FoamGPT: Fine-Tuning Large Language Model for Agentic Automation of CFD Simulations with OpenFOAM

Ling Yue, Peijing Xu, Tingwen Zhang, Nithin Somasekharan, Shaowu Pan[*]

Rensselaer Polytechnic Institute, Troy, NY 12180, USA

Abstract

Computational fluid dynamics (CFD) underpins critical applications across aerospace, energy, and biomedical engineering. OpenFOAM is the most widely used open-source CFD framework; meanwhile, automating its intricate setup remains a persistent challenge. A major barrier to progress has been the lack of a comprehensive, high-quality dataset linking natural language instructions to valid OpenFOAM configurations. To address this gap, we first curate a dataset covering all 202 canonical OpenFOAM tutorial cases, spanning a diverse set of physical and numerical setups. Given this dataset, a fine-tuned large-language model, namely FoamGPT, is then embedded in an agentic framework, where a planner agent first defines the simulation structure, followed by FoamGPT generating the configuration files sequentially. The proposed approach boosts the execution success rate, the fraction of generated cases that can be executed in OpenFOAM without compilation errors, from 0% (Qwen2.5-7B) to 22.73%, and outperforms the prior state-of-the-art fine-tuned model, AutoCFD (20.91%), despite being fine-tuned on a dataset that is two orders-of-magnitude smaller (0.2k v.s. 28.5k samples). Furthermore, we show the general applicability of our dataset by fine-tuning a wide range of models, including Qwen3-8B (0.00% $\rightarrow$ 26.36%), Llama3.1-8B (0.00% $\rightarrow$ 20.91%), and GPT-4.1 nano (0.00% $\rightarrow$ 9.09%). Even for powerful base models like GPT-4.1 mini, our fine-tuning boosts performance from 35.45% to 39.09%. Our results demonstrate that a high-quality, targeted dataset is a critical component for enabling LLMs to robustly automate complex scientific simulation workflows. The code and dataset are publicly available at `https://github.com/csml-rpi/FoamGPT`.

1 Introduction

Computational Fluid Dynamics (CFD) is a cornerstone of modern engineering and science, critical for applications ranging from designing aircraft and optimizing renewable energy systems to diagnosing cardiac disease [Anderson and Wendt, 1995, Ricci, 2025, Ashton and Skaperdas, 2019]. CFD accelerates engineering innovation by reducing the need for costly physical prototypes with numerical simulation. Within this landscape, OpenFOAM [Weller et al., 1998, Greenshields and Weller, 2022] has emerged as the most widely used open-source CFD software, offering a powerful and flexible platform with a diverse user base. However, its full potential is limited by the need to manually configure multiple long files under complex rules — a tedious, error-prone process. The recent advent of Large Language Models (LLMs) [Achiam et al., 2023, Yue et al., 2024, Guo et al., 2025, Jiang et al., 2025] presents a promising opportunity to automate this complex setup, allowing LLM agents

[*]Corresponding author. Email: pans2@rpi.edu

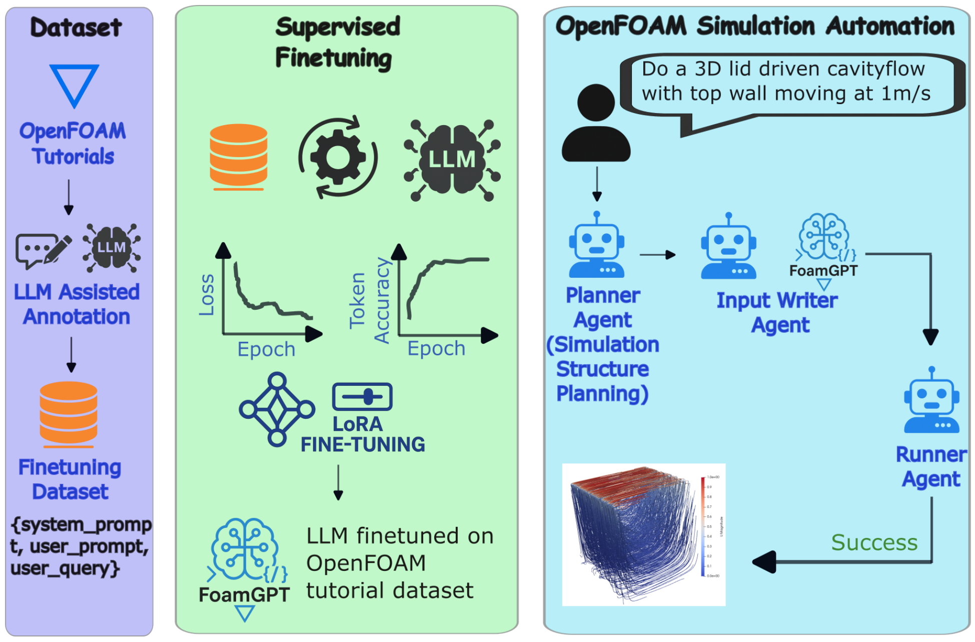
Dataset
OpenFOAM Tutorials
LLM
LLM Assisted Annotation
Finetuning Dataset
{system_prompt, user_prompt, user_query}
Supervised Finetuning
LLM
Loss
Epoch
Token Accuracy
Epoch
LoRA FINE-TUNING
FoamGPT
LLM finetuned on OpenFOAM tutorial dataset
OpenFOAM Simulation Automation
Do a 3D lid driven cavityflow with top wall moving at 1m/s
Planner Agent (Simulation Structure Planning)
FoamGPT
Input Writer Agent
Runner Agent
Success

Figure 1: Illustration of the proposed data curation, fine-tuning, and deployment workflow. Left: We construct a dataset for fine-tuning from OpenFOAM tutorial cases by leveraging a teacher LLM for automatic annotation of user queries. Middle: The specialized FoamGPT model is created by applying supervised fine-tuning, specifically using LoRA, to the curated OpenFOAM dataset. Right: FoamGPT is integrated into an agentic framework, i.e., Foam-Agent [Yue et al., 2025], enabling autonomous generation of OpenFOAM configuration files from user queries.

to translate high-level user requests into ready-to-run OpenFOAM simulation cases [Xu et al., 2024, Zhu et al., 2024, Pandey et al., 2025, Xu et al., 2025, Fan et al., 2025, Chen et al., 2024, 2025a,b, Feng et al., 2025, Zou et al., 2025].

Previous efforts by Dong et al. [2025], AutoCFD, have demonstrated the potential of fine-tuning LLMs for automating OpenFOAM simulation, yet a fundamental limitation remains. The existing dataset, NL2FOAM [Dong et al., 2025], is built upon only 16 basic cases. This lack of diversity inevitably restricts the exposure to the vast spectrum of physics and configurations that OpenFOAM supports, crippling the model's ability to generalize to novel simulation scenarios.

In this paper, we present our work to address this data limitation. To overcome the data bottleneck, we construct a new, comprehensive dataset from 202 distinct OpenFOAM tutorial cases, providing a far richer knowledge base for fine-tuning. We implement this approach through a multi-agent workflow, such as Foam-Agent [Yue et al., 2025]. Within this workflow, we fine-tune FoamGPT on our new dataset to serve as a specialized Input Writer Agent. This agent is responsible for generating the correct configuration file one by one. When a user query is received, a planner agent first decomposes the query into the required OpenFOAM file structure and identifies the list of configuration files to be generated. Then FoamGPT is invoked sequentially to generate each file individually.

Empirical evaluation on CFDLLMBench [Somasekharan et al., 2025] demonstrates the effectiveness of our curated dataset. Using the Qwen2.5-7B model, FoamGPT achieves a execution success rate of **22.73%**, outperforming the previous state-of-the-art AutoCFD [Dong et al., 2025] (20.91%). while using a fine-tuning dataset over 100 times smaller (0.2k vs. 28.5k). The broad applicability and effectiveness of our dataset are further validated across a range of models. For instance, fine-tuning elevates the performance of Qwen3-8B from 0.00% to **26.36%** and Llama3.1-8B from 0.00% to **20.91%**. Even for strong base models like GPT-4.1 mini, our dataset boosts the execution success rate from 35.45% to **39.09%**. These findings strongly suggest that data quality and diversity are more critical than sheer training data volume for the complex task of OpenFOAM automation.

To summarize, our primary contributions are: **1)** We release a diverse natural language dataset for OpenFOAM, curated from 202 distinct and verified OpenFOAM cases. **2)** We provide scripts that employ frontier LLM to process massive raw OpenFOAM cases into a structured dataset in a LLM-friendly format, enabling AI4Science researchers to easily curate proprietary data for fine-tuning. **3)** We demonstrate through comprehensive evaluation on the CFDLLMBench [Somasekharan et al., 2025] benchmark that our approach significantly outperforms both its non-fine-tuned base model and the prior state-of-the-art fine-tuned model [Dong et al., 2025].

2 Methodology

Our proposed framework is illustrated in Figure 1. First, the planner Agent interprets the user query to establish a plan, which involves determining the necessary directory structure and compiling a list of all required OpenFOAM files to be generated. Second, FoamGPT generates each OpenFOAM configuration file sequentially identified in the planning stage. It receives the overall user query as context, but is tasked with generating only one specific file at a time.

Data Curation and Preprocessing for Fine-Tuning We design a three-stage pipeline to convert the raw OpenFOAM tutorial into a structured, instruction-response dataset. (1) The first stage involves systematic parsing and quality filtering [Chen et al., 2024]. The pipeline begins with all 202 official OpenFOAM tutorials, followed by a filtering step that isolates each simulation case and its folder structures. In addition, high-level metadata (`case_name`, `domain`, `solver`) and the raw content of every configuration file for the simulation case are extracted. (2) The second stage addresses the challenge of synthesizing realistic and complete user queries based on the tutorial files. We use Claude Sonnet 4 [Anthropic, 2025] as a powerful "teacher" to synthesize user queries for each of the 202 cases. All 202 generated user queries are manually validated by a human expert, with the final set stored in the JSON file. The details can be found in Appendix A.2. (3) In the final stage, the script merges the structured file data with the synthetic user queries. For each case, a precise instruction is engineered: this instruction combines a `user_prompt` (providing the high-level query from Stage 2) with a dynamic `system_prompt`. This `system_prompt` serves a dual purpose: it primes the model to act as an OpenFOAM expert, and it is programmatically populated with the specific, task-critical metadata (e.g., `file_name`, `folder_name`, and `solver`) extracted in Stage 1. This design strategy simplifies the generation task, focusing the model on file content generation by providing explicit constraints. It also mimics the operational flow at inference time, where such metadata would be supplied by an upstream planner agent. The original, validated `file_content` serves as the ground-truth response. This produces the final dataset, which is then split into training and test sets.

Model Fine-Tuning We fine-tune the gpt-4.1-mini and gpt-4.1-nano [Achiam et al., 2023] using the official OpenAI fine-tuning API [OpenAI, 2025]. This model is selected as it offers a strong balance between performance and cost, making it an ideal candidate for validating the effectiveness of our fine-tuning dataset and method. The fine-tuning process minimizes the standard auto-regressive cross-entropy loss. Crucially, this loss is computed only on the target response tokens (the file content), while the `system_prompt` and `user_prompt` are treated as conditioning context. Let $C = (c_1, \ldots, c_M)$ represent the token sequence of the combined prompts, and $R = (r_1, \ldots, r_L)$ be the token sequence for the ground-truth OpenFOAM file (the response). The objective is to maximize the conditional likelihood of R given C, which is modeled as the negative log-likelihood loss $\mathcal{L}$ over the model parameters θ:

$$\mathcal{L}(\theta) = -\sum_{i=1}^{L-1} \log P(r_{i+1}|C, r_1, \ldots, r_i; \theta)$$

where $P(r_i|C, r_{<i}; \theta)$ is the probability assigned by the model to the response token r_i, given the full context C and the preceding response tokens $r_{<i}$. This objective trains the model to accurately predict the next token for the file as a direct response to the given instruction, effectively teaching it the syntax and structure of the domain in a conditional manner. The fine-tuning process is configured with the following hyperparameters: we train for 3 epochs with a batch size of 8 and a fixed seed of 2025 to ensure reproducibility. For additional details based on fine-tuning open-source models, please see Appendix A.1.

3 Experiments and Results

3.1 Experimental Setup

To evaluate FoamGPT, we use Foam-Agent [Yue et al., 2025] as our backbone multi-agent framework. This backbone uses Claude-Sonnet-4.0 [Anthropic, 2025] as its planner agent, which handles all planning tasks before writing the actual OpenFOAM files. The backbone also incorporates a retrieval-augmented generation (RAG) [Lewis et al., 2020] by including similar cases to the context. The workflow is then tested on the CFDLLMBench [Somasekharan et al., 2025]. This benchmark consists of 110 OpenFOAM cases across 11 distinct physics scenarios, covering a wide range of physical phenomena and geometric complexity. Each benchmark case is described using natural language prompts that include the problem description, physical scenario, geometry, solver requirements, boundary conditions, and simulation parameters. The execution success rate is measured by the percentage of cases that executed successfully, given the prompt describing the simulation scenarios. To ensure a fair evaluation and isolate our model's generative capabilities, we disabled the iterative review-and-fix mechanism within Foam-Agent (thereby removing automated error correction) while keeping only the planner and input writer agents.

Table 1: Performance comparison of models on the CFDLLMBench

Model	Framework	Execution Success Rate (%) ↑
Qwen2.5-7B	Base	0.00
	FoamGPT	**22.73**
	AutoCFD [Dong et al., 2025]	20.91
Qwen3-8B	Base	0.00
	FoamGPT	**26.36**
Llama3.1-8B	Base	0.00
	FoamGPT	**20.91**
Claude-3.5 Haiku	–	25.45
GPT-4.1 nano	Base	0.00
	FoamGPT	**9.09**
GPT-4.1 mini	Base	35.45
	FoamGPT	**39.09**

3.2 Experimental Results

Table 1 shows that compared to the existing AutoCFD method on Qwen2.5-7B [Bai et al., 2023], FoamGPT achieves a higher execution success rate (22.73% vs. 20.91%). This superior performance is particularly notable as it was achieved using a fine-tuning dataset of only 0.2k samples, which is less than 1% of the 28.5k samples required by AutoCFD.

Furthermore, the effectiveness of FoamGPT is consistent across a diverse range of base models. As shown in the table, our framework substantially boosts performance for open-source models like Qwen3-8B (0.00% to 26.36%) and Llama3.1-8B (0.00% to 20.91%). We also validated FoamGPT using the official OpenAI fine-tuning API, improving GPT-4.1 nano from 0.00% to 9.09% and elevating the strong GPT-4.1 mini baseline from 35.45% to 39.09%. For a baseline comparison, we evaluated the similarly-sized Claude-3.5 Haiku, which scored 25.45%. A direct fine-tuning comparison with FoamGPT on the Claude series was not possible, as Claude does not currently offer a public fine-tuning API.

While FoamGPT generates one file at a time, AutoCFD generates all required files simultaneously. During our experiments, we observed that AutoCFD's generated output suffered from significant formatting inconsistencies. The default parser provided with AutoCFD was unable to process this raw output, resulting in an initial execution success rate of 0%. To explore the potential of AutoCFD and establish a fair comparison, we implemented a custom parser powered by Claude-Sonnet to handle these formatting issues. With this correction, the execution success rate for AutoCFD improved to 20.91%.

For both FoamGPT and AutoCFD, the common reasons for failures are missing files, missing keywords, and incorrect solver selection. A problem exclusive to AutoCFD is its occasional generation of plain text. On 13/110 cases, AutoCFD outputs a general description of OpenFOAM files rather than their content. FoamGPT consistently generated valid OpenFOAM files on all benchmark cases. This crucial difference, along with FoamGPT's higher execution success rate, suggests that FoamGPT is better than AutoCFD in following the user prompt and system prompt to generate reliable OpenFOAM files. Regarding physical accurracy of the simulations, Figure 2 shows the comparison of the temperature contour from the Benard Cell simulation produced by AutoCFD and FoamGPT, compared to the ground truth simulation. FoamGPT predictions closely align with ground truth, while AutoCFD predictions seem to deviate, due to incorrect type assignment of "fixedFluxPressure" boundary condition at the walls of the domain instead of type "calculated".

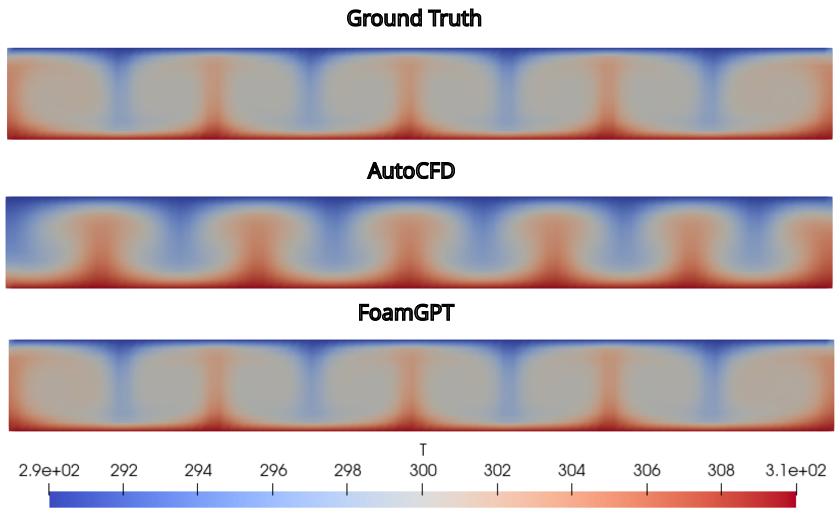

Figure 2: Temperature contour at the final timestep (t=1000s) for Benard Cell simulation, comparing AutoCFD, FoamGPT, and the ground truth. FoamGPT predictions align closely with the ground truth, while AutoCFD predictions produce a very different contour for temperature, emanating from the incorrect assignment of boundary conditions.

4 Conclusion

We introduced FoamGPT, covering automated data annotation, supervised fine-tuning, and agentic configuration of a computational simulation with OpenFOAM. Our experiments demonstrated that our diverse dataset yields significant performance improvements across a diverse range of both open-source and proprietary models on the CFDLLMBench. Future work will focus on (1) scaling FoamGPT to larger open source models, (2) enhancing the diversity of annotated data via data augmentation, and (3) fine-tuning other agents of FoamGPT. We have released the open-source code and dataset to welcome the entire CFD community to build upon our work.

Acknowledgements

This work received funding support by the U.S. Department of Energy with ID DE-SC0025425. Shaowu Pan is supported by the Google Research Scholar Program. Computing resources are supported by the Lambda research grant program, NSF-ACCESS-PHY240112 and by the National Energy Research Scientific Computing Center under award NERSC DDR-ERCAP0030714.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

John David Anderson and John Wendt. *Computational fluid dynamics*, volume 206. Springer, 1995.

Anthropic. Claude sonnet 4. `https://www.anthropic.com/`, 2025. Large Language Model.

Neil Ashton and Vangelis Skaperdas. Verification and validation of openfoam for high-lift aircraft flows. *Journal of Aircraft*, 56(4):1641–1657, 2019.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

Yuxuan Chen, Xu Zhu, Hua Zhou, and Zhuyin Ren. Metaopenfoam: an llm-based multi-agent framework for cfd. *arXiv preprint arXiv:2407.21320*, 2024.

Yuxuan Chen, Long Zhang, Xu Zhu, Hua Zhou, and Zhuyin Ren. Optmetaopenfoam: Large language model driven chain of thought for sensitivity analysis and parameter optimization based on cfd. *arXiv preprint arXiv:2503.01273*, 2025a.

Yuxuan Chen, Xu Zhu, Hua Zhou, and Zhuyin Ren. Metaopenfoam 2.0: Large language model driven chain of thought for automating cfd simulation and post-processing. *arXiv preprint arXiv:2502.00498*, 2025b.

Guanting Dong, Hongyi Yuan, Keming Lu, Chengpeng Li, Mingfeng Xue, Dayiheng Liu, Wei Wang, Zheng Yuan, Chang Zhou, and Jingren Zhou. How abilities in large language models are affected by supervised fine-tuning data composition. *arXiv preprint arXiv:2310.05492*, 2023.

Zhehao Dong, Zhen Lu, and Yue Yang. Fine-tuning a large language model for automating computational fluid dynamics simulations. *Theoretical and Applied Mechanics Letters*, page 100594, 2025.

E Fan, Weizong Wang, and Tianhan Zhang. Chatcfd: an end-to-end cfd agent with domain-specific structured thinking. *arXiv preprint arXiv:2506.02019*, 2025.

Jingsen Feng, Yupeng Qi, Ran Xu, Sandeep Pandey, and Xu Chu. turbulence. ai: an end-to-end ai scientist for fluid mechanics. *Theoretical and Applied Mechanics Letters*, page 100620, 2025.

Christopher J Greenshields and Henry G Weller. Notes on computational fluid dynamics: General principles. *(No Title)*, 2022.

Jiachen Guo, Chanwook Park, Dong Qian, Thomas JR Hughes, and Wing Kam Liu. Large language model-empowered next-generation computer-aided engineering. *arXiv preprint arXiv:2509.11447*, 2025.

Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, and Sai Qian Zhang. Parameter-efficient fine-tuning for large models: A comprehensive survey. *arXiv preprint arXiv:2403.14608*, 2024.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Shuo Jiang, Min Xie, Frank Youhua Chen, Jian Ma, and Jianxi Luo. Intelligent design 4.0: Paradigm evolution toward the agentic ai era, 2025. URL `https://arxiv.org/abs/2506.09755`.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33: 9459–9474, 2020.

Kai Lv, Yuqing Yang, Tengxiao Liu, Qinghui Gao, Qipeng Guo, and Xipeng Qiu. Full parameter fine-tuning for large language models with limited resources. *arXiv preprint arXiv:2306.09782*, 2023.

OpenAI. Model optimization. `https://platform.openai.com/docs/guides/model-optimization`, 2025. Accessed: 2025-08-30.

Sandeep Pandey, Ran Xu, Wenkang Wang, and Xu Chu. Openfoamgpt: A retrieval-augmented large language model (llm) agent for openfoam-based computational fluid dynamics. *Physics of Fluids*, 37(3), 2025.

Venkatesh Balavadhani Parthasarathy, Ahtsham Zafar, Aafaq Khan, and Arsalan Shahid. The ultimate guide to fine-tuning llms from basics to breakthroughs: An exhaustive review of technologies, research, best practices, applied research challenges and opportunities. *arXiv preprint arXiv:2408.13296*, 2024.

Alessio Ricci. Review of openfoam applications in the computational wind engineering: from wind environment to wind structural engineering. *Meccanica*, 60(6):1695–1735, 2025.

Arjun Singh, Nikhil Pandey, Anup Shirgaonkar, Pavan Manoj, and Vijay Aski. A study of optimizations for fine-tuning large language models. *arXiv preprint arXiv:2406.02290*, 2024.

Nithin Somasekharan, Ling Yue, Yadi Cao, Weichao Li, Patrick Emami, Pochinapeddi Sai Bhargav, Anurag Acharya, Xingyu Xie, and Shaowu Pan. Cfd-llmbench: A benchmark suite for evaluating large language models in computational fluid dynamics. *arXiv preprint arXiv:2509.20374*, 2025.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. Trl: Transformer reinforcement learning. `https://github.com/huggingface/trl`, 2020.

Henry G Weller, Gavin Tabor, Hrvoje Jasak, and Christer Fureby. A tensorial approach to computational continuum mechanics using object-oriented techniques. *Computers in physics*, 12(6): 620–631, 1998.

Leidong Xu, Danyal Mohaddes, and Yi Wang. Llm agent for fire dynamics simulations. *arXiv preprint arXiv:2412.17146*, 2024.

Lingling Xu, Haoran Xie, Si-Zhao Joe Qin, Xiaohui Tao, and Fu Lee Wang. Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment. *arXiv preprint arXiv:2312.12148*, 2023.

Zhaoyue Xu, Long Wang, Chunyu Wang, Yixin Chen, Qingyong Luo, Hua-Dong Yao, Shizhao Wang, and Guowei He. Cfdagent: A language-guided, zero-shot multi-agent system for complex flow simulation. *arXiv preprint arXiv:2507.23693*, 2025.

Ling Yue, Sixue Xing, Jintai Chen, and Tianfan Fu. Clinicalagent: Clinical trial multi-agent system with large language model-based reasoning. In *Proceedings of the 15th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics*, pages 1–10, 2024.

Ling Yue, Nithin Somasekharan, Tingwen Zhang, Yadi Cao, and Shaowu Pan. Foam-agent: An end-to-end composable multi-agent framework for automating cfd simulation in openfoam. *arXiv preprint arXiv:2509.18178*, 2025.

Max Zhu, Adrián Bazaga, and Pietro Liò. Fluid-llm: Learning computational fluid dynamics with spatiotemporal-aware large language models. *arXiv preprint arXiv:2406.04501*, 2024.

Weihao Zou, Weibing Feng, and Pin Wu. Flowbert: Prompt-tuned bert for variable flow field prediction. *arXiv preprint arXiv:2506.08021*, 2025.

A Appendix

A.1 Finetuning based on open source models

Supervised Fine-Tuning with LoRA To specialize our base models, we use Supervised Fine-Tuning (SFT) [Dong et al., 2023], which is highly effective for teaching models structured, domain-specific formats like OpenFOAM's dictionary files. For the practical implementation, we utilize the SFTTrainer class from the Hugging Face TRL (Transformer Reinforcement Learning) library [von Werra et al., 2020]. To make this process efficient, we employ Low-Rank Adaptation (LoRA) [Hu et al., 2021], a Parameter-Efficient Fine-Tuning (PEFT) technique. LoRA freezes the pre-trained model weights and injects trainable, low-rank matrices into the layers of the Transformer architecture. This allows us to adapt the model to the new domain by training only a small fraction of the total parameters, significantly reducing computational cost and memory queries [Lv et al., 2023, Xu et al., 2023, Singh et al., 2024, Parthasarathy et al., 2024, Han et al., 2024].

The key hyperparameters for our LoRA configuration are selected to ensure stable and effective training. We use a LoRA rank (r) of 32 and a scaling factor (alpha) of 16. The LoRA updates are applied to all linear layers (target_modules="all-linear") of the models. We train for 6 epochs using a learning rate of 3e-4, which is managed by a cosine scheduler to ensure smooth convergence.

A.2 Synthesizing User Queries via a Teacher LLM

A primary challenge in creating an instruction-following dataset from existing source code is the absence of natural language instructions. The OpenFOAM tutorial cases [Weller et al., 1998] provide the "answer" (the complete set of simulation files) but not the "question" (the user's original request). To bridge this gap, we implement a data synthesis pipeline to reverse-engineer a realistic user query for each of the 202 simulation cases. This process relies on a powerful "teacher" LLM (Claude 4.0 Sonnet) [Anthropic, 2025] tasked with summarizing the technical details of a given case into a concise, high-level user query.

The synthesis for each case is guided by a two-part prompt structure: a detailed **System Prompt** that defines the persona and output queries for the teacher LLM, and a context-rich **User Prompt** that provides all the raw data from the specific OpenFOAM case.

System Prompt: Defining the Expert The system prompt instructs the teacher LLM to act as an expert OpenFOAM simulation engineer. It provides a strict template and a comprehensive checklist of technical details to extract and include in the final ‘user_query‘. This ensures consistency and technical fidelity across all 202 synthesized queries.

System Prompt for User Query Synthesis

You are an expert OpenFOAM simulation engineer. Your task is to analyze OpenFOAM case files and generate a realistic user query that a simulation engineer would specify when requesting such a simulation.
Based on the provided OpenFOAM case files, generate a user_query that follows these patterns:
STRUCTURE QUERIES:

1. Start with "do a [simulation type]" or "Perform a [simulation type]" or "Conduct a [simulation type]"
2. Include solver specification: "using [solver name] solver" or "Use [solver name] solver"
3. Specify geometric details with precise dimensions.
4. When reporting dimensions, report values as is in the geometry file without scaling using convertToMeters parameter. Further report the convertToMeters value seperatly.
5. Define all boundary conditions for different patches/surfaces
6. Include time parameters (start time, end time, timestep, output frequency)
7. Specify physical properties (viscosity, density, temperature, pressure, etc.)
8. Mention grid/mesh details when relevant
9. Include algorithm details (PIMPLE, SIMPLE, etc.) when applicable
10. When reporting intial location of fluid, report their location in x,y,z coordinates. For example water occupies the region 0<=x<=1, 0<=y<=1, 0<=z<=1.
11. Detail the geometry of the domain as much as possible in a concise manner.

TECHNICAL ACCURACY:

- Use correct OpenFOAM terminology and solver names
- Include realistic engineering values with proper units
- Specify boundary condition types accurately (fixedValue, zeroGradient, noSlip, etc.)
- Include material properties relevant to the simulation type
- Mention turbulence models when applicable (k-epsilon, RAS, etc.)

FORMAT QUERIES:

- Generate a single, comprehensive sentence or paragraph
- Use technical language appropriate for CFD engineers
- Include specific numerical values extracted from the case files
- Maintain consistency with OpenFOAM naming conventions

Generate ONLY the user_query text as a single comprehensive statement, with no additional explanation, formatting, or metadata.

User Prompt: Providing Case-Specific Context For each case, the Python script programmatically constructs a detailed user prompt. This prompt collates all relevant metadata and the complete contents of every file associated with the case, presenting it in a structured and readable format. This provides the teacher LLM with all the necessary context to generate an accurate and comprehensive summary.

User Prompt Template for a Single Case

```
Analyze this OpenFOAM case and generate a realistic user query:

CASE METADATA:
- Case Name: <case_name>
- Domain: <case_domain>
- Category: <case_category>
- Solver: <case_solver>

CASE FILES ANALYSIS:

=== <folder_name_1>/ ===

--- <file_name_1> ---
<full_content_of_file_1>

--- <file_name_2> ---
<full_content_of_file_2>

=== <folder_name_2>/ ===

--- <file_name_3> ---
<full_content_of_file_3>

... (and so on for all files in the case)

TASK:
Based on the case files above, extract the key simulation
parameters and generate a realistic user_query that an
engineer would specify when requesting this simulation.
Focus on:

1. Simulation type and solver
2. Domain geometry and dimensions
3. Boundary conditions for all patches
4. Time settings (timestep, end time, output frequency)
5. Physical properties (viscosity, density, temperature, etc.)
6. Grid/mesh specifications
7. Algorithm settings (PIMPLE, SIMPLE, turbulence models, etc.)

Generate a single, comprehensive user_query statement that
captures all essential simulation parameters.
```

This automated process is executed for all 202 cases. Subsequently, each of the generated ‘user_query‘ strings is manually reviewed and validated by a domain expert to ensure its quality and correctness before being included in the final fine-tuning dataset.