Using Large Language Models to Automate Code Generation for PDE Solvers

Haoyang Wu[a], **Lailai Zhu**[a]

[a] *Department of Mechanical Engineering, National University of Singapore, 117575, Singapore* e1135486@u.nus.edu, lailai_zhu@nus.edu.sg

* Presenting author

1. Introduction

PDEs are ubiquitous in diverse domains including physics, engineering, biology, finance, computer science, the social sciences, and so on. Writing code to numerically solve PDEs is typically a time-consuming, cognitively demanding, and error-prone manual process, necessitating tedious and painstaking debugging.

Recent advances in large language models (LLMs) have created new possibilities for complex applications. One of the promising avenues is harnessing LLMs to generate code to solve partial differential equations (PDEs) [1, 2, 3, 4, 5, 6, 7, 8, 9]. These pioneering studies have focused on widely used software, such as OpenFOAM, MATLAB, FEniCS, COMSOL, and Dedalus, each of which provides comprehensive official documentation.

In contrast, the effectiveness of LLMs in generating code for specialized or poorly documented packages remains unclear. A particularly challenging scenario involves in-house numerical codebases lacking documentation or inline comments. In such cases, critical questions arise: Can LLMs still produce functional code when prompted, and what level of performance can be expected?

To answer these questions, this study explores LLM-based code generation in the framework of a lesser-known numerical library XLB [10], which employs the lattice Boltzmann method (LBM). We develop a strategy using multiple LLM agents to automate code-related developments aimed at solving PDEs, hence termed LLM-PDEveloper. Specifically, we adopt LLM-PDEveloper to address three representative categories of problems: 1) generate code for a new PDE; 2) generate code for new boundary conditions of a given PDE; 3) modify the code of a PDE solver to incorporate new features.

2. Methodology

We briefly outline the workflow of LLM-PDEveloper as follows. First, the user describes the mathematical form of the algorithm in a Markdown file. Second, using the Markdown-formatted description, LLM-PDEveloper generates the source codes for the algorithm in the language of XLB, including code generation, math verification, syntax error detection/correction. Third, LLM-PDEveloper merges the newly generated source codes into the existing codebase.

3. Tests and evaluation of LLM-PDEveloper

Originally, XLB solves the Navier-Stokes (NS) equations—the system of PDEs governing the motion of fluid and serving as the cornerstone of fluid mechanics. Specifically, the version of XLB adopted here solves solely incompressible Newtonian flows.

Here, we employ LLM-PDEveloperfor code generation to extend XLB's functionality to solve other PDEs, focusing on the following tasks: 1) generate a solver for the advection-diffusion (AD) equation; 2) generate code that implements the boundary conditions for the AD solver; 3) generate a solver for the advection-diffusion-reaction (ADR) equation; 4) generate a solver that simulates a specific non-Newtonian (NN) flow.

We have tested LLM-PDEveloper on these tasks using three LLMs, 'o1-preview' ('o1-preview-2024-09-12') and 'o3-mini' ('o3-mini-2025-01-31') from OpenAI, and 'claude-3-5-sonnet-20241022' from Anthropic. For each task, we conduct ten attempts and summarize the success rates in Table 1.

Table 1: Success rate of using LLM-PDEveloper to generate code for four PDE-related tasks

	AD solver	BCs for AD solver	ADR solver	NN solver
o1-preview	8/10	0/10	6/10	10/10
o3-mini	7/10	3/10	7/10	5/10
claude-3-5-sonnet-20241022	7/10	0/10	6/10	10/10

Acknowledgments

We acknowledge the support from the Singapore Ministry of Education Academic Research Fund Tier 2 (MOE-T2EP50221-0012) grant. We thank the useful discussions with Miss Zhuoqun Xu and Prof. Chang Shu. Some of the computation of the work was performed on resources of the National Supercomputing Centre, Singapore (https://www.nscc.sg). The template is based on the IAC 2024 unofficial template.

References

[1] A. Kashefi and T. Mukerji. ChatGPT for programming numerical methods. *Journal of Machine Learning for Modeling and Computing*, 4(2), 2023.

[2] Microsoft Research AI4Science and Microsoft Azure Quantum. The impact of large language models on scientific discovery: a preliminary study using GPT-4. *arXiv preprint arXiv:2311.07361*, 2023.

[3] B. Ni and M. J. Buehler. MechAgents: Large language model multi-agent collaborations can solve mechanics problems, generate new data, and integrate knowledge. *Extreme Mech. Lett.*, 67:102131, 2024.

[4] A. Mohamad and M. Kristen. Physics simulation capabilities of LLMs. *Phys. Scr.*, 99(11):116003, 2024.

[5] D. Kim, T. Kim, Y. Kim, Y. Byun, and T. Yun. A ChatGPT-MATLAB framework for numerical modeling in geotechnical engineering applications. *Comput. Geosci.*, 169:106237, 2024.

[6] Y. Chen, X. Zhu, H. Zhou, and Z. Ren. MetaOpenFOAM: an LLM-based multi-agent framework for CFD. *arXiv preprint arXiv:2407.21320*, 2024.

[7] C. Tian and Y. Zhang. Optimizing collaboration of LLM based agents for finite element analysis. *arXiv preprint arXiv:2408.13406*, 2024.

[8] N. Mudur, H. Cui, S. Venugopalan, P. Raccuglia, M. Brenner, and P. Norgaard. FEABench: Evaluating language models on real world physics reasoning ability. In *NeurIPS 2024 Workshop on Open-World Agents*.

[9] S. Pandey, R. Xu, W. Wang, and X. Chu. OpenFOAMGPT: a RAG-augmented LLM agent for OpenFOAM-based computational fluid dynamics. *arXiv preprint arXiv:2501.06327*, 2025.

[10] M. Ataei and H. Salehipour. XLB: A differentiable massively parallel lattice Boltzmann library in Python. *Comput. Phys. Commun.*, 300:109187, 2024.