H-HIGNN Toolkit: A Software for Efficient and Scalable Simulation of Large-Scale Particulate Suspensions Using GNNs andH-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.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Mehmet Hakan Satman

Reviewers:

- @rajeshrinet
- @jonaspleyer

Submitted: 20 June 2025
Published: unpublished

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. 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. (2025).

Statement of need

Simulating particulate suspensions in 3D poses substantial computational challenges, limiting prior work to small numbers of particles or requiring extensive computational resources. 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. 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 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 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 determines 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 functionality.

visuliaze.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.

References

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. (2025). *$\mathcal{H}$-HIGNN: A scalable graph neural network framework with hierarchical matrix acceleration for simulation of large-scale particulate* *suspensions*. https://doi.org/10.48550/arXiv.2505.08174

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

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

Torre, K. W., Schram, R. D., & Graaf, J. de. (2025). Python-JAX-based fast stokesian dynamics. *arXiv Preprint arXiv:2503.07847*. https://doi.org/10.48550/arXiv.2503.07847

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