The Journal of Open Source Software

TrixiParticles.jl: Particle-based multiphysics simulation in Julia

Niklas S. Neher [1*], **Erik Faulhaber** [2*], **Sven Berger** [3*], **Gregor J. Gassner** [2], **and Michael Schlottke-Lakemper** [1,4]

1 High-Performance Computing Center Stuttgart, University of Stuttgart, Germany **2** Department of Mathematics and Computer Science, University of Cologne, Germany **3** Institute of Surface Science, Helmholtz-Zentrum hereon, Germany **4** High-Performance Scientific Computing, University of Augsburg, Germany * These authors contributed equally.

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Rohit Goswami

Reviewers:

- @luraess
- @giordano
- @williamfgc

Submitted: 03 July 2024
Published: unpublished

Summary

TrixiParticles.jl is a Julia-based open-source package for particle-based multiphysics simulations and part of the Trixi Framework (Schlottke-Lakemper et al., 2021). It handles complex geometries and specialized applications, such as computational fluid dynamics (CFD) and structural dynamics, by providing a versatile platform for particle-based methods. TrixiParticles.jl allows for the straightforward addition of new particle systems and their interactions, facilitating the setup of coupled multiphysics simulations such as fluid-structure interaction (FSI). Furthermore, simulations are set up directly with Julia code, simplifying the integration of custom functionalities and promoting rapid prototyping.

Here, we give a brief overview of the software package TrixiParticles.jl, starting with the scientific background before going on to describe the functionality and benefit in more detail. Finally, exemplary results and implemented features are briefly presented.

Statement of need

Numerical simulations, such as CFD, structural mechanics, thermodynamics, or magneto-hydrodynamics, pose several challenges when simulating real-world problems. For example, they involve complex geometries, free surfaces, deformable boundaries, and moving material interfaces, as well as the coupling of multiple systems with different mathematical models.

One way to address these challenges is to use particle-based methods, in which the particles either represent physical particles or mathematical interpolation points. The former case refers to methods that model separate, discrete particles with rotational degrees of freedom such as the Discrete Element Method (DEM) proposed by (Cundall & Strack, 1979), whereas the latter case refers to methods such as Smoothed Particle Hydrodynamics (SPH), which is a numerical discretization method for solving problems in continuum mechanics. SPH was originally developed by (Gingold & Monaghan, 1977) to simulate astrophysical applications and is now widely used to simulate CFD, structural mechanics, and even heat conduction problems.

The Lagrangian formalism in particle-based methods allows particles to move along a velocity field without any connection to neighboring particles, thus eliminating the need for a mesh to discretize the simulation domain. This mesh-free approach simplifies the preprocessing, making it particularly suitable for simulating complex geometries and also facilitates simulations of large deformations and movements. By representing each material with its own set of particles, coupling multiple different physical systems into a single multiphysics setup is straightforward.

In addition, particle-based methods are inherently suited to simulating free surfaces, material interfaces, and moving boundaries.

There are several open-source software projects specialized for SPH methods, including DualSPHysics (Domínguez et al., 2021), SPlisHSPlasH (Bender & others, 2024), and SPHinXsys (Zhang et al., 2021), written in C++, and PySPH (Prabhu Ramachandran et al., 2021), written in Python. These softwares utilize the advantages of the SPH methods to simulate problems such as FSI and free surfaces (O'Connor & Rogers, 2021) or complex geometries (Laha et al., 2024).

TrixiParticles.jl is written in the Julia programming language and combines the advantage of easy and rapid prototyping with the ability for high-performance computing using multicore parallelization and hardware accelerators. It provides support for developing and testing new SPH methods and also for simulating and coupling other particle-based methods such as DEM. Since simulations are configured and set up using only Julia code, custom methods or particle interactions can be added without modifying the original source code.

Overview of particle-based simulation

In TrixiParticles.jl, particles of a single particle-based method, e.g. SPH or DEM, are grouped into a *system*. The interaction between two particles is defined entirely by the types of their systems. This approach makes it easy to support new methods and different physics by adding a new system and defining its pairwise interaction with other systems.

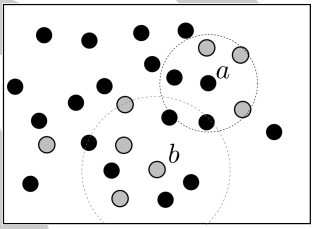

Figure 1: Particles of two different systems $\mathcal{S}_1$ and $\mathcal{S}_2$ in a simulation domain. The black and gray dashed circles represent the search radii for neighbors of particles a and b, respectively.

To illustrate this, Figure 1 depicts particles within a simulation domain. The black particles belong to system $\mathcal{S}_1$ and the gray particles belong to system $\mathcal{S}_2$. In general, the force f_a experienced by a particle a is calculated as

$$f_a = \sum_{b \in \mathcal{S}_1} f_{ab}^{\mathcal{S}_1} + \sum_{b \in \mathcal{S}_2} f_{ab}^{\mathcal{S}_2} + \cdots + \sum_{b \in \mathcal{S}_n} f_{ab}^{\mathcal{S}_n},$$

where the interaction force $f_{ab}^{\mathcal{S}_i}$ that particle a experiences due to particle b depends on the system type of particle a, the system type $\mathcal{S}_i$ of particle b, and the relative particle distance. For computational efficiency, only particles with a distance within a system-dependent search radius interact with each other.

For example, the SPH method determines the force between two SPH particles according to (Monaghan, 2005) as

$$f_{ab} = -m_a m_b \left(\frac{p_a}{\rho_a^2} + \frac{p_b}{\rho_b^2} \right) \nabla_a W_{ab} + \Pi_{ab},$$

where m_a, m_b, ρ_a, ρ_b, p_a, p_b are the mass, density, and pressure of particles a and b, respectively. The last term Π_{ab} includes dissipative terms such as artificial viscosity (Monaghan, 2005) and is scheme-specific. The weighting function W_{ab}, also called kernel-function, depends on the relative distance between particles a and b.

Code structure

Figure 2 depicts the basic building blocks of TrixiParticles.jl. A simulation essentially consists of spatial discretization (left block) and time integration (center block). For the latter, the Julia package OrdinaryDiffEq.jl is used. The callbacks (right block) provide additional functionality and communicate with the time integration method during the simulation.

The semidiscretization couples the systems of a simulation and also manages the corresponding neighborhood searches for each system. The resulting ordinary differential equation (ODE) problem is then fed into the time integrator and is solved by an appropriate numerical time integration scheme.

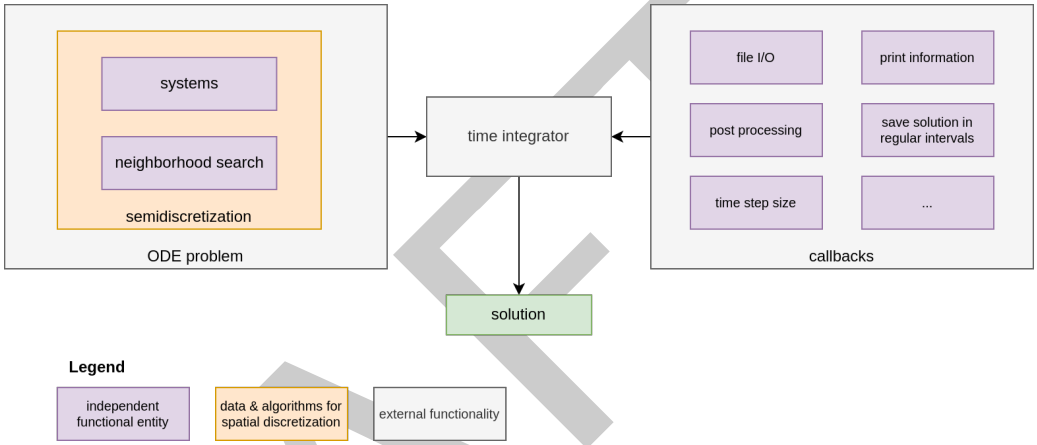

Figure 2: Main building blocks of TrixiParticles.jl.

Features

At the time of writing, the following feature highlights are available in TrixiParticles.jl:

- *Fluid Systems*
  - Weakly compressible SPH (WCSPH): Standard SPH method originally developed by (Gingold & Monaghan, 1977) to simulate astrophysics applications.
  - Entropically damped artificial compressibility (EDAC) for SPH: As opposed to the WCSPH scheme, which uses an equation of state, this scheme uses a pressure evolution equation to calculate the pressure, which is derived by (Clausen, 2013) and adapted to SPH by (P. Ramachandran & Puri, 2019).
- *Structure Systems*
  - Total lagrangian SPH (TLSPH): Method to simulate elastic structures where all quantities are calculated with respect to the initial configuration (O'Connor & Rogers, 2021).
  - DEM: Discretization of granular matter or bulk material into a finite set of distinct, interacting mass elements (Bićanić, 2004), (Cundall & Strack, 1979).
- *Boundary Systems*
  - Boundary system with several boundary models, where each model follows a different interaction rule.
  - Open boundary system to simulate non-reflecting (open) boundary conditions (Lastiwka et al., 2009).
- *Correction methods and models*
  - Density diffusion (Antuono et al., 2010)
  - Transport-velocity formulation (TVF) (Adami et al., 2013)
  - Intra-particle-force surface tension (Akinci et al., 2013)
- *Performance and parallelization*

- Shared memory parallelism using multithreading
- Highly optimized neighborhood search providing various approaches
- GPU support

TrixiParticles.jl is open source and available under the MIT license at GitHub, along with detailed documentation on how to use it. Additionally, we provide tutorials explaining how to set up a simulation of fluid flows, structure mechanics, or FSI. A collection of simulation setups to get started with can be found in the examples directory.

As one of the validation examples, Figure 3 compares SPH results of TrixiParticles.jl and (O'Connor & Rogers, 2021) against benchmark data from the finite element simulation of (Turek & Hron, 2007). The plots show the y-deflection of the tip of a beam oscillating under its own weight. The results obtained with TrixiParticles.jl match those of (O'Connor & Rogers, 2021) well.

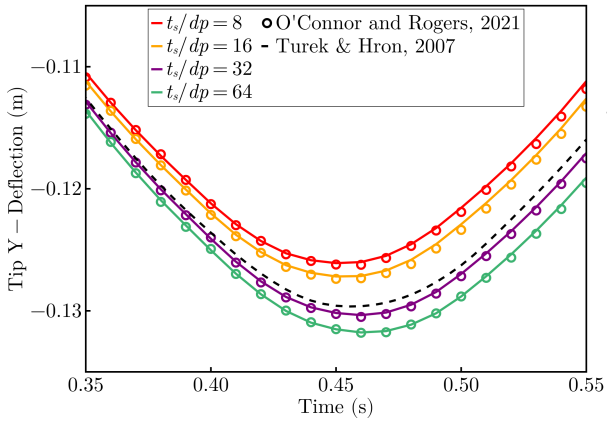

Figure 3: Comparison of TrixiParticles.jl and (O'Connor & Rogers, 2021) against (Turek & Hron, 2007): Tip y-deflection of an oscillating beam with different resolutions, where t_s is the thickness of the beam and dp is the particle spacing.

Figure 4 illustrates an exemplary simulation result, where an elastic sphere, modeled with TLSPH, falls into a tank filled with water, modeled by WCSPH.

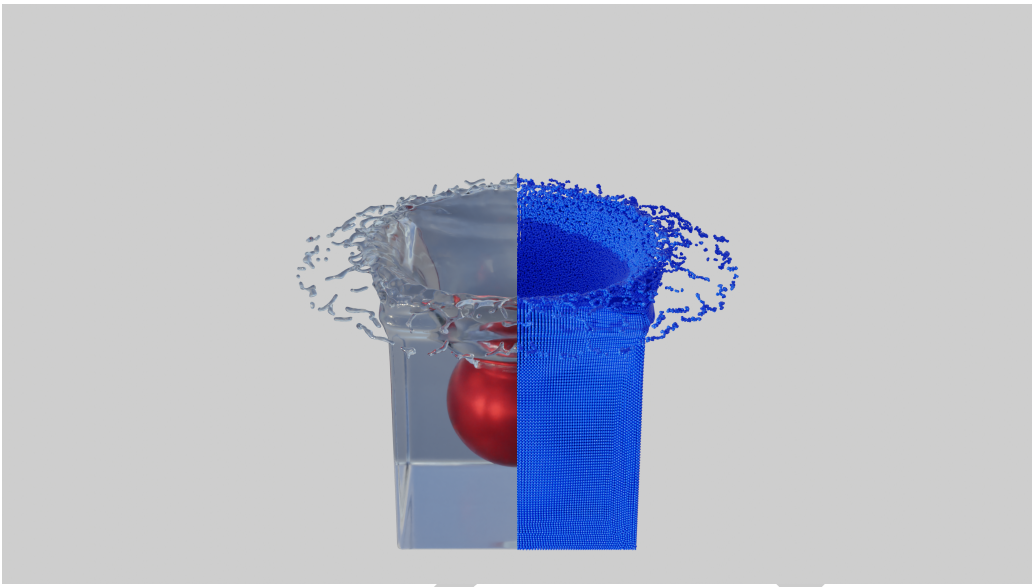

Figure 4: TrixiParticles.jl simulation of an elastic sphere falling into a water tank. Left: Results rendered with blender. Right: Underlying particle representation.

Acknowledgements

Sven Berger acknowledges funding from hereon and HiRSE. Michael Schlottke-Lakemper and Gregor Gassner receive funding through the DFG research unit FOR 5409 "Structure-Preserving Numerical Methods for Bulk- and Interface Coupling of Heterogeneous Models (SNuBIC)" (project number 463312734).

References

Adami, s., Hu, X. Y., & Adams, N. A. (2013). A transport-velocity formulation for smoothed particle hydrodynamics. *Journal of Computational Physics*, *241*. https://doi.org/10.1016/j.jcp.2013.01.043

Akinci, N., Akinci, G., & Teschner, M. (2013). Versatile surface tension and adhesion for SPH fluids. *ACM Trans. Graph.*, *32*(6). https://doi.org/10.1145/2508363.2508395

Antuono, M., Colagrossi, A., Marrone, S., & Molteni, D. (2010). Free-Surface Flows Solved by Means of SPH Schemes with Numerical Diffusive Terms. *Computer Physics Communications*, *181*. https://doi.org/10.1016/j.cpc.2009.11.002

Bender, J., & others. (2024). *SPlisHSPlasH Library*. https://github.com/InteractiveComputerGraphics/SPlisHSPlasH

Bićanić, N. (2004). Discrete element methods. In *Encyclopedia of Computational Mechanics*. Wiley. https://doi.org/10.1002/0470091355.ecm006.pub2

Clausen, J. R. (2013). Entropically damped form of artificial compressibility for explicit simulation of incompressible flow. *Physical Review E*, *87*. https://doi.org/10.1103/physreve.87.013309

Cundall, P. A., & Strack, O. D. L. (1979). A discrete numerical model for granular assemblies. *Géotechnique*, *29*(1), 47–65. https://doi.org/10.1680/geot.1979.29.1.47

Domínguez, J. M., Fourtakas, G., Altomare, C., Canelas, R. B., Tafuni, A., García-Feal, O., Martínez-Estévez, I., Mokos, A., Vacondio, R., Crespo, A. J. C., Rogers, B. D.,

Stansby, P. K., & Gómez-Gesteira, M. (2021). DualSPHysics: From fluid dynamics to multiphysics problems. *Computational Particle Mechanics*, *9*(5), 867–895. https://doi.org/10.1007/s40571-021-00404-2

Gingold, R. A., & Monaghan, J. J. (1977). Smoothed particle hydrodynamics: Theory and application to non-spherical stars. *Monthly Notices of The Royal Astronomical Society*, *181*. https://doi.org/10.1093/mnras/181.3.375

Laha, S., Fourtakas, G., Das, P. K., & Keshmiri, A. (2024). Smoothed particle hydrodynamics based FSI simulation of the native and mechanical heart valves in a patient-specific aortic model. *Scientific Reports*, *14*(1). https://doi.org/10.1038/s41598-024-57177-w

Lastiwka, M., Basa, M., & Quinlan, N. J. (2009). Permeable and non-reflecting boundary conditions in SPH. *International Journal for Numerical Methods in Fluids*, *61*. https://doi.org/10.1002/fld.1971

Monaghan, J. J. (2005). Smoothed particle hydrodynamics. *Reports on Progress in Physics*, *68*. https://doi.org/10.1088/0034-4885/68/8/r01

O'Connor, J., & Rogers, B. D. (2021). A fluid–structure interaction model for free-surface flows and flexible structures using smoothed particle hydrodynamics on a GPU. *Journal of Fluids and Structures*, *104*. https://doi.org/10.1016/j.jfluidstructs.2021.103312

Ramachandran, Prabhu, Bhosale, A., Puri, K., Negi, P., Muta, A., Dinesh, A., Menon, D., Govind, R., Sanka, S., Sebastian, A. S., Sen, A., Kaushik, R., Kumar, A., Kurapati, V., Patil, M., Tavker, D., Pandey, P., Kaushik, C., Dutt, A., & Agarwal, A. (2021). PySPH: A Python-based Framework for Smoothed Particle Hydrodynamics. *ACM Transactions on Mathematical Software*, *47*(4), 1–38. https://doi.org/10.1145/3460773

Ramachandran, P., & Puri, K. (2019). Entropically damped artificial compressibility for SPH. *Computers and Fluids*, *179*. https://doi.org/10.1016/j.compfluid.2018.11.023

Schlottke-Lakemper, M., Gassner, G. J., Ranocha, H., Winters, A. R., & Chan, J. (2021). *Trixi.jl: Adaptive high-order numerical simulations of hyperbolic PDEs in Julia*. https://doi.org/10.5281/zenodo.3996439

Turek, S., & Hron, J. (2007). Proposal for numerical benchmarking of fluid-structure interaction between an elastic object and laminar incompressible flow. In *Fluid-structure interaction*. Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-34596-5_15

Zhang, C., Rezavand, M., Zhu, Y., Yu, Y., Wu, D., Zhang, W., Wang, J., & Hu, X. (2021). SPHinXsys: An open-source multi-physics and multi-resolution library based on smoothed particle hydrodynamics. *Computer Physics Communications*, *267*, 108066. https://doi.org/10.1016/j.cpc.2021.108066