The Journal of Open Source Software

ESPResSo++: A Fast and Extensible Molecular Simulation Package for Coarse-Grained Models

Zhen-Hao Xu[4], James Vance[4], Nikita Tretyakov[4], Sebastian Eibl[2], Pavel Kus[2], Jakub Krajniak[6], Tristan Bereau[5], Horacio V. Guzman[7], Bin Song[1], Markus Rampp[2], Torsten Stuehn[1], and Christoph Junghans[3]

1 Max Planck Institute for Polymer Research, Mainz, Germany **2** Max Planck Computing and Data Facility, Garching, Germany **3** Los Alamos National Laboratory, Los Alamos, USA **4** Johannes Gutenberg University of Mainz, Mainz, Germany **5** Heidelberg University, Heidelberg, Germany **6** Independent researcher, Poznań, Poland **7** Institut de Ciència de Materials, Barcelona, Spain

DOI: 10.21105/joss.11072

Software

- Review
- Repository
- Archive

Editor: Patrick Diehl

Reviewers:

- @tingyu66
- @jngrad

Submitted: 07 March 2026
Published: 21 September 2026

Summary

ESPResSo++ is an open-source software package for molecular dynamics (MD) simulations with a particular emphasis on coarse-grained (CG) models of soft matter systems (Praprotnik et al., 2008). Written in C++ with a flexible Python interface, it is designed for high-performance computing (HPC) environments and supports massively parallel simulations through MPI. The package enables simulations of polymers, membranes, colloids and complex fluids with a wide range of interaction models and advanced algorithms.

ESPResSo++ builds upon the experience of its predecessor ESPResSo, but provides a cleaner, more modular codebase and enhanced extensibility. It is actively developed by an international community of researchers across multiple institutions and disciplines.

Statement of need

Molecular dynamics simulations are essential tools for exploring the behavior of soft matter systems at mesoscopic scales. General-purpose MD codes such as GROMACS (Abraham et al., 2015) and LAMMPS (Thompson et al., 2022) are primarily optimized for all-atom simulations and may require significant customization for coarse-grained models that need non-standard interactions or specialized multiscale algorithms. ESPResSo++ addresses this gap by providing:

- A modular and extensible design, enabling researchers to easily implement new interaction potentials and integrators.
- Efficient parallelization for large-scale simulations of complex systems.
- A rich library of coarse-grained interaction models and algorithms tailored to soft matter.
- A Python-based scripting interface for ease of use, reproducibility, and coupling with external analysis tools.
- Multi-scale simulation techniques such as AdResS and Lees-Edwards.

State of the field

Several established MD packages serve the soft matter and coarse-grained simulation communities. **LAMMPS** (Thompson et al., 2022) is a general-purpose, highly scalable code with broad force field support and a flexible plugin architecture; however, its input scripting language is less suited to complex CG workflows, and adaptive resolution is not a core feature. **GROMACS** (Abraham et al., 2015) provides exceptional performance for standard force

fields and supports the Martini CG model ecosystem, but adding truly novel CG interactions requires modifying the C++ source, and its file-based workflow is less interactive than a Python API. **HOOMD-blue** (Anderson et al., 2020) is the most directly comparable package, offering a Python-native API and GPU-accelerated CG simulations for soft matter; however, it lacks support for adaptive resolution methods. **OpenMM** (Eastman et al., 2017) excels at GPU-accelerated simulations and allows custom force definitions via algebraic expressions, but targets primarily biomolecular systems and lacks MPI-based multi-node parallelization. The sibling project **ESPResSo** (Weik et al., 2019) shares common roots but focuses on charged systems, hydrodynamic coupling (lattice Boltzmann), and electrokinetics rather than multiscale resolution bridging.

ESPResSo++ occupies a distinct niche as a package purpose-built for coarse-grained and multiscale soft matter simulations. Its key differentiator is first-class support for adaptive resolution simulation (AdResS), which allows seamless coupling of atomistic and coarse-grained representations within a single simulation. ESPResSo++ is currently the only actively maintained MD package that provides AdResS as a core feature; none of the other packages listed above offer this capability. It is worth noting that these capabilities were also contributed to existing general-purpose codes (Fritsch et al., 2012; Junghans & Poblete, 2010; Nagarajan et al., 2013), but got removed again later, hence a dedicated package allows tighter integration of multiscale algorithms with the domain decomposition, neighbor list construction, and force computation pipeline — design choices that would be difficult to retrofit into architectures optimized for all-atom throughput.

Software design

ESPResSo++ uses a layered C++/Python architecture. The performance-critical computational kernel (force calculations, integration, domain decomposition, and communication) is implemented in C++17 and exposed to Python through Boost.Python bindings. Each Python-exposed C++ class provides a static `registerPython()` method, and a central registration dispatcher ensures that the full object hierarchy is available as the `espressopp` Python module. This design lets users assemble, configure, and control simulations entirely from Python while retaining the performance of compiled C++ for the inner loops.

Modularity through templates and extensions.
Interactions are implemented using C++ class templates parameterized by a potential type (e.g., `VerletListInteractionTemplate<_Potential>`), so that adding a new pairwise potential requires only implementing the `computeEnergy()` and `computeForce()` methods of a `Potential` subclass. The integrator follows a similar pattern: an `MDIntegrator` base class exposes an `addExtension()` mechanism through which thermostats, barostats, constraints, and adaptive resolution layers are attached via Boost.Signals2 callbacks. This signal-based coupling keeps extensions decoupled from the integration loop and from each other.

Parallelization.
ESPResSo++ employs spatial domain decomposition with MPI. The simulation box is partitioned across MPI ranks using a `NodeGrid`, and each rank further subdivides its domain into linked cells via a `CellGrid`. Ghost particle exchange handles communication of boundary data. A non-blocking variant (`DomainDecompositionNonBlocking`) overlaps communication with computation, and the heterogeneous spatial domain decomposition algorithm (HeSpaDDA) (Guzman et al., 2017) provides load balancing for spatially inhomogeneous systems.

Performance optimizations.
Since version 2.0 (Guzman et al., 2019), ESPResSo++ has been modernized with SIMD vectorization through a structure-of-arrays (SOA) particle data layout (`ParticleArray`) with 64-byte alignment, yielding an overall three-times speedup for short-range non-bonded force calculations (Vance et al., 2023). An improved cell decomposition scheme (Yao et al., 2004) allows sub-decomposition into cells with a length of half or a third of the cutoff radius, reducing

the number of unnecessary distance calculations.

Testing and documentation.
The code is tested through a combination of Boost.Test (C++) and Python `unittest` test suites, executed via CMake/CTest and continuous integration on GitHub Actions. User documentation is built with Sphinx; developer API documentation with Doxygen.

Research impact statement

ESPResSo++ has enabled research across a broad range of soft matter topics over more than a decade of active development. It has been used in numerous peer-reviewed studies, including investigations of:

- Polymer rheology and entanglement effects (Grommes et al., 2021, 2022, 2024, 2025; Grommes & Reith, 2020; Hsu & Kremer, 2020, 2023, 2024; Lee & Paul, 2020; Ohkuma et al., 2023; Singh et al., 2020; Tubiana et al., 2021; Zhao, Singh, et al., 2020)
- Polymer concepts in Polysomes organization (Kobayashi & Guzman, 2026)
- Lipid membranes, protein and vesicle dynamics (Bause & Bereau, 2021; Papež et al., 2023; Zhao, Cortes-Huerto, et al., 2020)
- Adaptive resolution simulations (Bevc et al., 2013; Fiorentini et al., 2020; Thaler et al., 2020)
- Ionic liquids under shear flow (Gholami et al., 2025; Zhang et al., 2021)
- Coarse-grained conformational surface hopping (Rudzinski & Bereau, 2020)

The project is developed across multiple institutions — Max Planck Institute for Polymer Research, Max Planck Computing and Data Facility, Los Alamos National Laboratory, Johannes Gutenberg University of Mainz, and Heidelberg University — with contributions from both current and former developers documented in the AUTHORS file. The codebase has a public development history spanning more than ten years on GitHub, with continuous integration, code coverage tracking, and versioned releases.

Functionality

Key features of ESPResSo++ include:

- **Inter-particle interactions**: Lennard-Jones, Coulomb, soft repulsive, bonded interactions, tabulated potentials, and more.
- **Algorithms**: Molecular dynamics, Langevin dynamics, dissipative particle dynamics (DPD), Brownian dynamics, adaptive resolution simulations (AdResS), Monte Carlo sampling.
- **Electrostatics**: Particle–particle particle–mesh (P3M), Ewald summation, and other long-range methods.
- **Parallelization**: Domain decomposition using MPI, optimized for massively parallel architectures.
- **Python interface**: Full simulation control and analysis scripting in Python.
- **Extensibility**: Modular design allows easy addition of new force fields, integrators, or analysis tools.

New Features since ESPResSo++ v2.0

Since the last major release of ESPResSo++ v2.0 in 2018 (Guzman et al., 2019) a number of new functionalities and features have been added, including:

- **SIMD vectorization and related optimizations**: enhance compute performance on modern CPUs (Vance et al., 2023)

- **Cell decomposition**: allow sub-decomposition into cells with a length of half or a third of the cutoff for direct force calculations (Yao et al., 2004)
- **HeSpaDDA**: heterogeneous spatial domain decomposition algorithm (HeSpaDDA) for larger scale simulations (Guzman et al., 2017)
- **new potentials and simulation methods**: AngularCosineSquared, TabulatedSubEnsAngular, surface hopping MD, Lees-Edwards boundary conditions
- **Checkpoint the state of the random number generator (RNG)**: allow restarting from checkpointed state of RNG
- **I/O**: support for parallel writing and reading of H5MD checkpoints
- **Python 3 compatibility**

Example usage

A minimal Python script to run a simple (repulsive only) Lennard-Jones type particle simulation in ESPResSo++ looks like:

```
import espressopp
from mpi4py import MPI

# simulation system parameters
num_particles = 10000      # total number of particles in the system
box           = (20,20,20) # size of the simulationbox (all length are in sigma)
rc            = 1.12246    # cut off for the short range non bonded potential
skin          = 0.3        # skin used for verlet neighbor list
dt            = 0.005      # time step for 1 md step
epsilon       = 1.0        # energy unit
sigma         = 1.0        # length unit
temperature   = 1.0        # temperature of the simulation
LJcaprad      = 0.8        # initial capping radius for LJ interaction
                           # for random configurations


# system setup
system         = espressopp.System()
system.rng     = espressopp.esutil.RNG()
system.bc      = espressopp.bc.OrthorhombicBC(system.rng, box)
system.skin    = skin

# define underlying storage system for parallelisation
nodeGrid       = espressopp.tools.decomp.nodeGrid(MPI.COMM_WORLD.size,box,rc,skin)
cellGrid       = espressopp.tools.decomp.cellGrid(box, nodeGrid, rc, skin)
system.storage = espressopp.storage.DomainDecomposition(system, nodeGrid, cellGrid)

# interaction setup, here short range non-bonded Lennard Jones potential
interaction    = espressopp.interaction.VerletListLennardJonesCapped(
                   espressopp.VerletList(system, cutoff=rc))
interaction.setPotential(type1=0, type2=0,
                         potential=espressopp.interaction.LennardJonesCapped(
                           epsilon, sigma, rc, shift='auto', caprad=LJcaprad))
system.addInteraction(interaction)

# integrator setup
integrator     = espressopp.integrator.VelocityVerlet(system)

# thermostat setup
thermostat             = espressopp.integrator.LangevinThermostat(system)
```

```
thermostat.gamma       = 1.0
thermostat.temperature = temperature
integrator.addExtension(thermostat)

# create random particle setup in the simulation box
props = ['id', 'type', 'mass', 'pos', 'v']
new_particles = []
pid = 1
while pid <= num_particles:
    type = 0
    mass = 1.0
    pos  = system.bc.getRandomPos()
    vel  = espressopp.Real3D(0.0, 0.0, 0.0)
    part = [pid, type, mass, pos, vel]
    new_particles.append(part)
    if pid % 1000 == 0:
        system.storage.addParticles(new_particles, *props)
        system.storage.decompose()
        new_particles = []
    pid += 1
system.storage.addParticles(new_particles, *props)

integrator.dt = 0.0001 # very small timestep for initial warmup
for n in range(20):
  # warmup finished, switch to uncapped Lennard Jones potential and increase timestep dt
  if n == 10:
    interaction = espressopp.interaction.VerletListLennardJones(
                    espressopp.VerletList(system, cutoff=rc))
    interaction.setPotential(type1=0, type2=0,
                             potential=espressopp.interaction.LennardJones(
                               epsilon, sigma, rc, shift='auto'))
    system.removeInteraction(0)
    system.addInteraction(interaction)
    integrator.dt = dt
  integrator.run(10000)
  Etot = system.getInteraction(0).computeEnergy()
  print("md time = {:4.1f}, total energy: {:10.2f}".format(integrator.dt*n*10000, Etot))

# write PDB file of (quasi) equilibrated LJ system.
# At this temperature it is more or less crystallized.
espressopp.tools.pdbwrite("simplelj.pdb", system, molsize=num_particles)
```

AI usage disclosure

No generative AI tools were used in the creation of the ESPResSo++ software or its documentation. During the preparation of this manuscript, AI-assisted tools were used for copyediting and formatting. All content was reviewed and verified by the authors.

Acknowledgements

We thank the ESPResSo++ developer community and all contributors listed in the AUTHORS file. ESPResSo++ has been supported by the Transregio TRR146 of the German Research Foundation. The ESPResSo++ project is supported by the U.S. Department of Energy through

Los Alamos National Laboratory (LANL). Los Alamos National Laboratory is operated by Triad National Security, LLC, for the National Nuclear Security Administration of the U.S. Department of Energy (contract no. 89233218CNA000001). This paper has been assigned a Los Alamos Unlimited Release number of LA-UR-26-21700.

Abraham, M. J., Murtola, T., Schulz, R., Páll, S., Smith, J. C., Hess, B., & Lindahl, E. (2015). GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers. *SoftwareX*, *1–2*, 19–25. https://doi.org/10.1016/j.softx.2015.06.001

Anderson, J. A., Glaser, J., & Glotzer, S. C. (2020). HOOMD-blue: A python package for high-performance molecular dynamics and hard particle monte carlo simulations. *Computational Materials Science*, *173*, 109363. https://doi.org/10.1016/j.commatsci.2019.109363

Bause, M., & Bereau, T. (2021). Reweighting non-equilibrium steady-state dynamics along collective variables. *The Journal of Chemical Physics*, *154*(13). https://doi.org/10.1063/5.0042972

Bevc, S., Junghans, C., Kremer, K., & Praprotnik, M. (2013). Adaptive resolution simulation of salt solutions. *New Journal of Physics*, *15*(10), 105007. https://doi.org/10.1088/1367-2630/15/10/105007

Eastman, P., Swails, J., Chodera, J. D., McGibbon, R. T., Zhao, Y., Beauchamp, K. A., Wang, L.-P., Simmonett, A. C., Harrigan, M. P., Stern, C. D., Wiewiora, R. P., Brooks, B. R., & Pande, V. S. (2017). OpenMM 7: Rapid development of high performance algorithms for molecular dynamics. *PLOS Computational Biology*, *13*(7), e1005659. https://doi.org/10.1371/journal.pcbi.1005659

Fiorentini, R., Kremer, K., & Potestio, R. (2020). Ligand-protein interactions in lysozyme investigated through a dual-resolution model. *Proteins: Structure, Function, and Bioinformatics*, *88*(10), 1351–1360. https://doi.org/10.1002/prot.25954

Fritsch, S., Junghans, C., & Kremer, K. (2012). Structure formation of toluene around C60: Implementation of the adaptive resolution scheme (AdResS) into GROMACS. *Journal of Chemical Theory and Computation*, *8*(2), 398–403. https://doi.org/10.1021/ct200706f

Gholami, A., Kloth, S., Xu, Z.-H., Kremer, K., Vogel, M., Stuehn, T., & Rudzinski, J. F. (2025). Structure and dynamics of ionic liquids under shear flow. *The Journal of Chemical Physics*, *163*(7). https://doi.org/10.1063/5.0279946

Grommes, D., Bruch, O., Imhof, W., & Reith, D. (2025). Coarse-grained molecular dynamics study of the melting dynamics in long alkanes. *Polymers*, *17*(18). https://doi.org/10.3390/polym17182500

Grommes, D., Bruch, O., & Reith, D. (2024). Mimicking polymer processing conditions on the meso-scale: Relaxation and crystallization in polyethylene systems after uni- and biaxial stretching. *Molecules*, *29*(14), 3391. https://doi.org/10.3390/molecules29143391

Grommes, D., & Reith, D. (2020). Determination of relevant mechanical properties for the production process of polyethylene by using mesoscale molecular simulation techniques. *Soft Materials*, *18*(2–3), 242–261. https://doi.org/10.1080/1539445x.2020.1722692

Grommes, D., Schenk, M. R., Bruch, O., & Reith, D. (2021). Investigation of crystallization and relaxation effects in coarse-grained polyethylene systems after uniaxial stretching. *Polymers*, *13*(24), 4466. https://doi.org/10.3390/polym13244466

Grommes, D., Schenk, M. R., Bruch, O., & Reith, D. (2022). Initial crystallization effects in coarse-grained polyethylene systems after uni- and biaxial stretching in blow-molding cooling scenarios. *Polymers*, *14*(23), 5144. https://doi.org/10.3390/polym14235144

Guzman, H. V., Junghans, C., Kremer, K., & Stuehn, T. (2017). Scalable and fast heterogeneous molecular simulation with predictive parallelization schemes. *Physical*

Review E, *96*, 053311. https://doi.org/10.1103/PhysRevE.96.053311

Guzman, H. V., Tretyakov, N., Kobayashi, H., Fogarty, A. C., Kreis, K., Krajniak, J., Junghans, C., Kremer, K., & Stuehn, T. (2019). ESPResSo++ 2.0: Advanced methods for multiscale molecular simulation. *Computer Physics Communications*, *238*, 66–76. https://doi.org/10.1016/j.cpc.2018.12.017

Hsu, H.-P., & Kremer, K. (2020). Efficient equilibration of confined and free-standing films of highly entangled polymer melts. *The Journal of Chemical Physics*, *153*(14). https://doi.org/10.1063/5.0022781

Hsu, H.-P., & Kremer, K. (2023). Glass transition temperature of (ultra-)thin polymer films. *The Journal of Chemical Physics*, *159*(7). https://doi.org/10.1063/5.0165902

Hsu, H.-P., & Kremer, K. (2024). Entanglement-stabilized nanoporous polymer films made by mechanical deformation. *Macromolecules*, *57*(6), 2998–3012. https://doi.org/10.1021/acs.macromol.4c00187

Junghans, C., & Poblete, S. (2010). A reference implementation of the adaptive resolution scheme in ESPResSo. *Computer Physics Communications*, *181*(8), 1449–1454. https://doi.org/10.1016/j.cpc.2010.04.013

Kobayashi, H., & Guzman, H. V. (2026). Self-induced dimensional reduction and scaling transition of mRNA in polysomes: A multiscale simulation study. *The Journal of Chemical Physics*, *164*(16), 164907. https://doi.org/10.1063/5.0320598

Lee, E., & Paul, W. (2020). Additional entanglement effect imposed by small sized ring aggregates in supramolecular polymer melts: Molecular dynamics simulation study. *Macromolecules*, *53*(5), 1674–1684. https://doi.org/10.1021/acs.macromol.9b02209

Nagarajan, A., Junghans, C., & Matysiak, S. (2013). Multiscale simulation of liquid water using a four-to-one mapping for coarse-graining. *Journal of Chemical Theory and Computation*, *9*(11), 5168–5175. https://doi.org/10.1021/ct400566j

Ohkuma, T., Hagita, K., Murashima, T., & Deguchi, T. (2023). Miscibility and exchange chemical potential of ring polymers in symmetric ring–ring blends. *Soft Matter*, *19*(21), 3818–3827. https://doi.org/10.1039/d3sm00108c

Papež, P., Merzel, F., & Praprotnik, M. (2023). Rotational dynamics of a protein under shear flow studied by the eckart frame formalism. *The Journal of Physical Chemistry B*, *127*(33), 7231–7243. https://doi.org/10.1021/acs.jpcb.3c02324

Praprotnik, M., Junghans, C., Delle Site, L., & Kremer, K. (2008). Simulation approaches to soft matter: Generic statistical properties vs. Chemical details. *Computer Physics Communications*, *179*(1–3), 51–60. https://doi.org/10.1016/j.cpc.2008.01.018

Rudzinski, J. F., & Bereau, T. (2020). Coarse-grained conformational surface hopping: Methodology and transferability. *The Journal of Chemical Physics*, *153*(21). https://doi.org/10.1063/5.0031249

Singh, M. K., Hu, M., Cang, Y., Hsu, H.-P., Therien-Aubin, H., Koynov, K., Fytas, G., Landfester, K., & Kremer, K. (2020). Glass transition of disentangled and entangled polymer melts: Single-chain-nanoparticles approach. *Macromolecules*, *53*(17), 7312–7321. https://doi.org/10.1021/acs.macromol.0c00550

Thaler, S., Praprotnik, M., & Zavadlav, J. (2020). Back-mapping augmented adaptive resolution simulation. *The Journal of Chemical Physics*, *153*(16). https://doi.org/10.1063/5.0025728

Thompson, A. P., Aktulga, H. M., Berger, R., Bolintineanu, D. S., Brown, W. M., Crozier, P. S., Veld, P. J. in 't, Kohlmeyer, A., Moore, S. G., Nguyen, T. D., Shan, R., Stevens, M. J., Tranchida, J., Trott, C., & Plimpton, S. J. (2022). LAMMPS - a flexible simulation tool for

particle-based materials modeling at the atomic, meso, and continuum scales. *Computer Physics Communications*, *271*, 108171. https://doi.org/10.1016/j.cpc.2021.108171

Tubiana, L., Kobayashi, H., Potestio, R., Dünweg, B., Kremer, K., Virnau, P., & Daoulas, K. (2021). Comparing equilibration schemes of high-molecular-weight polymer melts with topological indicators. *Journal of Physics: Condensed Matter*, *33*(20), 204003. https://doi.org/10.1088/1361-648x/abf20c

Vance, J., Xu, Z.-H., Tretyakov, N., Stuehn, T., Rampp, M., Eibl, S., Junghans, C., & Brinkmann, A. (2023). Code modernization strategies for short-range non-bonded molecular dynamics simulations. *Computer Physics Communications*, *290*, 108760. https://doi.org/10.1016/j.cpc.2023.108760

Weik, F., Weeber, R., Szuttor, K., Breitsprecher, K., Graaf, J. de, Kuber, M., Kuron, J., McNamara, S. P., Menzel, A., & Holm, C. (2019). ESPResSo 4.0 – an extensible software package for simulating soft matter systems. *The European Physical Journal Special Topics*, *227*(14), 1789–1816. https://doi.org/10.1140/epjst/e2019-800186-9

Yao, Z., Wang, J.-S., Liu, G.-R., & Cheng, M. (2004). Improved neighbor list algorithm in molecular simulations using cell decomposition and data sorting method. *Computer Physics Communications*, *161*(1), 27–35. https://doi.org/10.1016/j.cpc.2004.04.004

Zhang, Z., Krajniak, J., & Ganesan, V. (2021). A multiscale simulation study of influence of morphology on ion transport in block copolymeric ionic liquids. *Macromolecules*, *54*(11), 4997–5010. https://doi.org/10.1021/acs.macromol.1c00025

Zhao, Y., Cortes-Huerto, R., Kremer, K., & Rudzinski, J. F. (2020). Investigating the conformational ensembles of intrinsically disordered proteins with a simple physics-based model. *The Journal of Physical Chemistry B*, *124*(20), 4097–4113. https://doi.org/10.1021/acs.jpcb.0c01949

Zhao, Y., Singh, M. K., Kremer, K., Cortes-Huerto, R., & Mukherji, D. (2020). Why do elastin-like polypeptides possibly have different solvation behaviors in water–ethanol and water–urea mixtures? *Macromolecules*, *53*(6), 2101–2110. https://doi.org/10.1021/acs.macromol.9b02123