and numerical mathematics (NumPy, SciPy, pandas) [89, 90]. In ESPResSo 4.0, we have therefore decided to switch the interface scripting language from Tcl to Python. This allows for the quick generation

[6] http://www.scafacos.de
[7] espressomd-users@nongnu.org
[8] https://www.cecam.org/
[9] https://github.com/espressomd/espresso/

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 [89], 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 [89] is very popular in numerical computing, especially 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 a P3M solver object, 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.

III. 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 [91]. 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.

A. Improvements in the simulation core

The ESPResSo code base has been developed continuously for more than 15 years through 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.

B. Testing

ESPResSo contains a test suite to verify the correctness of the code. Hitherto, these tests were run by the maintainers before accepting 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 using this method. It is our experience, 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 [92]. 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 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.

C. Continuous Integration

As mentioned earlier, the development of ESPResSo is organized through the social code hosting platform 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. 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[10]. 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.

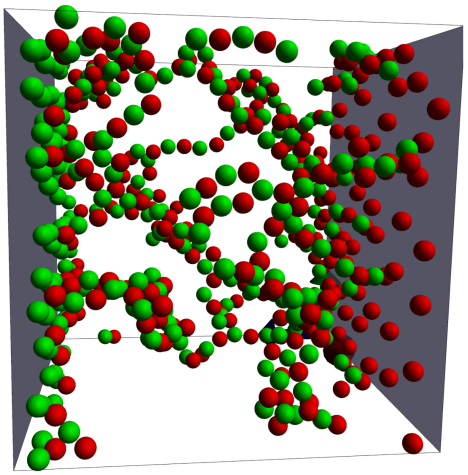


FIG. 1. Example of the OpenGL visualizer of ESPResSo for charged particles in a plate capacitor. Positive (red) and negative (green) ions experience a constant potential boundary condition.

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 [93] 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 detect undefined behavior during runtime, thus often detecting bugs that are hard to discover for a human software developer.

IV. ONLINE VISUALIZATION

ESPResSo features two options for visualizing simulations while they are running, *i.e.*, live or on-line visualization. The first one uses Mayavi [88], 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,

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

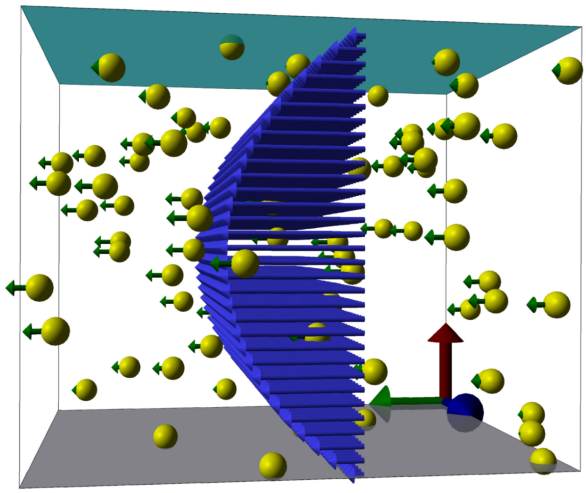


FIG. 2. Example of the OpenGL visualizer of ESPResSo for particles advected in a Poiseuille flow between two walls (Green arrows: Particle velocity; blue arrows: Fluid velocity).

ESPResSo 4.0 comes with a 3D rendering engine that uses PyOpenGL [94, 95]. Both visualizers are not meant 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 Figs. 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 [96] (Fig. 1). It has a large number of options to adjust colors, materials, lightning, camera perspective, and many more. The *OpenGL visualizer* can also be used for offline visualization to create snapshots of completed simulations. Furthermore, most vectorial particle properties can be visualized by 3D arrows on the particles to display forces, velocities, etc.

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 included 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.

V. INTEGRATION WITH EXTERNAL PACKAGES

A. 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 endless online resources and introductory material. Some examples are:

- Mean, variance and median from NumPy
- Random number generation from `numpy.random` and `scipy.stats`, e.g., for setting up a random system
- The use of `numpy.histogram()` for sampling results into histograms, e.g., to obtain effective potentials via Boltzmann inversion
- The confidence interval estimator from `scipy.stats.t.interval()` to control the amount of sampling in a simulation
- Linear algebra methods from `numpy.linalg`, e.g., for calculating the main axes of and moments of inertia
- Defining custom bonded and non-bonded potentials interpolated from NumPy arrays

It is also possible to create Jupyter notebooks 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 are presented in this fashion.

ESPResSo 4.0 also 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.

B. 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, because 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 (P3M) algorithm for fully periodic systems, as well as the electrostatic layer correction (ELC) and MMM2D 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 [63, 97] (**Sca**lable **Fa**st **Co**ulomb **S**olvers). 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 $N \log N$ scaling for periodic, mixed, and open boundary conditions. When this version of the library is used, it can be used as a solver for magnetostatic interactions [64, 98].

VI. PARTICLE-BASED REACTIONS

Weak polyelectrolytes, for example most proteins [99], have titratable groups that can be protonated or deprotonated depending on the pH [100]. The chemical reactions promote phenomena like protonation-configuration coupling [101, 102]. 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 [103, 104]. Here, the net charge of the particles fluctuates around its mean 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 Fig. 3: 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 [105], is distinguished 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 first appeared in the 1990s [106] in the form of the constant-pH ensemble. A little later the reaction ensemble [107, 108] was introduced which allows for the simulation of arbitrary chemical reactions. In both reaction schemes, particle properties (like for example the charge) need to be changed and new particles need to be created or deleted based on probabilistic criteria. The reaction ensemble and the constant-pH

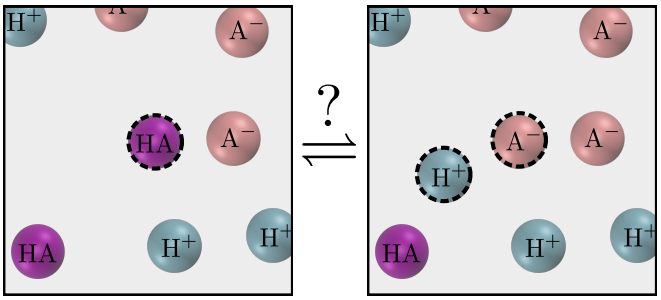


FIG. 3. The protonated/deprotonated states of a Monte-Carlo reaction step. Target particles are circled with a dotted line.

method only differ in the probabilistic criteria [109] and therefore both methods are implemented in ESPResSo 4.0.

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

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

where z chemical species of type s_i with stoichiometric coefficients ν_i are reacting [110]. The acceptance probability in the reaction ensemble for a reaction from state r to l is given by [111]

$$\mathrm{acc}^{\mathrm{RE},\xi}(l|r) = \min\left\{1, (V\Gamma)^{\overline{\nu}\xi} \prod_{i=1}^{z}\left[\frac{N_i^r!}{(N_i^r+\xi\nu_i)!}\right] \exp(-\beta\Delta E_{\mathrm{pot},r\to l})\right\}, \tag{2}$$

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 based reaction constant $\Gamma = c^{\circ\overline{\nu}} \exp(-\beta\Delta G^\circ)$ 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 $\overline{\nu} = \sum_i \nu_i$ 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 [111].

In addition to the above presented standard reaction ensemble algorithm, we implemented a rare event sampling method on top of the reaction ensemble [112], called the Wang-Landau reaction ensemble algorithm, which might prove useful when investigating systems with metastable states and rare transitions between them (*e.g.*, under poor solvent conditions).

In the other chemical equilibrium sampling method which is present in literature—the constant-pH ensemble—the acceptance probability is given by:

$$\mathrm{acc}^{\mathrm{cpH},\xi}(l|r) = \min\big(1, \exp\big[-\beta(\Delta E_{\mathrm{pot}} \pm (\ln(10)/\beta)\,(\mathrm{pH} - \mathrm{pK}_a))\big]\big), \tag{3}$$

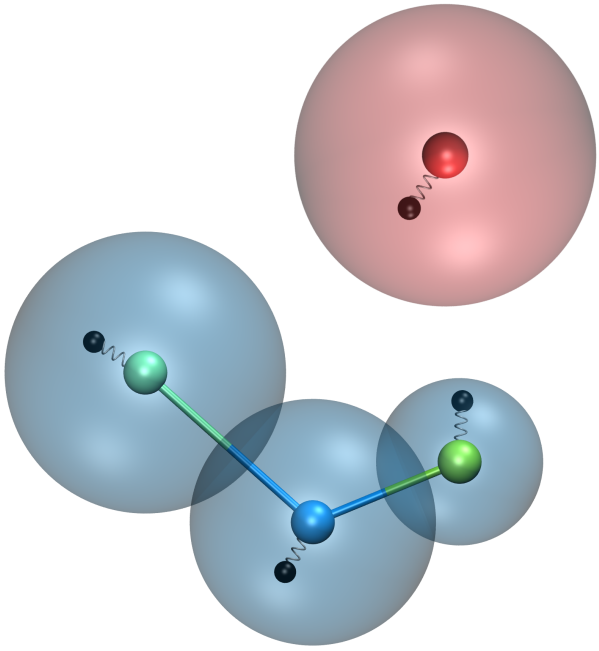


FIG. 4. Sketch of the polarizable coarse grained model for the ionic liquid BMIM PF_6.

where pH is the pH of an implicitly imposed proton reservoir, $\mathrm{pK}_a = -\log_{10}\exp(-\beta\Delta G^\circ)$ 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^-.

The grand canonical simulation scheme [113] 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 Widoms insertion method [113]. 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 [113]

$$\mu^{\mathrm{ex}} = -k_{\mathrm{B}}T \ln\Big(\Big\langle e^{-\beta(E_{\mathrm{pot}}(N+1)-E_{\mathrm{pot}}(N))}\Big\rangle\Big). \qquad (4)$$

VII. INCLUDING EXPLICIT DIPOLAR POLARIZATION WITH DRUDE OSCILLATORS

Thermalized cold Drude oscillators [114] can be used to simulate dynamic particle polarization, often effectively included in the force-field (*e.g.* via reduced charges) to match continuum properties [115]. 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. 4). The energetic minimum of the Drude charge can be obtained self-consistently, which requires several iterations of the system's electrostatics and is usually considered computational expensive [116]. 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 low 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+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 [117]. 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 P3M 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 with the partial charges of the cores $q_d \leftrightarrow q_{\mathrm{partial}}$ within the molecule. This can be done with the *subtracts P3M short-range bond*, which is also used to cancel the electrostatics between Drude core and Drude charge $q_d \leftrightarrow q_{\mathrm{core}}$. 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 P3M. In ESPResSo, helper methods assist setting up this exclusion. In combination with particle polarizability, the *Thole correction* [118] is often used to correct for overestimation of induced dipoles at short distances. Ultimately, it alters the short-range electrostatics of P3M to result in a damped Coulomb interaction potential

$$V(r) = \frac{q_i q_j}{r}\left[1 - e^{-sr}\left(1 + \frac{sr}{2}\right)\right]. \tag{5}$$

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

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

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 P3M accuracy and the split between short-range and long-range electrostatics interaction. ESPResSo assist with the bookkeeping of mixed scaling coefficients and has convenient methods to set up all necessary Thole interactions.

VIII. 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. External fields are decomposed into two parts to increase flexibility. They are described as a global scalar or vectorial field source and a coupling. 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.*,

$$\boldsymbol{F}_i = -\gamma(\boldsymbol{v}_i - \boldsymbol{u}). \tag{7}$$

The field source itself either interpolates user-provided data from a regular grid onto the particle positions, is constant throughout the whole domain, or defined by an affine map A according to

$$\boldsymbol{u}(\boldsymbol{r}) = A\boldsymbol{r} + \boldsymbol{b}, \tag{8}$$

with a user-provided matrix A and shift b.

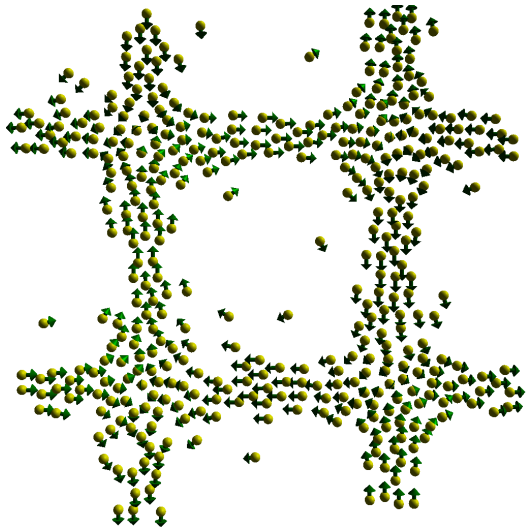


FIG. 5. Example for the use of external fields in ESPResSo 4.0. We show the stationary state of particles in a static Taylor-Green vortex flow. The green arrows denote the particle velocity and direction.

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

$$\begin{aligned} u_x &= \cos(x)\sin(y), \\ u_y &= -\sin(x)\cos(y), \end{aligned} \tag{9}$$

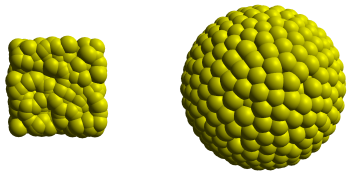
where $\boldsymbol{u}$ is the flow velocity. It has the vorticity

$$\omega \equiv \nabla \times \boldsymbol{u} = 2\left|\cos(x)\cos(y)\right|. \tag{10}$$

In Fig. 5 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, because the particles in the example have inertia. A complete simulation script can be found in the supplementary information.

IX. 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.



(a) (b)

FIG. 6. (a) Initial configuration of the particles filling the raspberry. (b) Configuration after 1000 steps energy relaxation.

It allows relaxing the position as well as the orientation of the particles by running the following iterative scheme:

$$\Delta_i^{x,y,z} = \mathrm{sgn}(F_i^{x,y,z}) \min(\gamma F_i^{x,y,z}, \Delta_{\max}) \quad (11)$$
$$x_i^{x,y,z} = x_i^{x,y,z} + \Delta_i^{x,y,z}, \quad (12)$$

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 i-th 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 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 [92]. This type of coupling limits the maximal coupling strength for one particle which can lead to unrealistic transport properties for larger objects like colloids. One way to overcome this problem is to use so-called raspberry models, 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 [68, 69]. To setup such a raspberry particle, a domain 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, Fig. 6 show 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.

X. 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 [119, 120], clustering of magnetic nanoparticles in magnetic soft matter systems [121–124], and the irreversible agglomeration of soot particles in combustion processes [65, 66, 125]. 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 [126] 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 [122, 123], for the agglomeration of soot particles, the bonds created by the collision algorithms are used [66].

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 storage space and the I/O time, 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 [127], 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 Ref. [128]. 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 [127]. 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

$$\lambda = \frac{\mu_0 m^2}{4\pi\sigma^3 k_\mathrm{B} T}, \tag{13}$$

where μ_0 denotes the vacuum permittivity and m the particles' dipole moment. The dipolar interactions are calculated by means of the dipolar P3M method [129] and the dipolar layer correction [130]. 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 Fig. 7, 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.

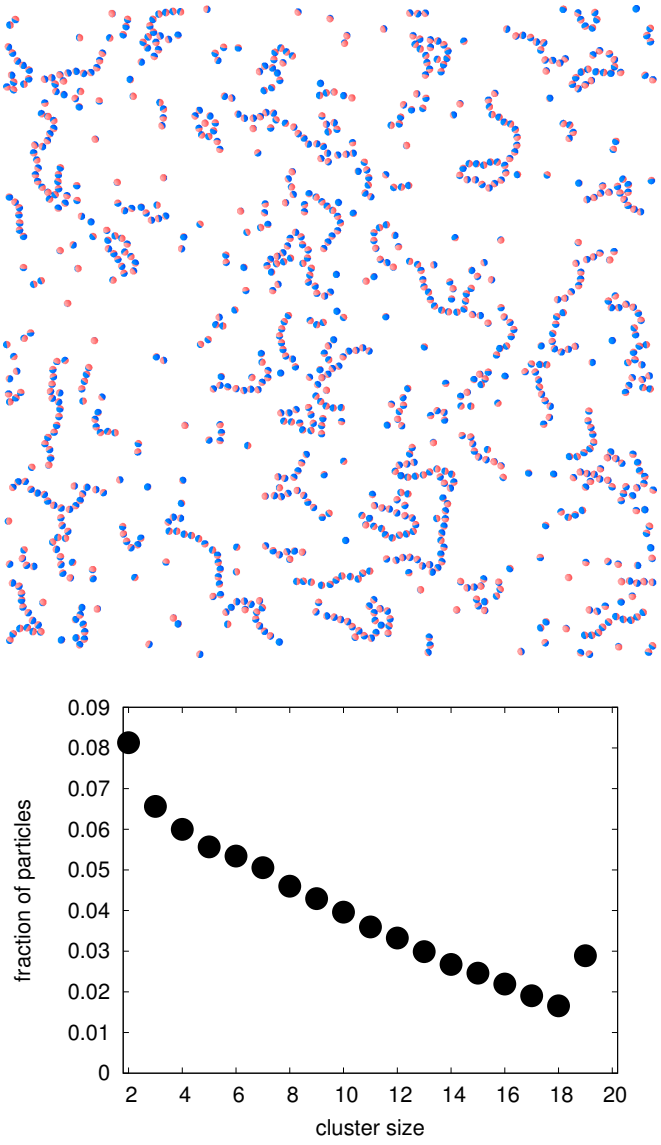


FIG. 7. Top: Snapshot of the ferrofluid monolayer. Bottom: Composition of the suspension plotted as fraction of particles being part of a cluster of a given size.

XI. 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. On a macroscopic length scale, one can think of flocks of birds [131], schools of fish [132], and human crowds [133–135]. On the microscopic level, examples include swarms of bacteria [136–138], sperm [139–141], and algae [142, 143].

In addition to these biological examples, the last decade and a half have seen the rapid development of man-made counterparts that exhibit many of the same remarkable behaviors found in nature. Specifically, a wide range of colloidal self-propelled particles has become available following the first experimental realizations of two types of "chemical swimmers" by Paxton *et al.* [144] and Howse *et al.* [145], respectively. These artificial swimmers eliminate some of the complexity found in living matter, but still display signature features of being out of equilibrium, such as the ability to perform work [146] and motility-induced phase separation [147, 148].

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 [149], followed closely by the active Brownian model [150–152]. In our developments, we have drawn inspiration from the Active Brownian Model 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 [57, 72] to account for the characteristic dipolar flow field that is associated with self-propulsion by microorganisms [153, 154] as well as chemical swimmers [155]. Our implementation [70] is similar to the sub-lattice approach introduced by Nash *et al.* [156, 157]. We refer to this extension as the hydrodynamic ALM (HALM) and we will briefly touch upon it here.

In the following, we assume that the swimmers are shape-anisotropic without axisymmetry so that we obtain a general form of the dynamics. ALM is then defined by the following equation of motion for the translation of the

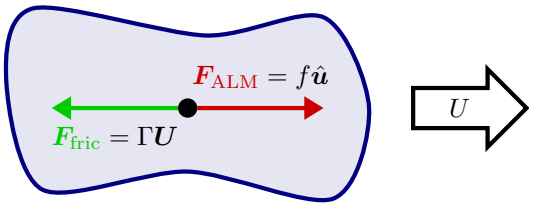


FIG. 8. A two-dimensional schematic representation of a single shape-anisotropic, self-propelled particle that is simulated using the Active Langevin Method (ALM). The particle achieves motion through the application of a constant force f directed along a unit vector $\hat{\boldsymbol{u}}$, *i.e.*, $\boldsymbol{F}_{\mathrm{ALM}} = f\hat{\boldsymbol{u}}$. This force balances against the friction experienced in this direction, $\boldsymbol{F}_{\mathrm{fric}} = \underline{\boldsymbol{\Gamma}}_t \boldsymbol{U}$, with $U = |\boldsymbol{U}|$ the resulting swim speed.

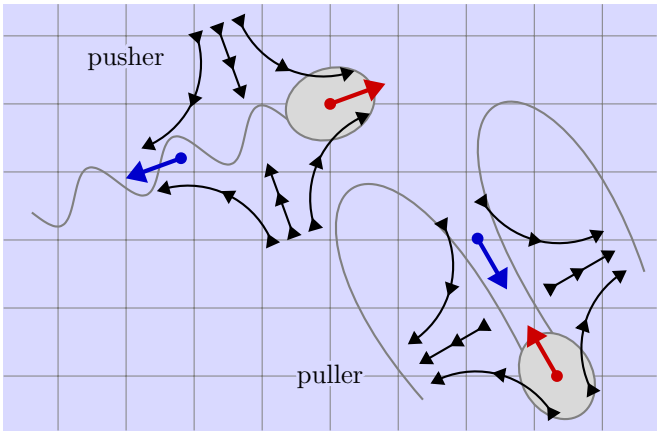


FIG. 9. Sketches of two types of force-free hydrodynamic ALM (HALM) swimmers. Pusher and puller swimmers result from a change in the placement of the counter force (blue arrow) with respect to the swimmer's center, on which a propulsive force (red arrow) is applied. These forces are resolved sublattice (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 Fig. 4.1 from R. Nash's PhD thesis [157] is intentional and meant to emphasize the origin of our HALM developments.

i-th swimmer's center of mass $\boldsymbol{r}_i$:

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

Here, M is the swimmer's mass; $\underline{\boldsymbol{\Gamma}}_t$ is the translational diffusion coefficient matrix, which accounts for shape anisotropy; $\hat{\boldsymbol{u}}_i$ denotes the swimmer's orientational unit vector, which co-moves and co-rotates; f is the self-propulsion force; $\boldsymbol{\nabla}$ denotes the gradient; V is a pair potential depending on the particle separation $r_{ij} \equiv |\boldsymbol{r}_i - \boldsymbol{r}_j|$, and the orientation of both particles, specified here by quaternions $\boldsymbol{Q}_i$ and $\boldsymbol{Q}_j$. Thermal noise is introduced via $\boldsymbol{\xi}_{i,t}(t)$, which satisfies $\langle\boldsymbol{\xi}_{i,t}(t)\rangle = \boldsymbol{0}$ and $\langle\boldsymbol{\xi}_{i,t}(t)\otimes\boldsymbol{\xi}_{j,t}(t')\rangle = 6k_{\mathrm{B}}T\underline{\boldsymbol{\Gamma}}_t\delta_{ij}\delta(t-t')$, with $\otimes$ the dyadic product, δ_{ij} the Kronecker delta, and δ the one-dimensional (1D) delta distribution; $\langle\cdots\rangle$ indicates time averaging. The equation of motion for quaternions is lengthy and well-described in Ref. [158], it is therefore not reproduced here.

The above set of equations may be cast into words as follows, with Fig. 8 illustrating the idea behind ALM for a generic shape-anisotropic particle. 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, as introduced via the coupling of the translational friction coefficient matrix to the particle velocity $\boldsymbol{F}_{\mathrm{fric}} = \underline{\boldsymbol{\Gamma}}_t\boldsymbol{U}$, leading to persistent directed motion with swim speed $U = |\boldsymbol{U}|$. The direction of self-propulsion may be changed by rotational Brownian motion or by torques that act on the particle either through applied external fields or via collisions.

ESPResSo 4.0 allows one to chose different masses and friction matrices for the individual swimmers, providing users with tremendous flexibility in creating mixtures of swimmers with a range of mobilities. Coupling between translational and rotational degrees of freedom through the action of a full friction tensor is not included in the latest release, hence the form of Eq. (14). In its most general incarnation ALM would allow for such coupling and this would enable the simulation of, *e.g.*, ensembles of L-shaped [159] and chiral swimmers [160].

It should be noted that ALM does not have a fully damped dynamics, which is uncommon in simulating colloidal particles. That is, Langevin dynamics permits simulation of molecular and nanoscopically small particles, where the ballistic regime of the thermal noise becomes relevant to the dynamics of the particles. It is debatable whether one can reasonably ignore details of the self-propulsion mechanism in this limit. ALM approaches the Active Brownian Model when the friction coefficient becomes very large, making the study of the colloidal regime possible. Care should be taken with the size of the time step in taking this limit, such that the algorithm produces physically correct results.

Finally, the reason for introducing ALM here, rather than use the more common Active Brownian Model, is related to the way particles in ESPResSo couple to the LB fluid. Using ALM and HALM in tandem, the effect of the hydrodynamic interactions between swimmers may be disentangled without introducing additional differences in the dynamics, in much the same way as this can be done for passive particles [161]. We will turn to a description of the HALM algorithm next.

ALM may be extended to HALM by coupling to a LB fluid to account for hydrodynamic interactions between swimmers, particles, and obstacles, as follows. Self-propulsion is a momentum-free process, *i.e.*, motion may be achieved without the action of an external force. A

force / counter-force pair is applied on the LB fluid to ensure this condition and induce a flow field with the typical leading-order dipolar contribution. Figure 9 illustrates two such combinations leading to the well-known puller and pusher dipolar flow fields. Common representatives of the puller flow field are bi-flagellate algae such as *Chlamydomonas reinhardtii* [153, 162]. Pusher flow fields are realized for bull sperm and other mammalian spermatozoa [163]. We refer to Ref. [70] for full details of the implementation and we will only touch upon the essentials of the HALM algorithm in the following.

HALM resolves the position of the force and counter-force pair sub-lattice, similar to the method introduced by Nash *et al.* [156, 157], as illustrated in Fig. 9. However, unlike in Refs. [156, 157] we do not permit the separation between the two points to go below one LB cell length in our algorithm for reasons of numerical stability. In LB, a single point particle gains an effective radius in coupling to a LB fluid [164], or rather an inherent friction. This effect causes the swimmer to move at a constant speed under the action of the self-propulsion force. However, lattice coupling does not lead to an effective rotational component, thus a point particle does not experience rotation due to local vorticity of the fluid. Again following Refs. [156, 157], one can account for rotation of the swimmer in an external flow field by using an approximation to Faxén's laws. That is, the angular velocity of the swimmer is related to the hydrodynamic torque applied to the sphere, which can be approximated using a finite difference. In our implementation, we found that this approximation leads to substantial discretization artifacts, even when employing higher-order interpolation schemes [70]. This effect can be ameliorated, as we will come back to in the last paragraph to this section.

It should be noted that careful tuning of the relevant hydrodynamic parameters and the self-propulsion forces is key to reproducing low-Reynolds-number hydrodynamic solutions, we refer to Ref. [165] for an in-depth discussion. The authors of Ref. [165] have verified, but not published, that the implementation by Nash *et al.* [156, 157] and our HALM algorithm have the same near-field flow characteristics and long-range hydrodynamic retardation effects. This gives confidence that despite HALM having an inertial component to its dynamics, parameters can be selected for which this does not affect the behavior of the model swimmers.

ALM and HALM can both be used in conjunction with the raspberry method of creating shape-anisotropic particles [68, 69, 166, 167]. This enables, for example, the study of the effect of polarization and roughness on motility-induced clustering [168]. 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 [70]. This property has been exploited to determine the effect of these hydrodynamic moments on the motion of these swimmers in confining geometries [71], as well as their interaction with passive particles in their surrounding [70, 165]. The use of raspberry particles in combination with HALM removes most of the lattice artifacts that point-particle HALM suffers from. 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.

XII. PARALLEL OUTPUT

Different options are available for writing simulation trajectories to file. We still support ESPResSo's VTF format which outputs system and particle properties as straightforward 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.

In ESPResSo 4.0, there is the added possibility to write files using the H5MD specification [169] which utilizes the HDF5 binary format [170]. HDF5 is intended to be self-contained and therefore enhances 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, which also supports parallel input/output [171]. In typical ESPResSo simulations, 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 [172]. Simulation trajectories may also be visualized in VMD [173] 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 be traced back to the ESPResSo Python script, which helps tremendously in reproducing data.

XIII. 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 https://espressomd.org, or on GitHub https://github.com/espressomd/.

ACKNOWLEDGMENTS

We would like to acknowledge the over 100 researchers who contributed over the last fifteen years to the ESPResSo software, and whose names can be found on our website https://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 [174], 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 CONTRIBUTIONS

All co-authors contributed to the preparation of this manuscript. 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 http://github.com/espressomd/espresso/graphs/contributors. Funding for ESPResSo was obtained by CH, who also directed the project.

[1] P. G. de Gennes, Reviews of Modern Physics **64**, 645 (1992).
[2] M. Doi, *Soft Matter Physics* (Oxford University Press, 2013).
[3] J.-L. Barrat and J.-P. Hansen, *Basic Concepts fo Simple and Complex Liquids* (Cambridge University Press, 2003) iSBN: 0-521-78953-2.
[4] M. Doi and S. F. Edwards, *The Theory of Polymer Dynamics*, Vol. 73 (Oxford University Press, 1988).
[5] M. Rubinstein and R. H. Colby, *Polymer Physics* (Oxford University Press, Oxford, UK, 2003).
[6] E. J. Verwey and J. T. G. Overbeek, *Theory of the stability of Lyophobic Colloids* (Elsevier, Amsterdam, 1948).
[7] H. Löwen, Journal of Physics: Condensed Matter **13**, R415 (2001).
[8] S. Chandrasekhar, *Liquid Crystals* (Cambridge, 1992).
[9] I. V. Hamley, *Introduction to Soft Matter* (Wiley, 2003) iSBN: 0-471-89952-6.
[10] P. Nelson, *Biological Physics - Energy, Information, Life* (Freeman, 2004) iSBN: 0-7167-4372-8.
[11] I. Levental, P. C. Georges, and P. A. Janmey, Soft Matter **3**, 299 (2007).
[12] J. Ubbink, A. Burbidge, and R. Mezzenga, Soft Matter **4**, 1569 (2008).
[13] S. Polarz and M. Antonietti, Chemical Communications , 2593 (2002).
[14] K. Kroy and E. Frey, Annalen der Physik **14**, 20 (2005).
[15] D. Frenkel, Science **296**, 65 (2002).
[16] U. Seifert, Reports on Progress in Physics **75**, 126001 (2012).
[17] M. E. Cates, Reports on Progress in Physics **75**, 042601 (2012).
[18] É. Fodor and M. Marchetti, Physica A: Statistical Mechanics and its Applications **504**, 106 (2018).
[19] B. J. Alder and T. E. Wainwright, Journal of Chemical Physics **27**, 1208 (1957).
[20] B. Alder and T. Wainwright, Physical Review **127**, 359 (1962).
[21] P. Pusey, W. C. Poon, S. Ilett, and P. Bartlett, Journal of Physics: Condensed Matter **6**, A29 (1994).
[22] M. A. Bates and D. Frenkel, The Journal of chemical physics **109**, 6193 (1998).
[23] R. van Roij, M. Dijkstra, and J.-P. Hansen, Physical Review 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, and A. van Blaaderen, Nature **437**, 235 (2005).
[25] P. J. Camp, J. C. Shelley, and G. N. Patey, Physical Review Letters **84**, 115 (2000).
[26] Z. Wang, C. Holm, and H. W. Müller, Physical Review E **66**, 021405 (2002).
[27] S. H. L. Klapp and M. Schoen, Journal of Molecular Liquids **109**, 55 (2004), sixth Liblice Conference on the Statistical Mechanics of Liquids.
[28] M. Klinkigt, R. Weeber, S. Kantorovich, and C. Holm, Soft Matter **9**, 3535 (2013).

[29] M. J. Stevens and K. Kremer, Physical Review Letters **71**, 2228 (1993).
[30] A. V. Dobrynin, M. Rubinstein, and S. P. Obukhov, Macromolecules **29**, 2974 (1996).
[31] U. Micka, C. Holm, and K. Kremer, Langmuir **15**, 4033 (1999).
[32] H. J. Limbach, C. Holm, and K. Kremer, Europhysics Letters **60**, 566 (2002).
[33] S. Schneider and P. Linse, The European Physical Journal E **8**, 457 (2002).
[34] Q. Yan and J. J. de Pablo, Physical Review Letters **91**, 018301 (2003).
[35] B. A. Mann, R. Everaers, C. Holm, and K. Kremer, Europhysics Letters **67**, 786 (2004).
[36] R. Weeber, S. Kantorovich, and C. Holm, Soft Matter **8**, 9923 (2012).
[37] R. Weeber, M. Hermes, A. M. Schmidt, and C. Holm, Journal of Physics: Condensed Matter **30**, 063002 (2018).
[38] H. J. C. Berendsen, D. van der Spoel, and R. van Drunen, Computer Physics Communications **91**, 43 (1995).
[39] D. van der Spoel, E. Lindahl, B. Hess, G. Groenhof, A. E. Mark, and H. J. C. Berendsen, Journal of Computational Chemistry **26**, 1701 (2005).
[40] 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, and E. Lindahl, Bioinformatics **29**, 845 (2013).
[41] M. Nelson, W. Humphrey, A. Gursoy, A. Dalke, L. Kale, R. D. Skeel, and K. Schulten, International Journal of Supercomputer Applications **10**, 251 (1996).
[42] J. C. Phillips, R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R. D. Skeel, L. Kale, and K. Schulten, Journal of Computational Chemistry **26**, 1781 (2005).
[43] J. C. Phillips, J. E. Stone, K. L. Vandivort, T. G. Armstrong, J. M. Wozniak, M. Wilde, and K. Schulten, in *Proceedings of the 1st First Workshop for High Performance Technical Computing in Dynamic Languages* (IEEE Press, 2014) pp. 6–17.
[44] D. A. Pearlman, D. A. Case, J. W. Caldwell, W. R. Ross, I. T. E. Cheatham, S. DeBolt, D. Ferguson, G. Seibel, and P. Kollman, Computer Physics Commications **91**, 1 (1995).
[45] D. A. Case, T. E. Cheatham, T. Darden, H. Gohlke, R. Luo, K. M. Merz, A. Onufriev, C. Simmerling, B. Wang, and R. Woods, Journal of Computational Chemistry **26**, 1668 (2005).
[46] S. J. Plimpton, Journal of Computational Physics **117**, 1 (1995).
[47] W. Smith and T. Forester, Journal of Molecular Graphics **14**, 136 (1996).
[48] M. Deserno and C. Holm, Journal of Chemical Physics **109**, 7678 (1998).
[49] M. Deserno and C. Holm, Journal of Chemical Physics **109**, 7694 (1998).
[50] A. Arnold and C. Holm, Computer Physics Communications **148**, 327 (2002).
[51] A. Arnold and C. Holm, Chemical Physics Letters **354**, 324 (2002).
[52] A. Arnold, J. de Joannis, and C. Holm, Journal of Chemical Physics **117**, 2496 (2002).
[53] A. Arnold, J. de Joannis, and C. Holm, Journal of Chemical Physics **117**, 2503 (2002).
[54] A. Arnold and C. Holm, in *Advanced Computer Simulation Approaches for Soft Matter Sciences II*, Advances in Polymer Sciences, Vol. II, edited by C. Holm and K. Kremer (Springer, Berlin, 2005) pp. 59–109.
[55] A. Arnold and C. Holm, Journal of Chemical Physics **123**, 144103 (2005).
[56] A. Arnold, B. A. Mann, and C. Holm, in *Computer Simulations in Condensed Matter: from Materials to Chemical Biology*, Lecture Notes in Physics, Vol. 703, edited by M. Ferrario, G. Ciccotti, and K. Binder (Springer, Berlin, Germany, 2006) pp. 193–222.
[57] A. Arnold, O. Lenz, S. Kesselheim, R. Weeber, F. Fahrenberger, D. Röhm, P. Košovan, and C. Holm, in *Meshfree Methods for Partial Differential Equations VI*, Lecture Notes in Computational Science and Engineering, Vol. 89, edited by M. Griebel and M. A. Schweitzer (Springer Berlin Heidelberg, 2013) pp. 1–23.
[58] S. Tyagi, A. Arnold, and C. Holm, Journal of Chemical Physics **127**, 154723 (2007).
[59] S. Tyagi, A. Arnold, and C. Holm, Journal of Chemical Physics **129**, 204102 (2008).
[60] C. Tyagi, M. Süzen, M. Sega, M. Barbosa, S. S. Kantorovich, and C. Holm, The Journal of Chemical Physics **132**, 154112 (2010).
[61] A. Arnold, K. Breitsprecher, F. Fahrenberger, S. Kesselheim, O. Lenz, and C. Holm, Entropy **15**, 4569 (2013).
[62] F. Fahrenberger and C. Holm, Physical Review E **90**, 063304 (2014).
[63] 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, and G. Sutmann, Physical Review E **88**, 063308 (2013).
[64] R. Weeber, F. Nestler, F. Weik, M. Pippig, D. Potts, and C. Holm, Arxiv pre-print arXiv:1808.10341 (2018).
[65] G. Inci, A. Arnold, A. Kronenburg, and R. Weeber, Aerosol Science and Technology **48**, 842 (2014).
[66] G. Inci, A. Kronenburg, R. Weeber, and D. Pflüger, Flow, Turbulence and Combustion , 1 (2017).
[67] C. Schober, D. Keerl, M. J. Lehmann, and M. Mehl, in *European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS 2016)* (ECCOMAS, Hersonissos, Greece, 2016).
[68] L. P. Fischer, T. Peter, C. Holm, and J. de Graaf, The Journal of Chemical Physics **143**, 084107 (2015).
[69] J. de Graaf, T. Peter, L. P. Fischer, and C. Holm, The Journal of Chemical Physics **143**, 084108 (2015).
[70] J. de Graaf, H. Menke, A. J. Mathijssen, M. Fabritius, C. Holm, and T. N. Shendruk, The Journal of Chemical Physics **144**, 134106 (2016).
[71] J. de Graaf, A. J. Mathijssen, M. Fabritius, H. Menke, C. Holm, and T. N. Shendruk, Soft Matter **12**, 4704 (2016).
[72] D. Röhm and A. Arnold, The European Physical Journal Special Topics **210**, 89 (2012).
[73] I. Cimrák, M. Gusenbauer, and T. Schrefl, Computers & Mathematics with Applications **64**, 278 (2012).
[74] I. Cimrak, M. Gusenbauer, and I. Jančigová, Computer Physics Communications **185**, 900 (2014).
[75] I. Cimrák, I. Jancigová, K. Bachratá, and H. Bachratỳ, in *III International Conference on Particle-based Methods–Fundamentals and Applications PARTICLES*, Vol. 2013 (2013) pp. 133–144.

[76] G. Rempfer, G. B. Davies, C. Holm, and J. de Graaf, The Journal of Chemical Physics **145**, 044901 (2016).
[77] A. Guckenberger, M. P. Schraml, P. G. Chen, M. Leonetti, and S. Gekle, Computer Physics Communications **207**, 1 (2016).
[78] C. Bächer, L. Schrack, and S. Gekle, Physical Review Fluids **2**, 013102 (2017).
[79] K. Kratzer, A. Arnold, and R. J. Allen, Journal of Chemical Physics **138**, 164112 (2013).
[80] K. Kratzer, J. T. Berryman, A. Taudt, J. Zeman, and A. Arnold, Computer Physics Communications **185**, 1875 (2014).
[81] S. Samin, Y. Tsori, and C. Holm, Physical Review E **87** (2013), 10.1103/PhysRevE.87.052128.
[82] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, in *NIPS-W* (2017).
[83] 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, and E. Duchesnay, Journal of Machine Learning Research **12**, 2825 (2011).
[84] F. Chollet *et al.*, "Keras," https://keras.io (2015).
[85] 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, and A. Scopatz, PeerJ Computer Science **3**, e103 (2017).
[86] T. S. Developers, *SageMath, the Sage Mathematics Software System (Version x.y.z)* (2018), http://www.sagemath.org.
[87] J. D. Hunter, Computing In Science & Engineering **9**, 90 (2007).
[88] P. Ramachandran and G. Varoquaux, Computing in Science & Engineering **13**, 40 (2011).
[89] E. Jones, T. Oliphant, P. Peterson, *et al.*, "SciPy: Open source scientific tools for Python," (2001–).
[90] W. McKinney, in *Proceedings of the 9th Python in Science Conference*, edited by S. van der Walt and J. Millman (2010) pp. 51 – 56.
[91] "boost c++ libraries," https://www.boost.org ().
[92] B. Dünweg and A. J. C. Ladd, in *Advanced Computer Simulation Approaches for Soft Matter Sciences III*, Advances in Polymer Science, Vol. 221 (Springer-Verlag Berlin, Berlin, Germany, 2009) pp. 89–166.
[93] "Clang: a c language family frontend for llvm," https://clang.llvm.org/ ().
[94] M. Fletcher and R. Liebscher, (2005).
[95] D. Shreiner, *OpenGL Reference Manual: The Official Reference Document to OpenGL, Version 1.2*, 3rd ed. (Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999).
[96] D. Röhm, *Lattice Boltzmann Simulations on GPUs*, Diplomarbeit, University of Stuttgart (2011).
[97] M. Bolten, F. Fahrenberger, R. Halver, F. Heber, M. Hofmann, I. Kabadshow, O. Lenz, M. Pippig, and G. Sutmann, "ScaFaCoS, C subroutine library," http://scafacos.github.com.
[98] F. Nestler, Applied Numerical Mathematics **105**, 25 (2016).
[99] W. Richtering, Smart Colloidal Materials **133**, 9 (2006).
[100] J. M. Berg, ed., *Biochemistry*, 8th ed. (Freeman, New York, NY (USA), 2015).
[101] M. Castelnovo, P. Sens, and J.-F. Joanny, European Physical Journal E: Soft Matter **1**, 115 (2000).
[102] C. Shi, J. A. Wallace, and J. K. Shen, Biophysical Journal **102**, 1590 (2012).
[103] M. Lund and B. Jönsson, Biochemistry **44**, 5722 (2005).
[104] M. Lund and B. Jönsson, Quarterly Reviews of Biophysics **46**, 265 (2013).
[105] H. J. Limbach, A. Arnold, B. A. Mann, and C. Holm, Computer Physics Communications **174**, 704 (2006).
[106] C. E. Reed and W. F. Reed, Journal of Chemical Physics **96**, 1609 (1992).
[107] E. R. Smith, Journal of Statistical Physics **77**, 449 (1994).
[108] J. K. Johnson, A. Z. Panagiotopoulos, and K. E. Gubbins, Molecular Physics **81**, 717 (1994).
[109] J. Landsgesell, C. Holm, and J. Smiatek, The European Physical Journal Special Topics **226**, 725 (2017).
[110] P. W. Atkins and J. de Paula, *Physical Chemistry* (Oxford Univ. Press, Oxford (UK), 2010).
[111] C. Heath Turner, J. K. Brennan, M. Lisal, W. R. Smith, J. Karl Johnson, and K. E. Gubbins, Molecular Simulation **34**, 119 (2008).
[112] J. Landsgesell, C. Holm, and J. Smiatek, Journal of Chemical Theory and Computation **13**, 852 (2017).
[113] D. Frenkel and B. Smit, *Understanding Molecular Simulation*, 1st ed. (Academic Press, San Diego, 1996).
[114] G. Lamoureux and B. Roux, Journal of Chemical Physics **119**, 3025 (2003).
[115] M. Kohagen, P. E. Mason, and P. Jungwirth, The Journal of Physical Chemistry B **120**, 1454 (2016), pMID: 26172524, https://doi.org/10.1021/acs.jpcb.5b05221.
[116] H. Yu, T. Hansson, and W. F. van Gunsteren, The Journal of Chemical Physics **118**, 221 (2003).
[117] P. Mitchell and D. Fincham, Journal of Physics: Condensed Matter **5**, 1031 (1993).
[118] B. T. Thole, Chemical Physics **59**, 341 (1981).
[119] M. Radu and T. Schilling, Europhysics Letters **105**, 26001 (2014).
[120] K. Kratzer and A. Arnold, Soft Matter **11**, 2174 (2015).
[121] K. Butter, P. H. H. Bomans, P. M. Frederik, G. J. Vroege, and A. P. Philipse, Nature Materials **2**, 88 (2003).
[122] J. J. Cerdà, S. Kantorovich, and C. Holm, Journal of Physics: Condensed Matter **20**, 204125 (2008).
[123] R. Weeber, M. Klinkigt, S. Kantorovich, and C. Holm, Journal of Chemical Physics **139**, 214901 (2013).
[124] J. G. Donaldson and S. S. Kantorovich, Nanoscale **7**, 3217 (2015).
[125] A. Attili, F. Bisetti, M. E. Mueller, and H. Pitsch, Combustion and Flame **161**, 1849 (2014).
[126] W. Lechner and C. Dellago, The Journal of Chemical Physics **129**, 114707 (2008).
[127] J. D. Weeks, D. Chandler, and H. C. Andersen, The Journal of Chemical Physics **54**, 5237 (1971).
[128] J. J. Cerdà, T. Sintes, and K. Sumithra, Journal of Chemical Physics **123**, 204703 (2005).
[129] J. J. Cerdà, V. Ballenegger, O. Lenz, and C. Holm, Journal of Chemical Physics **129**, 234104 (2008).
[130] A. Bródka, Chemical Physics Letters **400**, 62 (2004).
[131] M. Ballerini, N. Cabibbo, R. Candelier, A. Cavagna, E. Cisbani, I. Giardina, V. Lecomte, A. Orlandi, G. Parisi, A. Procaccini, M. Viale, and V. Zdravkovic, Proceedings of the National Academy of Sciences **105**,

1232 (2008).
[132] Y. Katz, K. Tunstrøm, C. Ioannou, C. Huepe, and I. Couzin, Proceedings of the National Academy of Sciences **108**, 18720 (2011).
[133] D. Helbing, I. Farkas, and T. Vicsek, Nature **407**, 487 (2000).
[134] J. Zhang, W. Klingsch, A. Schadschneider, and A. Seyfried, in *Traffic and Granular Flow '11*, edited by V. V. Kozlov, A. P. Buslaev, A. S. Bugaev, M. V. Yashina, A. Schadschneider, and M. Schreckenberg (Springer (Berlin/Heidelberg), 2013) p. 241.
[135] J. Silverberg, M. Bierbaum, J. Sethna, and I. Cohen, Physical Review Letters **110**, 228701 (2013).
[136] A. Sokolov, I. S. Aranson, J. O. Kessler, and R. E. Goldstein, Physical Review Letters **98**, 158102 (2007).
[137] M. Reufer, R. Besseling, J. Schwarz-Linek, V. Martinez, A. Morozov, J. Arlt, D. Trubitsyn, F. Ward, and W. Poon, Biophysical Journal **106**, 37 (2014).
[138] J. Schwarz-Linek, J. Arlt, A. Jepson, A. Dawson, T. Vissers, D. Miroli, T. Pilizota, V. A. Martinez, and W. C. Poon, Colloids and Surfaces B: Biointerfaces **137**, 2 (2016).
[139] D. Woolley, Reproduction **126**, 259 (2003).
[140] I. Riedel, K. Kruse, and J. Howard, Science **309**, 300 (2005).
[141] R. Ma, G. Klindt, I. Riedel-Kruse, F. Jülicher, and B. Friedrich, Physical Review Letters **113**, 048101 (2014).
[142] M. Polin, I. Tuval, K. Drescher, J. Gollub, and R. Goldstein, Science **325**, 487 (2009).
[143] V. Geyer, F. Jülicher, J. Howard, and B. Friedrich, Proceedings of the National Academy of Sciences **110**, 18058 (2013).
[144] W. F. Paxton, K. C. Kistler, C. C. Olmeda, A. Sen, S. K. St. Angelo, Y. Cao, T. E. Mallouk, P. E. Lammert, and V. H. Crespi, Journal of the American Chemical Society **126**, 13424 (2004).
[145] J. R. Howse, R. A. Jones, A. J. Ryan, T. Gough, R. Vafabakhsh, and R. Golestanian, Physical Review Letters **99**, 048102 (2007).
[146] C. Maggi, J. Simmchen, F. Saglimbeni, J. Katuri, M. Dipalo, F. De Angelis, S. Sanchez, and R. Di Leonardo, Small **12**, 446 (2016).
[147] I. Theurkauff, C. Cottin-Bizonne, J. Palacci, C. Ybert, and L. Bocquet, Physical Review Letters **108**, 268303 (2012).
[148] J. Palacci, S. Sacanna, A. P. Steinberg, D. J. Pine, and P. M. Chaikin, Science **339**, 936 (2013).
[149] T. Vicsek, A. Czirók, E. Ben-Jacob, I. Cohen, and O. Shochet, Physical Review Letters **75**, 1226 (1995).
[150] W. Ebeling, F. Schweitzer, and B. Tilch, BioSystems **49**, 17 (1999).
[151] J. Stenhammar, A. Tiribocchi, R. Allen, D. Marenduzzo, and M. Cates, Physical Review Letters **111**, 145702 (2013).
[152] X. Zheng, B. ten Hagen, A. Kaiser, M. Wu, H. Cui, Z. Silber-Li, and H. Löwen, Physical Review E **88**, 032304 (2013).
[153] K. Drescher, R. Goldstein, N. Michel, M. Polin, and I. Tuval, Physical Review Letters **105**, 168101 (2010).
[154] K. Drescher, J. Dunkel, L. Cisneros, S. Ganguly, and R. Goldstein, Proceedings of the National Academy of Sciences **108**, 10940 (2011).
[155] A. I. Campbell, S. J. Ebbens, P. Illien, and R. Golestanian, arXiv preprint arXiv:1802.04600 (2018).
[156] R. Nash, R. Adhikari, and M. Cates, Physical Review E **77**, 026709 (2008).
[157] R. W. Nash, *Efficient lattice Boltzmann simulations of self-propelled particles with singular forces*, Ph.D. thesis, The University of Edinburgh (2010).
[158] N. S. Martys and R. D. Mountain, Physical Review E **59**, 3733 (1999).
[159] F. Kümmel, B. ten Hagen, R. Wittkowski, I. Buttinoni, R. Eichhorn, G. Volpe, H. Löwen, and C. Bechinger, Physical Review Letters **110**, 198302 (2013).
[160] H. Wensink, V. Kantsler, R. Goldstein, and J. Dunkel, Physical Review E **89**, 010302 (2014).
[161] D. Röhm, S. Kesselheim, and A. Arnold, Soft Matter **10**, 5503 (2014).
[162] E. Harris, *The Chlamydomonas Sourcebook: A Comprehensive Guide to Biology and Laboratory Use* (Elsevier Science, 2013).
[163] V. Kantsler, J. Dunkel, M. Polin, and R. E. Goldstein, Proceedings of the National Academy of Sciences **110**, 1187 (2013).
[164] P. Ahlrichs and B. Dünweg, Journal of Chemical Physics **111**, 8225 (1999).
[165] J. de Graaf and J. Stenhammar, Physical Review E **95**, 023302 (2017).
[166] V. Lobaskin and B. Dünweg, New Journal of Physics **6**, 54 (2004).
[167] A. Chatterji and J. Horbach, Journal of Chemical Physics **122**, 184903 (2005).
[168] S. E. Ilse, C. Holm, and J. de Graaf, The Journal of Chemical Physics **145**, 134904 (2016).
[169] P. de Buyl, P. H. Colberg, and F. Höfling, Computer Physics Communications **185**, 1546 (2014).
[170] The HDF Group, "Hierarchical data format, version 5," http://www.hdfgroup.org/HDF5/.
[171] "h5xx - a template-based c++ wrapper for the hdf5 library," https://github.com/h5md/h5xx ().
[172] A. Collette, *Python and HDF5* (O'Reilly, 2013).
[173] VMD, "Visual molecular dynamics – homepage," http://www.ks.uiuc.edu/Research/vmd.
[174] M. Smiljanic, R. Weeber, D. Pflüger, C. Holm, and A. Kronenburg, Submitted to EPJST (2018).