CFDLLMBench: A Benchmark Suite for Evaluating Large Language Models in Computational Fluid Dynamics

Anonymous Author(s)
Affiliation
Address
email

Abstract

Large Language Models (LLMs) have demonstrated strong performance across general NLP tasks, but their utility in automating numerical experiments of complex physical system—a critical and labor-intensive component—remains underexplored. As the major workhorse of computational science over the past decades, Computational Fluid Dynamics (CFD) offers a uniquely challenging testbed for evaluating the scientific capabilities of LLMs. We introduce *CFDLLMBench*, a benchmark suite comprising three complementary components—*CFDQuery*, *CFDCodeBench*, and *FoamBench*—designed to holistically evaluate LLM performance across three key competencies: graduate-level CFD knowledge, numerical and physical reasoning of CFD, and context-dependent implementation of CFD workflows. Grounded in real-world CFD practices, our benchmark combines a detailed task taxonomy with a rigorous evaluation framework to deliver reproducible results and quantify LLM performance across code executability, solution accuracy, and numerical convergence behavior. *CFDLLMBench* establishes a solid foundation for the development and evaluation of LLM-driven automation of numerical experiments for complex physical systems.

1 Introduction

Recent advances in large language models (LLMs) have shown remarkable performance across general natural language processing tasks [19, 1]. However, their potential as scientific assistants—specifically, their ability to automate numerical simulation workflows—remains largely underexplored [10, 25]. Computational Fluid Dynamics (CFD) is critical in domains such as urban physics [7, 6], aerospace [46], climate [42], and aerial [43] and underwater robotics [28], and has labor-intensive workflows for computationally expensive numerical simulations of fluid dynamics. CFD workflows involve multiple steps, such as mesh generation, setup of boundary and initial conditions, and solver configuration. Such scientific workflows require an understanding of highly specialized knowledge [51], numerical and physical reasoning [50], and have context-dependent implementations involving domain-specific tool calling [20].

In this paper, we introduce *CFDLLMBench* (Figure 1), the first LLM benchmark for CFD composed of curated datasets designed to holistically evaluate LLMs' performance across three key competencies:

Graduate-level CFD knowledge: Understanding of fluid mechanics and concepts of numerical analysis relevant to CFD.

Numerical and physical reasoning: Applying advanced math and physics knowledge to solve difficult problems. For example, selecting a suitable numerical method that solves the governing equation, with the appropriate boundary conditions and initial conditions.

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

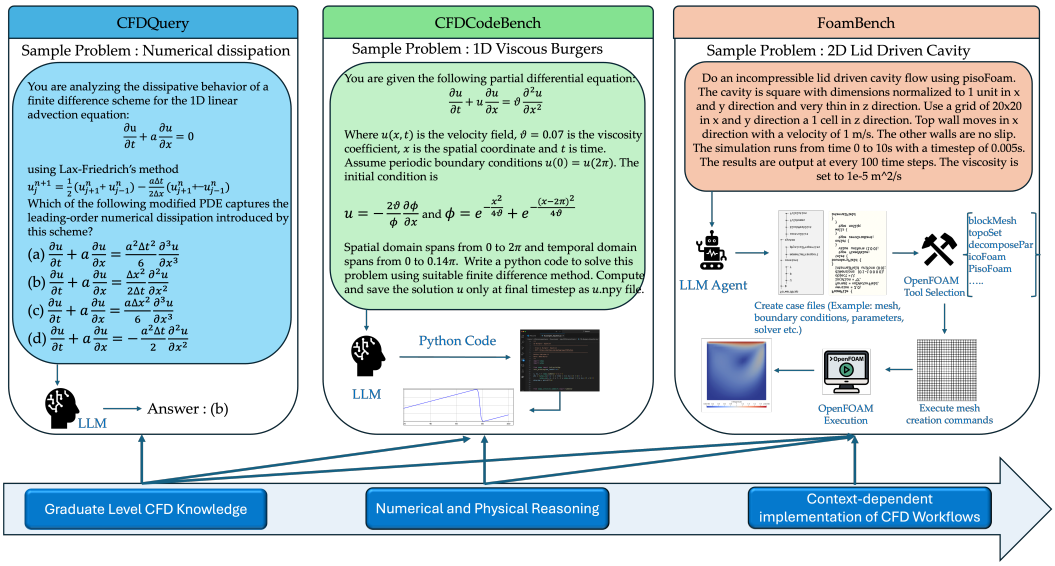

Figure 1: Overview of CFDLLMBench: As the first ever LLM benchmark designed to holistically evaluate LLM's capabilities for CFD, it consists of three different tasks and datasets. (1) *CFDQuery*: Graduate-level CFD QA. (2) *CFDCodeBench*: Coding questions about solving common linear/nonlinear PDEs encountered in CFD. (3) *FoamBench*: Configuring OpenFOAM case files for simulating realistic engineering scenarios such as incompressible flow over obstacles, supersonic flow with shockwaves, Rayleigh-Benard convection, etc.

Context-dependent implementation of CFD workflows: Selecting and configuring CFD preprocessing and numerical solver settings according to physical context.

The *CFDLLMBench* benchmark suite evaluates these competencies using three benchmark tasks: **1) CFDQuery:** 90 multiple-choice questions curated from graduate-level CFD lecture notes that assess LLM's ability in the conceptual understanding of CFD knowledge. **2) CFDCodeBench:** 24 CFD programming tasks designed to assess an LLM's ability to generate correct simulation code from descriptions of physical problems. **3) FoamBench:** 110 basic and 16 advanced numerical simulation tasks, drawn from practical engineering problems, designed to assess the LLM's ability to implement OpenFOAM [53] workflows. OpenFOAM projects typically have 6-7 configuration files, totaling ∼300-600 lines of code per case.

Although strong performance on *CFDQuery* indicates excellent recall of relevant CFD knowledge, success in solving *CFDCodeBench* and *FoamBench* would suggest that LLM possesses reasoning and workflow implementation capabilities near the proficiency of a competent CFD assistant. To support a holistic evaluation of these diverse benchmark tasks, we equip each benchmark task with one or more tailored metrics, which are developed in collaboration with CFD experts.

We use *CFDLLMBench* to evaluate both state-of-the-art proprietary and open-source LLMs. Despite relatively strong performance on *CFDQuery*, the results highlight the challenge of the latter two tasks (see Figure 2): the best performing model achieves only **14%** on *CFDCodeBench* and **34%** on *FoamBench*. In the more complex *FoamBench Advanced* split, generally, performance is poor, e.g., Gemini 2.5 Flash drops to **0%**. In *FoamBench*, all models show major improvement when deployed in a multi-agent framework, as opposed to zero-shot prompting (near 0 performance).

The remainder of the paper is organized as follows. Section 2 describes related work. Section 3 presents our holistic CFD benchmark. Section 4 summarizes our experimental setup and results, which are discussed in Section 5. Section 6 has limitations and Section 7 concludes the paper.

2 Related Work

LLMs for science & engineering LLMs are becoming increasingly proficient at knowledge-intensive tasks in general science [48, 5, 33, 45] and engineering [21], aided by dedicated pretraining on scientific corpora. The development of language agents with tool-use [40, 9, 10, 36] further

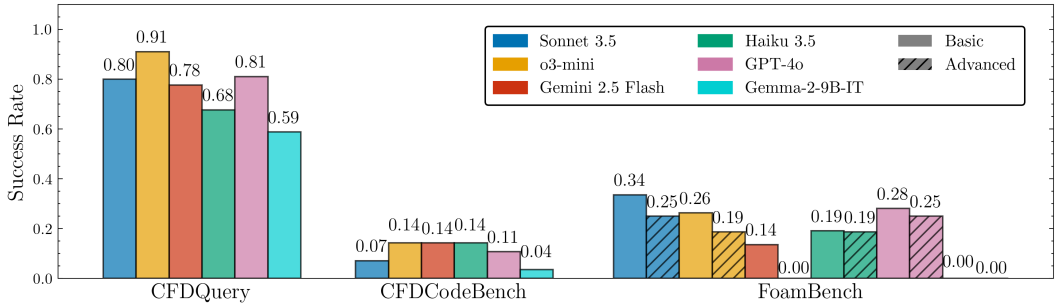

Figure 2: Success Rate comparison of different models across the three tasks. Success Rate is the fraction of cases in the benchmark that produce physically accurate results (higher is better). The detailed definition of Success Rate for each benchmark task can be found in section 3.3. The results for *FoamBench* are produced using the *Foam-Agent* framework with RAG, Reviewer, and Sonnet 3.5. There is a steep drop in performance from graduate-level knowledge (*CFDQuery*) to practical simulation workflow automation *FoamBench*.

enhances LLMs' capabilities, enabling them to integrate with complex scientific and engineering software [15]. Recent work explores the use of LLMs to generate input files in domain-specific languages for quantum chemistry [20] and building energy [23] simulators, tasks which demand substantial time from a researcher to master. LLMs are also accelerating workflow automation in computational physics. MyCrunchGPT [25] demonstrates the use of automated scientific machine learning workflows to optimize airfoils in aerodynamics. MetaOpenFOAM [13], OpenFOAMGPT [39], and Foam-Agent [54] exemplify this trend by automatically configuring and conducting complex CFD simulations based on human requests. These examples highlight the critical need for effective workflow automation benchmarking.

LLM benchmarks for science & engineering Recent interest in the use of LLMs in science and engineering has led to benchmarks measuring specific advanced LLM capabilities such as graduate-level scientific problem solving [41, 52, 18, 55] and long-context reasoning [29, 16]. Our benchmark aims at practicality, providing a holistic evaluation that includes a real-world numerical simulation workflow automation task. Other related workflow benchmarks focus on paper reproduction [47, 8, 44] or data analysis workflows [14, 34, 35]. Paper reproduction, data analysis, and simulation automation (ours) are all critical workflows in the scientific discovery life cycle. Differently, our benchmark uniquely evaluates numerical and physical reasoning, an underexplored capability in LLMs. Thus, these benchmarks assess distinct yet complementary capabilities for scientific workflow automation. The most closely related benchmark is FEABench [31], which evaluates the ability of LLMs as agents for solving PDEs using COMSOL, a commercial finite element analysis software that requires a license of several thousand dollars per year. In contrast, our work is a comprehensive benchmark that consists of domain-specific knowledge, reasoning, and OpenFOAM [22]workflow automation, one of the most widely used open-source numerical simulation software.

LLM benchmarks for code generation Code generation benchmarks such as MBPP [3], HumanEval [12], DS-1000 [27], and SWE-Bench [24] evaluate general coding yet lack the complexity of scientific and engineering tasks. These require understanding advanced concepts and implementing sophisticated algorithms that involve specialized libraries. SciCode [50] is a related scientific coding benchmark, but their CFD examples-1D heat transfer and 1D Burgers equation-are far from enough to represent the algorithmic, physical, and geometrical complexity in CFD. There is a clear need for a comprehensive code generation benchmark that meets the scientific standards for CFD.

3 CFDLLMBench: a benchmark suite for evaluating LLMs in CFD

We present *CFDLLMBench*, which holistically assesses three capabilities of LLMs necessary to perform CFD-related tasks (Figure 1). We begin with *CFDQuery* which evaluates graduate-level conceptual understanding, after which the benchmark progresses to the application of this knowledge through *CFDCodeBench*, where LLMs must use numerical and physical reasoning over a description of a physical problem to correctly generate CFD code in Python. Finally, the most practical and challenging benchmark task is *FoamBench*, where LLMs write input files for a CFD software suite

that must correctly pre-process, configure, and execute simulations given physical context expressed in natural language.

OpenFOAM OpenFOAM [53] is an open-source, license-free CFD software suite (a collection of software for fluid-flow simulation that covers meshing, solving, and post-processing) widely used in academia and industry. OpenFOAM projects have a precise file organization and various configuration and source files arranged in a strict folder hierarchy. OpenFOAM's accessibility, extensibility, and rich community resources make it an attractive platform for an LLM benchmark. However, automating OpenFOAM workflows poses significant challenges for language models and agents. Writing code for OpenFOAM requires long-context understanding to track simulation parameters across multiple files, domain-specific tool usage, and accurate implementation of complex physical models. The third benchmark task in our suite, *FoamBench*, uses OpenFOAM as the underlying CFD software suite.

3.1 Datasets Overview

CFDQuery This dataset consists of 108 multiple-choice questions pertaining to CFD curated by three domain experts. These questions probe core concepts in fluid mechanics, linear algebra, and numerical methods, with source materials adapted from both web-scraped content and CFD lecture notes. The solution to these problems require the LLMs to have deep knowledge about topics in CFD like linear algebra, numerical methods and fluid dynamics.

CFDCodeBench This dataset consists of 24 CFD problems that require LLMs to generate Python code for their numerical solution. Each problem is described in natural language and specifies the governing Partial Differential Equation (PDE), boundary and initial conditions, the spatial and temporal domain, and the target variable(s) to be computed and saved. The dataset includes both 1D and 2D problems, spanning linear and nonlinear PDEs, representative of those encountered in the CFD domain. Reference solutions are provided either as closed-form analytical expressions or as expert-authored Python implementations. Further details can be found in Appendix A.2. Our 24 coding problems span fluid mechanics, thermal transport, and turbulence, include both 1D and 2D simulation scenarios, extending beyond prior work in terms of complexity, which only evaluates the 1D heat transfer and 1D Burgers' equation [50], in both scope and complexity. Solving these problems requires not only reasoning about the physics but also integrating numerical methods, discretization schemes, and data handling into coherent, executable Python scripts containing 70 lines of code on average per problem.

FoamBench This task requires LLMs to generate all input files for an OpenFOAM simulation using the proper project folder structure and for the simulation to execute correctly, producing a physically accurate result with respect to a reference project. It consists of 126 OpenFOAM cases spread over more than 15 distinct geometric and physics scenarios. This dataset is further divided into two. *(1) FoamBench Basic*: This consists of 110 OpenFoam cases obtained from 11 tutorial cases [53]. We create variations within them by altering the boundary conditions and the parametric values on a case-specific basis (more details can be found in Appendix A.3.1). *(2) FoamBench Advanced*: This consists of 16 challenging OpenFOAM cases, which are not similar to the tutorials and are hand-crafted by CFD experts. Unlike *Basic*, the *Advanced* split tasks LLMs with choosing a proper turbulence model, creating a new geometry, and creating an appropriate mesh, based on the natural language input, without potentially relying on a tutorial project for guidance. For example, in the *Advanced* flow over double square case, the prompt specifies two square obstacles with details of their location and size. The LLM must correctly understand this prompt, then use appropriate one or more meshing tools from the OpenFOAM suite (e.g. blockMesh) to generate a valid computational mesh. Such cases bring us closer to real-world scenarios, where engineers analyze flow over complex geometries based on design specifications. Further details of the cases are provided in Appendix A.3.2.

For each case in *FoamBench*, the prompt (Appendix A.3.1) is designed to be concise and sufficient. The prompt contains (1) a clear description of the problem (e.g., flow over a cylinder), physical scenario (compressible or incompressible), geometry including computational domain and obstacle locations (with retrieval mechanisms handling complex geometries) and specifies the exact OpenFOAM solver for consistency; (2) the boundary conditions, relevant parameters (viscosity, Prandtl number), turbulence models (e.g., $k-\epsilon$, SA, LES), and specifies the timestep and solution-saving intervals for comparison against reference solutions.

3.2 Dataset Creation

In this section, we describe the dataset curation process. Due to the complex and technical nature of our benchmark, we relied on human experts at several stages during the creation of *CFDLLMBench*, involving them in both curation of data from existing sources, as well as authoring new content for the benchmark. A complete description of our process is presented in Appendix A.

Expert contributors For all three datasets, human experts curated or authored the initial set of problems. Our team of experts included six domain experts with advanced degrees and professional experience in the field of CFD, including two doctoral students, one Master's student, one undergraduate student, one post-doctoral researcher, and one university professor, with the latter two reviewing the work of the other four at each step. Despite being experts in CFD, they were still provided an orientation ahead of the curation process. For *CFDQuery*, the human experts created the multiple choice problems, and for *CFDCodeBench*, the human experts authored descriptions for the advanced problems by reviewing the source code. For *FoamBench*, the experts curated the dataset by varying parameters and boundary conditions for the tutorial problems, designing novel geometries for the non-tutorial cases, and authoring corresponding prompts based on the case files to guide LLMs in generating valid simulation setups. While the nature of the human work did not warrant an IRB review, we nevertheless followed all ethical norms and standards of the host academic institute when performing the human tasks for this dataset. All human experts involved are individuals involved in the project and well-compensated for their time.

Data sources For this benchmark, we ensured that we only used highly vetted data sources. The *CFDQuery* dataset was created exclusively for this benchmark, but the reference sources include university-level CFD lecture notes and vetted online sources. The problems in *CFDCodeBench* were curated from publicly available GitHub repositories and established numerical solver packages, including *CFD Python: the 12 Steps to Navier-Stokes Equations* repository [4] and *ENGR 491 - Computational Fluid Dynamics*, while more challenging scenarios were curated from the Dedalus Project [11]. For *FoamBench*, we curated the dataset based on the 11 OpenFOAM tutorials [53].

Quality assessment Since the solutions to our problems include objective, scientific answers, we did not perform traditional measures of human agreement. Rather, we went through an iterative process of review and revision of human work by independent experts to ensure the quality of the work. This review included both human-curated and human-authored portions of the benchmark.

3.3 Evaluation Metrics

Here we define expert-informed metrics used to assess performance on *CFDLLMBench*.

CFDQuery We evaluate multiple choice accuracy using a single standard accuracy metric, *Success Rate*, defined as ratio of the number of correctly answered questions to the total number of questions.

CFDCodeBench We evaluate an LLM's ability to generate executable and physically accurate python code for the numerical solution of a given CFD problem using four metrics. The holistic metric we use has three components: code executability, relative numerical error, and numerical convergence. We aggregate these three into a single score, which we call the *Success Rate*. **1) Executability** (M_{exec}): This is a binary metric which takes on a value of 1 if the LLM generated python code executes successfully and 0 is it is a failure. This metric is akin to the common pass@1 metric [12]. **2) Relative Error** (M_{NMSE}): We compare the LLM generated solution to the reference solution at the *final time of the prescribed simulation interval*. A normalized mean squared error percentage is calculated and a score is assigned based on the value of the NMSE percentage given by

$$\text{NMSE}\% = \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N} y_i^2} \times 100, \quad M_{\text{NMSE}} = \begin{cases} 1, & \text{NMSE} \leq 10\%\,, \\ 0.5, & 10\% < \text{NMSE} \leq 30\%\,, \\ 0, & \text{NMSE} > 30\%\,. \end{cases} \tag{1}$$

An M_{NMSE} of 0 means the solution is not physically accurate while a score of 0.5 is considered partial success. A score of 1 means the solution is acceptably accurate. **3) Numerical convergence** (M_{conv}): To evaluate the numerical convergence of the solution generated by the LLM, we refine both

the spatial and temporal discretization and assess the corresponding change in relative error. If the error decreases with mesh and time-step refinement, the solution is deemed convergent and awarded a score of 1; otherwise, it receives a score of 0. Unlike conventional LLM code generation benchmarks, we cannot rely on code similarity with respect to a reference solution, as numerical simulation code can vary significantly in implementation while yielding identical or equivalent solutions. **4) Success Rate**: We also define a stringent criterion to assess successful runs by looking at the fraction of problems where *all three* metrics achieve a score of 1. Specifically, defined for each problem i:

$$M_{\text{success}}^{(i)} = \begin{cases} 1, & M_{\text{exec}}^{(i)} = 1 \ \wedge \ M_{\text{NMSE}}^{(i)} = 1 \ \wedge \ M_{\text{conv}}^{(i)} = 1 \,, \\ 0, & \text{otherwise,} \end{cases} \quad \text{Success Rate} = \frac{1}{K} \sum_{i=1}^{K} M_{\text{success}}^{(i)}, \tag{2}$$

where K is the total number of problems. This provides us with a stringent measure of the percentage of problems within the benchmark where the model was able to produce an executable, physically accurate, and convergent solution.

FoamBench This task requires an LLM to create the required OpenFOAM input files, save them in appropriate directories, and call different solvers and tools within OpenFOAM to run a physically accurate simulation, all based on a natural language prompt. Prior work [13] focuses only on the ability of LLMs to generate files that produces a successful execution of OpenFOAM. Though executability is important, it does not capture the physical accuracy of the generated solution and thus fails to provide insights into whether the solution satisfies the user requirements. Text similarity metrics are widely used in comparing LLM-generated text to human text. For code generation, this is a useful metric for giving us an idea of how complete the files generated by LLMs are in comparison to the reference files, but again fails to provide the complete picture.

To tackle these challenges, we use four metrics to evaluate the LLM generated code, capturing code quality and physical accuracy of the solution, plus a holistic statistic, Success Rate. The details are as follows. **1) Executability** (M_{exec}): Similar to *CFDCodeBench*, we assign a value of 1 for successful execution of OpenFOAM using LLM generated case files and 0 otherwise. **2) Folder and File Structure** (M_{struct}): Generating the correct files and placing them in their respective folders is critical to the successful and accurate execution of the simulation workflow. The absence or misplacement of files can lead to failed execution of the case and/or inaccuracy of the generated output. Here, we use the ROUGE similarity metric [32] to compare the reference folder structure of the OpenFOAM cases with the LLM generated folder structure and provide a score between 0 and 1. **3) File Similarity** (M_{file}): This metric compares the content of the generated files with the reference OpenFOAM files using the ROUGE metric. **4) Relative Error** (M_{NMSE}): We use the same approach as *CFDCodeBench* Equation (1), comparing the LLM generated solution to a reference solution at the final time of the prescribed simulation window. **5) Success Rate**: We define Success Rate as the fraction of cases where just M_{exec} and M_{NMSE} achieves a score of 1.

3.4 Licensing, Accessibility, and Usability

All problems in our benchmark were collected from open, publicly available sources or were authored specifically for this benchmark. Accordingly, *CFDLLMBench* is released under the terms of the BSD 3-Clause License, making it free to use, modify, and redistribute, including for commercial purposes, provided that the license conditions are met. Our benchmark pipeline relies exclusively on free and open-source software, ensuring that it is accessible to all users without the need for paid subscriptions. Furthermore, we release not only the dataset (prompts), but also the complete codebase, fully containerized with Docker, to enable reproducibility. This comprehensive release allows future researchers to easily utilize, reproduce, or extend our benchmark with minimal overhead.

4 Experiments

In this section, we present results across a wide range of LLMs and agent frameworks that demonstrate the difficulty and realism of our benchmark.

4.1 Experimental Setup

For benchmark tasks, we compare the performance of five closed-weight models including Claude Sonnet 3.5 [2], o3-mini [38], Gemini 2.5 Flash [17], Claude Haiku 3.5 [2], and GPT-4o [37], and one open-source model Gemma-2-9B-IT [49]. The temperature parameter is set to 0.0 for the models in evaluation in all experiments, except for o3-mini, which does not allow us to change the default temperature parameters and the value of this parameter is undisclosed. On *CFDQuery* and *CFDCodeBench*, LLMs use a standard zero-shot prompt template that describes the task and the output format. For *FoamBench*, we evaluate LLMs zero-shot, as well as with agentic frameworks (described next). We use OpenFOAM v10 for all experiments.

Agentic frameworks for FoamBench Automating OpenFOAM using LLM is a complicated task, which we find benefits from agentic frameworks. Hence, for *FoamBench*, we not only compare various LLMs, but we also compare two agentic frameworks: *MetaOpenFoam* [13] and *Foam-Agent* [54]. Both of them assign agent roles for Retrieval-Augmented Generation (RAG) [30] , file generation, running, and reviewing (Reviewer). These components enable the system to retrieve files from similar simulations to use as exemplars and to get intermediate feedback for re-attempting file generation if necessary. To assess the individual contributions of these components, we benchmark three configurations: (1) with RAG, with Reviewer; (2) with RAG, without Reviewer (3) without RAG, with Reviewer. The absence of RAG and Reviewer indicates zero-shot LLM prompting-based generation, which is used as a baseline to compare the improvements due to these agent roles.

4.2 Results

The Success Rate of different models for the three benchmark tasks is shown in Figure 2. The *FoamBench* results are from the Foam-Agent framework, consisting of RAG and Reviewer, and using Sonnet 3.5, as this configuration yielded the strongest performance in our evaluations. Detailed *FoamBench* results are shown in Table 4. All closed-weight models perform well on *CFDQuery*, while the open sourced model could only answer 60% of the questions correctly. O3-mini performs the best in this task, which is not unexpected as it excels at logical reasoning and structured responses, producing 92% correct answers. On *CFDCodebench* and *FoamBench*, we see a drastic fall in Success Rate dropping to 14% in *CFDCodeBench* and 34% *FoamBench Basic* and 25% in *FoamBench Advanced* for the best performing models. It is interesting to note that Sonnet 3.5 performs the best among other models by some margin in *FoamBench*, which is not seen in the other tasks. However, it costs higher per run on average ($6.56) than, e.g., GPT-4o ($0.42)-see Table 5.

CFDCodeBench Figure 3 illustrates the breakdown of metric scores and Success Rate as defined in Section 3.3 for different models. The accuracy and convergence metrics highlight the importance of holistic evaluation beyond syntactic correctness, which is often lacking in studies.

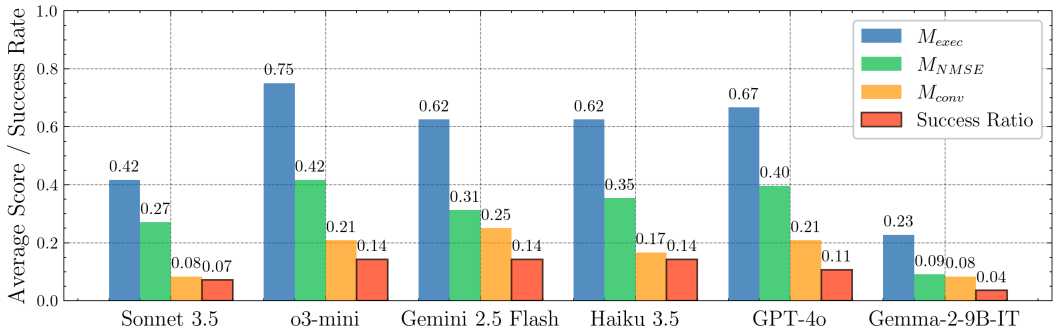

Figure 3: Average metric score and Success Rate for *CFDCodeBench*. The Success Rate for even the best performing models are around 14%, suggesting the challenging nature of the problems in this benchmark.

FoamBench Average metric scores and Success Rate of different models using the *Foam-Agent* framework with RAG and Reviewer is shown in Figure 4. Sonnet 3.5 was found to the best performing model for *FoamBench* tasks. The results of non-agentic zero-shot prompting with Sonnet 3.5

is provided in Table 1 to serve as a baseline for improvements due to the RAG and Reviewer roles (Table 2). This table also shows a comprehensive comparison between the two frameworks, MetaOpenFOAM and Foam-Agent, on *FoamBench* Basic and Advanced datasets. Detailed results on the impact of different models, framework and variations are provided in Appendix B.1.

Table 1: Zero-shot prompt LLM performance with Sonnet 3.5 (best performing model) on *FoamBench Basic* and *Advanced*.

Dataset	M_{exec}	M_{struct}	M_{file}	M_{NMSE}	*Success Rate*
FoamBench Basic	0.064	0.670	0.506	0.050	*0.045*
FoamBench Advanced	0.017	0.773	0.573	0.009	*0.007*

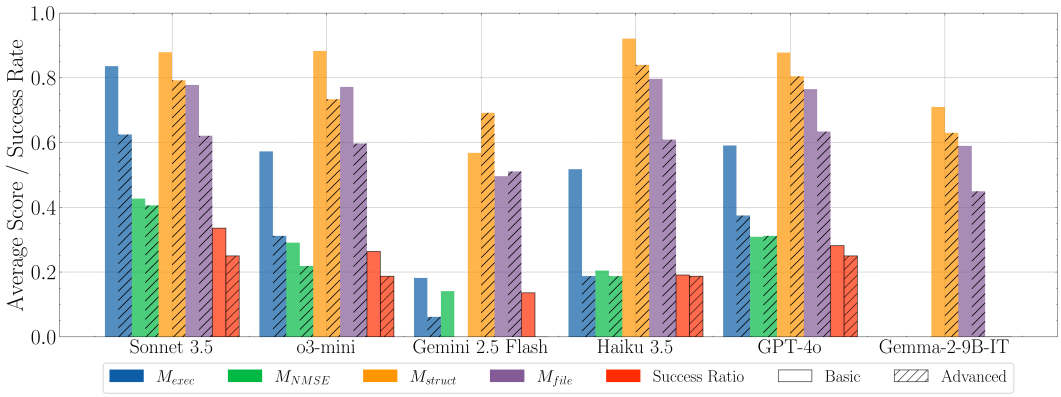

Figure 4: Average metric score and Success Rate for different models on *FoamBench* using *Foam-Agent* framework with RAG and reviewer. The Success Rate for even the best performing model (Sonnet 3.5) is 34% in basic dataset and 25% in the advanced dataset.

5 Discussion

Importance of physical and numerical accuracy metrics While all models demonstrate strong performance on *CFDQuery*—with Success Rate ranging from 60% (Gemma-2-9B-IT) to 92% (o3-mini), *performance significantly declines on tasks requiring physical and numerical accuracy*. To provide a holistic evaluation of model performance in *CFDCodeBench* and *FoamBench*, we reported multiple metrics and the stricter Success Rate. The latter aggregates success across code executability M_{exec}, numerical convergence M_{conv}, and physical accuracy M_{NMSE}, offering a practical view of model capabilities. From Figure 3, it is evident that most closed-weights models produce executable Python code in over 60% of cases, but these numbers are significantly worse for physical and numerical accuracy. For instance, in *FoamBench Basic*, the best Foam-Agent (Table 2) achieves good coding metrics $M_{\text{exec}} = 0.836$, $M_{\text{struct}} = 0.879$, $M_{\text{file}} = 0.778$, but the Success Rate is only 34% because of low physical accuracy. We see that the LLMs often fail to fully understand the prompts and lack domain-specific reasoning required to correctly apply fundamental CFD concepts—such as flux discretization schemes, appropriate time integration strategies, and consistent boundary treatments. This highlights a critical gap in current models' capabilities when it comes to generating reliable and physically consistent CFD code.

Zero-shot prompting for OpenFOAM Zero-shot prompting produces close to 0% Success Rate even for the best performing model (Sonnet 3.5) as shown in Table 1, highlighting the need for agentic frameworks when it comes to running OpenFOAM. For example, it is difficult for current LLMs to produce all of the required input files in a zero-shot manner. We observe that Sonnet 3.5 and o3-mini (Appendix B.1) have the most successful zero-shot runs.

Role of RAG and Reviewer RAG provides the framework with similar simulation files and the Reviewer allows for a trial and error approach to running OpenFOAM cases, mimicking human troubleshooting. The absence of either decreases the Success Rate by approximately 10% (Table 2), underscoring their critical roles in achieving optimal performance within the proposed framework.

Table 2: Component-wise mean scores and Success Rate for Claude Sonnet 3.5 on *FoamBench* Basic and Advanced, comparing MetaOpenFOAM vs. Foam-Agent.

Dataset	Variation	MetaOpenFOAM					Foam-Agent				
		M_{exec}	M_{struct}	M_{file}	M_{NMSE}	*Success Rate*	M_{exec}	M_{struct}	M_{file}	M_{NMSE}	*Success Rate*
FoamBench Basic	RAG + Reviewer	0.555	0.883	0.763	0.173	*0.136*	0.836	0.879	0.778	0.427	*0.336*
	RAG + No Reviewer	0.064	0.810	0.728	0.023	*0.009*	0.373	0.668	0.599	0.232	*0.200*
	No RAG + Reviewer	0.400	0.747	0.522	0.195	*0.145*	0.473	0.862	0.647	0.291	*0.245*
FoamBench Advanced	RAG + Reviewer	0.125	0.775	0.599	0.125	*0.125*	0.625	0.792	0.621	0.406	*0.250*
	RAG + No Reviewer	0.000	0.743	0.594	0.000	*0.000*	0.188	0.771	0.609	0.156	*0.125*
	No RAG + Reviewer	0.375	0.655	0.451	0.344	*0.187*	0.250	0.806	0.592	0.188	*0.125*

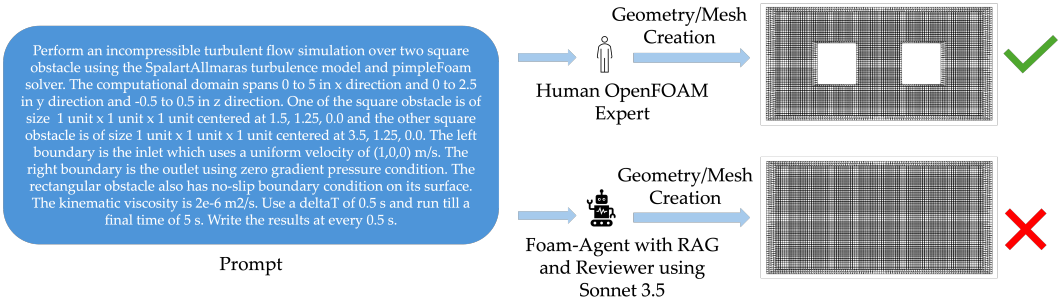

Figure 5: Comparison of the geometry and mesh generated by the *Foam-Agent* [54] (RAG and Reviewer) with Sonnet 3.5 for the `doubleSquare` case against human expert.

Spatial reasoning The CFD simulation workflows in *FoamBench* have preprocessing steps where a correct geometry and mesh file must be generated by the LLM. To handle real-world workflows, LLMs should be able to extrapolate to novel geometries. We highlight a particular case from *FoamBench Advanced*, `doubleSquare`, which is an incompressible flow over two square obstacles. The geometry produced by the *Foam-Agent*, in comparison to the reference geometry, is visualized in Figure 5. The prompt clearly defines the location of the obstacles, but the lack of spatial reasoning capabilities in LLMs appears to produce an incorrect geometry and mesh. We highlight that the ability of LLMs to understand geometry is a major area in need of improvement.

6 Limitations

First, one limitation of our work is that we currently do not provide human baselines for benchmark tasks. This is primarily due to the difficulty in determining appropriate human baselines, since the ability of a human to solve these problems depends on their domain knowledge, which is hard to quantify. For example, in a future iteration of the benchmark, we may explore adding a human baseline measuring the time taken for experts to solve these problems. Second, we did not perform an extensive automated prompt tuning for the baselines. Additional prompt engineering for the tasks, as well as automatic design of an agentic framework, may lead to stronger baseline performance.

7 Conclusion

In this work, we introduced *CFDLLMBench*, the first benchmark to holistically evaluate graduate-level knowledge, numerical and physical reasoning, and practical simulation capabilities of LLMs for CFD. We accomplish this by structuring the benchmark into three progressively challenging tiers, namely, *CFDQuery*, *CFDCodeBench*, and *FoamBench*. Our results highlight both the promise and the current limitations of LLMs in solving advanced scientific workflow automation problems, which require software expertise such as tool-calling and long-context understanding, as well as accurate physical modeling. We expect that *CFDLLMBench* will serve as a valuable testbed for advancing LLM capabilities in scientific computing, and encourage future work on domain-grounded, execution-based benchmarks across other areas of science and engineering.

References

[1] Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Anthropic. Claude 3.5 sonnet model card addendum. `https://www-cdn.anthropic.com/fed9cc193a14b84131812372d8d5857f8f304c52/Model_Card_Claude_3_Addendum.pdf`, 2024. Accessed: 2025-05-03.

[3] Austin, J., Odena, A., Nye, M., Bosma, M., Michalewski, H., Dohan, D., Jiang, E., Cai, C., Terry, M., Le, Q., et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

[4] Barba, L. A. and Forsyth, G. F. Cfd python: the 12 steps to navier-stokes equations. *Journal of Open Source Education*, 2(16):21, 2018.

[5] Beltagy, I., Lo, K., and Cohan, A. Scibert: A pretrained language model for scientific text. *arXiv preprint arXiv:1903.10676*, 2019.

[6] Blocken, B. Computational fluid dynamics for urban physics: Importance, scales, possibilities, limitations and ten tips and tricks towards accurate and reliable simulations. *Building and Environment*, 91:219–245, 2015.

[7] Blocken, B., Stathopoulos, T., Carmeliet, J., and Hensen, J. L. Application of computational fluid dynamics in building performance simulation for the outdoor environment: an overview. *Journal of building performance simulation*, 4(2):157–184, 2011.

[8] Bogin, B., Yang, K., Gupta, S., Richardson, K., Bransom, E., Clark, P., Sabharwal, A., and Khot, T. Super: Evaluating agents on setting up and executing tasks from research repositories. *arXiv preprint arXiv:2409.07440*, 2024.

[9] Boiko, D. A., MacKnight, R., Kline, B., and Gomes, G. Autonomous chemical research with large language models. *Nature*, 624(7992):570–578, 2023.

[10] Bran, A. M., Cox, S., Schilter, O., Baldassari, C., White, A. D., and Schwaller, P. Chemcrow: Augmenting large-language models with chemistry tools. *arXiv preprint arXiv:2304.05376*, 2023.

[11] Burns, K. J., Vasil, G. M., Oishi, J. S., Lecoanet, D., and Brown, B. P. Dedalus: A flexible framework for numerical simulations with spectral methods. *Physical Review Research*, 2(2): 023068, 2020.

[12] Chen, M., Tworek, J., Jun, H., Yuan, Q., de Oliveira Pinto, H. P., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavarian, M., Winter, C., Tillet, P., Such, F. P., Cummings, D., Plappert, M., Chantzis, F., Barnes, E., Herbert-Voss, A., Guss, W. H., Nichol, A., Paino, A., Tezak, N., Tang, J., Babuschkin, I., Balaji, S., Jain, S., Saunders, W., Hesse, C., Carr, A. N., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I., and Zaremba, W. Evaluating large language models trained on code, 2021. URL `https://arxiv.org/abs/2107.03374`.

[13] Chen, Y., Zhu, X., Zhou, H., and Ren, Z. Metaopenfoam: an llm-based multi-agent framework for cfd. *arXiv preprint arXiv:2407.21320*, 2024.

[14] Chen, Z., Chen, S., Ning, Y., Zhang, Q., Wang, B., Yu, B., Li, Y., Liao, Z., Wei, C., Lu, Z., et al. Scienceagentbench: Toward rigorous assessment of language agents for data-driven scientific discovery. *arXiv preprint arXiv:2410.05080*, 2024.

[15] Cherian, A., Corcodel, R., Jain, S., and Romeres, D. Llmphy: Complex physical reasoning using large language models and world models. *arXiv preprint arXiv:2411.08027*, 2024.

[16] Cui, H., Shamsi, Z., Cheon, G., Ma, X., Li, S., Tikhanovskaya, M., Norgaard, P., Mudur, N., Plomecka, M., Raccuglia, P., et al. Curie: Evaluating llms on multitask scientific long context understanding and reasoning. *arXiv preprint arXiv:2503.13517*, 2025.

[17] DeepMind, G. Start building with gemini 2.5 flash. `https://developers.googleblog.com/en/start-building-with-gemini-25-flash/?utm_source=deepmind.google&utm_medium=referral&utm_campaign=gdm&utm_content=`, 2025. Accessed: 2025-05-03.

[18] Glazer, E., Erdil, E., Besiroglu, T., Chicharro, D., Chen, E., Gunning, A., Olsson, C. F., Denain, J.-S., Ho, A., Santos, E. d. O., et al. Frontiermath: A benchmark for evaluating advanced mathematical reasoning in ai. *arXiv preprint arXiv:2411.04872*, 2024.

[19] Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[20] Jacobs, P. F. and Pollice, R. Developing large language models for quantum chemistry simulation input generation. *Digital Discovery*, 2025.

[21] Jadhav, Y. and Farimani, A. B. Large language model agent as a mechanical designer. *arXiv preprint arXiv:2404.17525*, 2024.

[22] Jasak, H., Jemcov, A., Tukovic, Z., et al. Openfoam: A c++ library for complex physics simulations. In *International workshop on coupled methods in numerical dynamics*, volume 1000, pp. 1–20. IUC Dubrovnik Croatia, 2007.

[23] Jiang, G., Ma, Z., Zhang, L., and Chen, J. Eplus-llm: A large language model-based computing platform for automated building energy modeling. *Applied Energy*, 367:123431, 2024.

[24] Jimenez, C. E., Yang, J., Wettig, A., Yao, S., Pei, K., Press, O., and Narasimhan, K. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770*, 2023.

[25] Kumar, V., Gleyzer, L., Kahana, A., Shukla, K., and Karniadakis, G. E. Mycrunchgpt: A llm assisted framework for scientific machine learning. *Journal of Machine Learning for Modeling and Computing*, 4(4), 2023.

[26] Lab, O. Engr 491: Computational fluid dynamics. `https://github.com/okcfdlab/engr491`, 2024. Accessed: 2025-05-16.

[27] Lai, Y., Li, C., Wang, Y., Zhang, T., Zhong, R., Zettlemoyer, L., Yih, W.-t., Fried, D., Wang, S., and Yu, T. Ds-1000: A natural and reliable benchmark for data science code generation. In *International Conference on Machine Learning*, pp. 18319–18345. PMLR, 2023.

[28] Lee, J. H., Michelis, M. Y., Katzschmann, R., and Manchester, Z. Aquarium: A fully differentiable fluid-structure interaction solver for robotics applications. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11272–11279. IEEE, 2023.

[29] Lee, Y., Lee, K., Park, S., Hwang, D., Kim, J., Lee, H.-i., and Lee, M. Qasa: advanced question answering on scientific articles. In *International Conference on Machine Learning*, pp. 19036–19052. PMLR, 2023.

[30] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.

[31] Li, W., Zhang, X., Guo, Z., Mao, S., Luo, W., Peng, G., Huang, Y., Wang, H., and Li, S. Feabench: A benchmark for evaluating repository-level code generation for feature implementation. *arXiv preprint arXiv:2503.06680*, 2025.

[32] Lin, C.-Y. Rouge: A package for automatic evaluation of summaries. pp. 10, 01 2004.

[33] Luo, R., Sun, L., Xia, Y., Qin, T., Zhang, S., Poon, H., and Liu, T.-Y. Biogpt: generative pre-trained transformer for biomedical text generation and mining. *Briefings in bioinformatics*, 23(6):bbac409, 2022.

[34] Majumder, B. P., Surana, H., Agarwal, D., Mishra, B. D., Meena, A., Prakhar, A., Vora, T., Khot, T., Sabharwal, A., and Clark, P. Discoverybench: Towards data-driven discovery with large language models. *arXiv preprint arXiv:2407.01725*, 2024.

[35] Mitchener, L., Laurent, J. M., Tenmann, B., Narayanan, S., Wellawatte, G. P., White, A., Sani, L., and Rodriques, S. G. Bixbench: a comprehensive benchmark for llm-based agents in computational biology. *arXiv preprint arXiv:2503.00096*, 2025.

[36] Narayanan, S., Braza, J. D., Griffiths, R.-R., Ponnapati, M., Bou, A., Laurent, J., Kabeli, O., Wellawatte, G., Cox, S., Rodriques, S. G., et al. Aviary: training language agents on challenging scientific tasks. *arXiv preprint arXiv:2412.21154*, 2024.

[37] OpenAI. Hello gpt-4o. `https://openai.com/index/hello-gpt-4o/`, 2024. Accessed: 2025-05-03.

[38] OpenAI. Openai o3-mini. `https://openai.com/index/openai-o3-mini/`, 2024. Accessed: 2025-05-03.

[39] Pandey, S., Xu, R., Wang, W., and Chu, X. Openfoamgpt: A retrieval-augmented large language model (llm) agent for openfoam-based computational fluid dynamics. *Physics of Fluids*, 37(3), 2025.

[40] Qin, Y., Liang, S., Ye, Y., Zhu, K., Yan, L., Lu, Y., Lin, Y., Cong, X., Tang, X., Qian, B., et al. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*, 2023.

[41] Rein, D., Hou, B. L., Stickland, A. C., Petty, J., Pang, R. Y., Dirani, J., Michael, J., and Bowman, S. R. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.

[42] Shah, M., Norris, S. E., Turner, R., and Flay, R. G. A review of computational fluid dynamics application to investigate tropical cyclone wind speeds. *Natural Hazards*, 117(1):897–915, 2023.

[43] Shi, G., Shi, X., O'Connell, M., Yu, R., Azizzadenesheli, K., Anandkumar, A., Yue, Y., and Chung, S.-J. Neural lander: Stable drone landing control using learned dynamics. In *2019 international conference on robotics and automation (icra)*, pp. 9784–9790. IEEE, 2019.

[44] Siegel, Z. S., Kapoor, S., Nagdir, N., Stroebl, B., and Narayanan, A. Core-bench: Fostering the credibility of published research through a computational reproducibility agent benchmark. *arXiv preprint arXiv:2409.11363*, 2024.

[45] Singhal, K., Tu, T., Gottweis, J., Sayres, R., Wulczyn, E., Amin, M., Hou, L., Clark, K., Pfohl, S. R., Cole-Lewis, H., et al. Toward expert-level medical question answering with large language models. *Nature Medicine*, pp. 1–8, 2025.

[46] Slotnick, J. P., Khodadoust, A., Alonso, J., Darmofal, D., Gropp, W., Lurie, E., and Mavriplis, D. J. Cfd vision 2030 study: a path to revolutionary computational aerosciences. Technical report, 2014.

[47] Starace, G., Jaffe, O., Sherburn, D., Aung, J., Chan, J. S., Maksin, L., Dias, R., Mays, E., Kinsella, B., Thompson, W., et al. Paperbench: Evaluating ai's ability to replicate ai research. *arXiv preprint arXiv:2504.01848*, 2025.

[48] Taylor, R., Kardas, M., Cucurull, G., Scialom, T., Hartshorn, A., Saravia, E., Poulton, A., Kerkez, V., and Stojnic, R. Galactica: A large language model for science. *arXiv preprint arXiv:2211.09085*, 2022.

[49] Team, G. Gemma. 2024. doi: 10.34740/KAGGLE/M/3301. URL `https://www.kaggle.com/m/3301`.

[50] Tian, M., Gao, L., Zhang, S., Chen, X., Fan, C., Guo, X., Haas, R., Ji, P., Krongchon, K., Li, Y., et al. Scicode: A research coding benchmark curated by scientists. *Advances in Neural Information Processing Systems*, 37:30624–30650, 2024.

[51] Wang, X., Hu, Z., Lu, P., Zhu, Y., Zhang, J., Subramaniam, S., Loomba, A. R., Zhang, S., Sun, Y., and Wang, W. Scibench: Evaluating college-level scientific problem-solving abilities of large language models. *arXiv preprint arXiv:2307.10635*, 2023.

[52] Wang, X., Hu, Z., Lu, P., Zhu, Y., Zhang, J., Subramaniam, S., Loomba, A. R., Zhang, S., Sun, Y., and Wang, W. Scibench: Evaluating college-level scientific problem-solving abilities of large language models. *arXiv preprint arXiv:2307.10635*, 2023.

[53] Weller, H. G., Tabor, G., Jasak, H., and Fureby, C. A tensorial approach to computational continuum mechanics using object-oriented techniques. *Computers in physics*, 12(6):620–631, 1998.

[54] Yue, L., Somasekharan, N., Cao, Y., and Pan, S. Foam-agent: Towards automated intelligent cfd workflows. *arXiv preprint arXiv:2505.04997*, 2025.

[55] Zhang, X., Dong, Y., Wu, Y., Huang, J., Jia, C., Fernando, B., Shou, M. Z., Zhang, L., and Liu, J. Physreason: A comprehensive benchmark towards physics-based reasoning. *arXiv preprint arXiv:2502.12054*, 2025.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state the creation of a benchmark suite (CFDLLMBench) targeting three core CFD-related competencies, which is consistently supported by the rest of the paper's experiments and scope.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 6 of the paper explicitly acknowledges limitations, including the lack of human baselines and minimal prompt tuning, and discusses how these could impact results and future directions in overcoming them.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification:The paper does not present formal theoretical results or proofs; it is an empirical benchmark study.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The benchmark datasets, evaluation metrics, and experimental setups are thoroughly described, with mention of Docker-based reproducibility and code/data release (Section 3.4). The code is provided to the reviewers and will be fully public by camera ready deadline. The datasets are also shared using private link in Harvard Dataverse. The data too will be made public by the camera ready deadline. The code and dataset are awaiting internal review before public release.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The authors state that the dataset and containerized codebase will be released under a permissive open-source license (e.g., BSD 3-Clause) (Section 3.4). The code and dataset will be made available to reviewers and will be made public by the corresponding deadline for the same. The code and data are undergoing internal review before public release. The codebase is compressed into a zip file and uploaded to the Supplementary Material.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 4.1 describes the models used, prompting formats, agent configurations, and evaluation frameworks; additional dataset details are included in the Appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: While detailed metric breakdowns are provided, there are no error bars or statistical variability reported for performance metrics across runs. Error bars would require multiple experimental runs, which would be costly considering the experiments are done using closed-weight LLMs.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [NA]

Justification: The study does not involve usage of GPU time for training. All models are called using closed weights model API. Hence it did not account for any compute time of the authors. Further, the token usage for the models including API cost and number of reviewer runs are given in the appendix (Table 5).

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: The authors mention adherence to institutional ethical standards, and no ethical violations are apparent in data usage or methodology (Section 3.2).

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.

- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: The paper focuses on technical aspects and does not include a broader impact section analyzing societal implications of automating CFD workflows.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [Yes]

    Justification: The paper does not release any pretrained language models or scraped datasets and poses no high risk for misuse. It provides only a benchmark suite (CFDLLMBench) based on curated or openly available scientific problems, with no sensitive or dual-use content.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The paper uses only open-source and publicly available datasets, such as OpenFOAM tutorials [53], CFD Python, and Dedalus. These sources are properly cited (see Section 3.2), and the authors affirm that all assets are used under appropriate licenses without restrictions. The license and accessibility are addressed in Section 3.4.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: The paper introduces CFDLLMBench, a new benchmark suite consisting of three datasets (CFDQuery, CFDCodeBench, and FoamBench). The datasets are described in detail in Sections 3.1–3.4, and the paper states that the full codebase and assets will be released with a Docker container and documentation to support reproducibility.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [Yes]

Justification: The paper does not involve crowdsourcing but does involve expert human contributors for dataset creation. Section 3.2 confirms that all contributors were part of the project team and were "well-compensated" in accordance with ethical norms of the host academic institution.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: No IRB approval was required because the human work (curation by expert collaborators) did not involve traditional human-subjects research. The paper explicitly notes that IRB review was not warranted but that ethical norms were followed (Section 3.2).

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: The entire benchmark is explicitly designed to evaluate LLMs, and their use is central to all experiments. Multiple models and agentic frameworks involving LLMs are described and compared in detail (Sections 1, 3, and 4).

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.