The Journal of Open Source Software

$\mathcal{H}$-HIGNN Toolkit: A Software for Efficient and Scalable Simulation of Large-Scale Particulate Suspensions Using GNNs and $\mathcal{H}$-Matrices

Zisheng Ye [1], **Zhan Ma**[1], **Ebrahim Safdarian**[1], **Shirindokht Yazdani**[1], **and Wenxiao Pan** [1¶]

1 Department of Mechanical Engineering, University of Wisconsin-Madison, Madison, WI, USA ¶ Corresponding author

DOI: 10.21105/joss.08777

Software

- Review
- Repository
- Archive

Editor: Mehmet Hakan Satman

Reviewers:

- @rajeshrinet
- @jonaspleyer
- @marcobonici

Submitted: 20 June 2025
Published: 26 January 2026

Summary

Particulate suspensions, systems of particles dispersed in viscous fluids, play a critical role in various scientific and engineering applications (Maxey, 2017; Shelley, 2016). This software implements $\mathcal{H}$-HIGNN, a framework designed for efficient and scalable simulation of large-scale particulate suspensions (Figure 1). It extends the Hydrodynamic Interaction Graph Neural Network (HIGNN) approach (Ma et al., 2022; Ma & Pan, 2024), which utilizes GNNs to model the mobility tensor that dictates particle dynamics under hydrodynamic interactions (HIs) and external forces. HIGNN effectively captures both short- and long-range HIs and their many-body effects and enables substantial computational acceleration by harvesting the power of machine learning. By incorporating hierarchical matrix ($\mathcal{H}$-matrix) techniques, $\mathcal{H}$-HIGNN further improves computational efficiency, achieving quasi-linear prediction cost with respect to the number of particles. Its GPU-optimized implementation delivers near-theoretical quasi-linear wall-time scaling and near-ideal strong scalability for parallel efficiency. The methodology, validation, and efficiency demonstrations of $\mathcal{H}$-HIGNN are detailed in Ma et al. (2026).

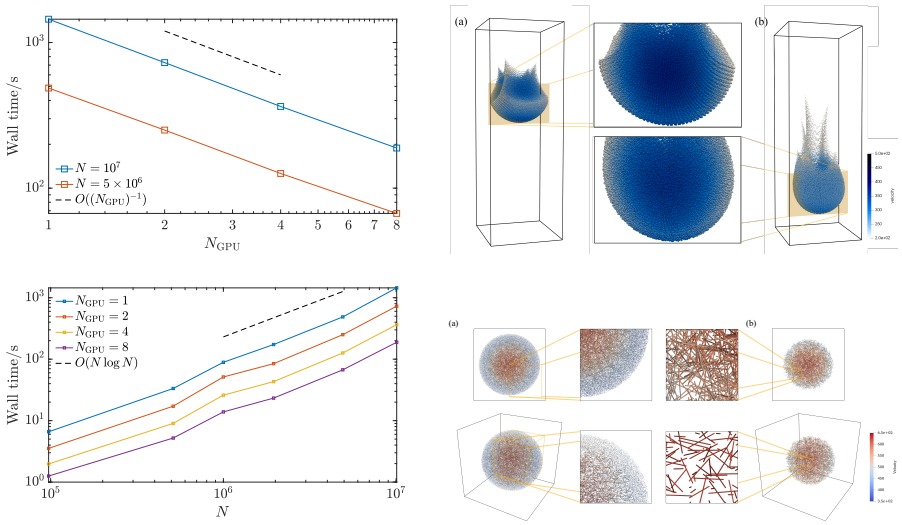

Figure 1: (Left) Scalability test of wall time with respect to N (number of particles) and N_{GPU} (number of GPUs). (Right) Example large-scale particulate suspensions simulated by $\mathcal{H}$-HIGNN Toolkit.

Statement of need

Simulating particulate suspensions in 3D poses substantial computational challenges (Maxey, 2017), limiting prior work to small numbers of particles or requiring extensive computational resources (Singh & Adhikari, 2020; *StokesDT*, 2015; Torre et al., 2025; Townsend, 2024). This emphasizes the need for an efficient, scalable, and flexible toolkit that enables researchers to investigate practically relevant, large-scale suspensions, pushing the boundaries beyond previously accessible scales while minimizing computational resource demands. The present software addresses this gap by introducing the first linearly scalable toolkit capable of simulating suspensions with millions of particles or more using only modest resources, such as a few mid-range GPUs (Ma et al., 2026). Beyond rigid passive particles, the $\mathcal{H}$-HIGNN toolkit is flexible to be extended to simulate suspensions of soft matter systems, such as flexible filaments (Figure 1) or membranes, through the inclusion of additional interparticle interaction forces, and to support the simulation of active matter, such as microswimmers, by incorporating active forces or actuation fields. Therefore, this software offers a powerful platform for exploring hydrodynamic effects across a broad range of systems in soft and active matter.

Description of the software

This software is managed through a single-entry script, **engine.py**, which orchestrates its overall execution. It begins by parsing a JSON config file that contains the problem configuration. Based on the specified input arguments, **engine.py** invokes one of the three modules – **generate.py**, **simulate.py**, or **visualize.py** – corresponding to the software's core functionalities: generating the initial configuration of particles, performing simulations based on the framework of $\mathcal{H}$-HIGNN, and post-processing for visualization, respectively.

generate.py is capable of generating random configurations within a user-defined region, with the current implementation using a spherical domain. It ensures that the particles or filaments are initially spaced by at least a prescribed minimum distance. This initial configuration can be saved to the hard disk for subsequent loading by **simulate.py**.

simulate.py performs simulations based on the framework of $\mathcal{H}$-HIGNN. It loads the initial configuration from the hard disk and invokes the time integrator specified in the JSON config file. The software currently supports two time integration schemes: explicit Euler and 4-th order Runge-Kutta. The script assembles the external forces exerted on each particle in the system. At present, it supports gravitational forces, inter-particle potential forces, e.g., derived from Morse potential, and elastic bonding and bending forces for the particles in a filament. The script loads the pre-trained GNN models for two-body and three-body HIs from the prescribed path, which in turn determine the mobility tensor based on the configuration of particles. The particles' velocities are then calculated from the multiplication of the mobility tensor and the assembled force vector, accelerated by $\mathcal{H}$-matrix. Finally, the particles' positions are advanced using the chosen time integrator. All calculations can be performed in parallel on arbitrary numbers of GPUs. The lower end implementation is based on C++ and wrapped by Pybind11 for easy access to the functionalities.

visualize.py handles the post-processing of all particles' positions updated by **simulate.py**. Filament structures are reconstructed by parsing configuration data from the JSON config file, which specifies the connectivity of particles within each filament chain.

Related software

Stokesian Dynamics in Python (Townsend, 2024) is a Python implementation of the Stokesian Dynamics method for simulating particulate suspensions. It allows for simulating suspensions in both unbounded and periodic domains, with the capability to include particles of several

different sizes. Due to its serial Python implementation, the software is limited to small-scale simulations.

Python-JAX-based Fast Stokesian Dynamics (Torre et al., 2025) is a Python implementation of the fast Stokesian Dynamics method for simulating particulate suspensions. It relies on Google JAX library and leverages its Just-In-Time compilation capabilities. The method's reliance on solving a full linear system at each time step demands pre-computation and storage of the entire mobility matrix within GPU memory. If the matrix size surpasses GPU memory capacity, the high bandwidth advantage cannot be realized, limiting simulations using this software to the order of 10^4 particles subject to the memory limitations of mid-range GPUs.

PyStokes (Singh & Adhikari, 2020) is a Python library for computing phoretic and Stokesian hydrodynamic interactions between particles. It employs a grid-free approach that combines the integral formulations of the Laplace and Stokes equations with spectral expansions and Galerkin discretization. PyStokes has been used to model suspensions of microorganisms, synthetic autophoretic particles, and self-propelled droplets. Its computational cost scales quadratically with the number of particles, and simulations with up to 10^5 particles can be accommodated on multicore computers.

StokesDT (*StokesDT*, 2015) is a C++ toolkit for Stokesian and Brownian dynamics simulations. It employs the particle-mesh Ewald method to evaluate long range hydrodynamic interactions and a block Krylov subspace method to compute Brownian displacements. The implementation supports parallel CPU execution with optional GPU acceleration, enabling simulations with up to 5×10^5 particles (Liu & Chow, 2014).

HYDROLIB (Hinsen, 1995) is a Fortran-based library designed to compute mobility and resistance matrices that characterize hydrodynamic interactions of identical spherical particles in Stokes flow. It accounts for both far-field many-body interactions using multipole expansions and near-field singular behavior through lubrication corrections, and supports both unbounded and periodic boundary conditions. Its big-O computational scaling is not explicitly provided, but the reliance on truncated multipoles and lubrication can lead to steeply increasing costs with particle number and multipole order, making it more suited for theoretical calculations rather than large-scale dynamic simulations.

OpenFOAM (Olsen et al., 2023) also provides a solver for studying large-scale particulate suspensions, but it adopts an Eulerian approach where particles are represented as spatially averaged concentration fields on a computational grid. Therefore, it can capture how particle concentration evolves with the fluid flow, but cannot resolve the dynamics of individual particles. In contrast, our software and other related software discussed above explicitly track the motion of each particle over time. Thus, the two approaches target different levels of description and address complementary aspects of suspension behaviors.

Acknowledgments

We gratefully acknowledge the funding support for this work provided by Army Research Office Grant No. W911NF2310256. During the preparation of this manuscript, the authors used ChatGPT (OpenAI) to assist with grammar correction and language polishing.

References

Hinsen, K. (1995). HYDROLIB: A library for the evaluation of hydrodynamic interactions in colloidal suspensions. *Computer Physics Communications*, *88*(2-3), 327–340. https://doi.org/10.1016/0010-4655(95)00029-f

Liu, X., & Chow, E. (2014). Large-scale hydrodynamic Brownian simulations on multicore and manycore architectures. *2014 IEEE 28th International Parallel and Distributed Processing*

Symposium, 563–572. https://doi.org/10.1109/ipdps.2014.65

Ma, Z., & Pan, W. (2024). Shape deformation, disintegration, and coalescence of suspension drops: Efficient simulation enabled by graph neural networks. *International Journal of Multiphase Flow*, *176*, 104845. https://doi.org/10.1016/j.ijmultiphaseflow.2024.104845

Ma, Z., Ye, Z., & Pan, W. (2022). Fast simulation of particulate suspensions enabled by graph neural network. *Computer Methods in Applied Mechanics and Engineering*, *400*, 115496. https://doi.org/10.1016/j.cma.2022.115496

Ma, Z., Ye, Z., Safdarian, E., & Pan, W. (2026). $\mathcal{H}$-HIGNN: A scalable graph neural network framework with hierarchical matrix acceleration for simulation of large-scale particulate suspensions. *Journal of Computational Physics*, *544*, 114429. https://doi.org/10.1016/j.jcp.2025.114429

Maxey, M. (2017). Simulation methods for particulate flows and concentrated suspensions. *Annual Review of Fluid Mechanics*, *49*(1), 171–193. https://doi.org/10.1146/annurev-fluid-122414-034408

Olsen, N. R., Kadia, S., Pummer, E., & Hillebrand, G. (2023). An OpenFOAM solver for computing suspended particles in water currents. *Journal of Hydroinformatics*, *25*(5), 1949–1959. https://doi.org/10.2166/hydro.2023.309

Shelley, M. J. (2016). The dynamics of microtubule/motor-protein assemblies in biology and physics. *Annual Review of Fluid Mechanics*, *48*(1), 487–506. https://doi.org/10.1146/annurev-fluid-010814-013639

Singh, R., & Adhikari, R. (2020). PyStokes: Phoresis and Stokesian hydrodynamics in Python. *Journal of Open Source Software*, *5*(50), 2318. https://doi.org/10.21105/joss.02318

StokesDT. (2015). https://github.com/xing-liu/stokesdt/wiki

Torre, K. W., Schram, R. D., & Graaf, J. de. (2025). Python-JAX-based fast Stokesian dynamics. *arXiv Preprint arXiv:2503.07847*. https://doi.org/10.21468/scipostphyscodeb.56

Townsend, A. K. (2024). Stokesian dynamics in Python. *Journal of Open Source Software*, *9*(94), 6011. https://doi.org/10.21105/joss.06011