MetaOpenFOAM: an LLM-based multi-agent framework for CFD

Yuxuan Chen[a], Xu Zhu[a], Hua Zhou[a], Zhuyin Ren[a*]

[a] *Institute for Aero Engine, Tsinghua University, Beijing 100084, China*

* *Corresponding author:* zhuyinren@tsinghua.edu.cn

Submitted to **Computer Physics Communications**

MetaOpenFOAM：一种基于LLM的多智能体框架用于计算流体动力学

陈宇轩[a]、朱旭[a]、周华[a]、任竹韵[a*]

[a] 清华大学航空发动机研究院，中国北京100084

* 通讯作者：zhuyinren@tsinghua.edu.cn

Submitted to **计算机物理通讯**

Abstract

Remarkable progress has been made in automated problem solving through societies of agents based on large language models (LLMs). Computational fluid dynamics (CFD), as a complex problem, presents unique challenges in automated simulations that require sophisticated solutions. MetaOpenFOAM, as a novel multi-agent collaborations framework, aims to complete CFD simulation tasks with only natural language as input. These simulation tasks include mesh pre-processing, simulation and post-processing, etc. MetaOpenFOAM harnesses the power of MetaGPT's assembly line paradigm, which assigns diverse roles to various agents, efficiently breaking down complex CFD tasks into manageable subtasks. Langchain further complements MetaOpenFOAM by integrating Retrieval-Augmented Generation (RAG) technology, which enhances the framework's ability by integrating a searchable database of OpenFOAM tutorials for LLMs. Tests on a benchmark for natural language-based CFD solver, consisting of eight CFD simulation tasks, have shown that MetaOpenFOAM achieved a high pass rate per test (85%), with each test case costing only $0.22 on average. The eight CFD simulation tasks encompass a range of multidimensional flow problems, covering compressible and incompressible flows with different physical processes such as turbulence, heat transfer and combustion. This demonstrates the capability to automate CFD simulations using only natural language input, iteratively correcting errors to achieve the desired simulations at a low cost. An ablation study was conducted to verify the necessity of each component in the multi-agent system and the RAG technology. A sensitivity study on the randomness of LLM showed that LLM with low randomness can obtain more stable and accurate results. Additionally, MetaOpenFOAM owns the ability to identify and modify key parameters in user requirements, excels in correcting bugs when failure match occur, and enhances simulation capabilities through human participation, which demonstrates the generalization of MetaOpenFOAM.

Keywords: large language models, multi-agent system, computational fluid dynamics, retrieval-augmented generation, automated simulation

摘要

基于大语言模型（LLMs）的智能体社会在自动化问题求解领域取得了显著进展。计算流体力学（CFD）作为复杂问题，在自动化模拟中提出了需要复杂解决方案的独特挑战。MetaOpenFOAM作为一种新颖的多智能体协作框架，旨在仅以自然语言作为输入完成CFD模拟任务。这些模拟任务包括网格预处理、模拟及后处理等环节。MetaOpenFOAM利用MetaGPT的流水线范式，为不同智能体分配多样化角色，将复杂CFD任务高效分解为可管理的子任务。Langchain通过集成检索增强生成（RAG）技术进一步增强了该框架的能力，该技术通过为大语言模型整合OpenFOAM教程的可搜索数据库来实现。在基于自然语言的CFD求解器基准测试中（包含八项CFD模拟任务），MetaOpenFOAM展现出高通过率（85%），每个测试用例平均仅需$0.22成本。这八项CFD模拟任务涵盖多维流动问题，包括涉及湍流、传热和燃烧等不同物理过程的可压缩与不可压缩流动。这证明了仅通过自然语言输入即可实现CFD模拟自动化，并通过迭代纠错以低成本达成目标模拟的能力。消融研究验证了多智能体系统中各组件与RAG技术的必要性。针对LLM随机性的敏感性研究表明，低随机性的LLM能获得更稳定且准确的结果。此外，MetaOpenFOAM还具备识别并修改用户需求中关键参数的能力，在发生匹配失败时擅长错误修正，并通过人工参与增强模拟能力，展现了其强大的泛化能力。

关键词: 大语言模型, 多智能体系统, computational fluid dynamics, retrieval-augmented generation, automated simulation

1. Introduction

Computational Fluid Dynamics (CFD) is a discipline that uses numerical methods and physical models to solve fluid mechanics problems [1]. Since the introduction of CFD, scientists and engineers have employed complex code to simulate and predict fluid behavior, with open-source software like OpenFOAM being a notable and mature example [2]. Although OpenFOAM has been successfully applied in many fields [3-6], it still requires researchers to possess high-level programming and specialized skills. As technology progresses, automated tools and user-friendly interfaces have emerged, allowing users to perform complex CFD simulations with simply clicking buttons on GUI, leading to the development of industrial software like Fluent [7] and COMSOL [8]. However, even with these improvements, conducting CFD simulations remains highly technical, requiring specialized knowledge and substantial manual operations. Recently, the rapid advancement of natural language processing (NLP) technologies, particularly the advent of large language model (LLM) [9-14], has brought new hope to CFD research, promising to revolutionize the field.

The emergence of LLM represents a significant breakthrough in natural language processing. LLM can understand and generate natural language, handling vast amounts of information and providing intelligent feedback. Recent advancements, such as GPT-4 [15], Llama 2 [16], and ChatGLM [17], have demonstrated powerful capabilities in tasks like translation, text generation, and question-answering. However, despite their impressive performance in many tasks, single LLM still face limitations in solving complex problems requiring extensive text generation [10, 13, 14, 18-20]. For example, CFD problems typically involve intricate geometric modeling, physical modeling, and numerical methods, which exceed the current capabilities of individual LLM. To harness the potential of LLM in the CFD field, new approaches are needed to enhance their ability to tackle complex problems.

To address the limitations of single LLM in solving complex problems, the development of Multi-Agent System (MAS) has emerged as a promising approach [9-14, 20]. MAS involves the collaboration of multiple intelligent agents to complete complex tasks, with each agent focusing on different sub-tasks or domains, thereby improving the overall system's efficiency and accuracy. Notably, MAS can achieve unsupervised adversarial generation by iteratively having certain agents evaluate the text generated by other agents, helping refine and enhance the precision of the generated

1. 引言

计算流体力学 (CFD) 是一门运用数值方法和物理模型解决流体力学问题的学科 [1]。自CFD问世以来，科学家和工程师们便通过复杂代码来模拟和预测流体行为，其中开源软件OpenFOAM便是典型且成熟的代表[2]。尽管OpenFOAM已在诸多领域成功应用 [3-6], ，但仍要求研究者具备高阶编程能力和专业知识。随着技术进步，自动化工具和友好界面相继涌现，用户仅需点击图形界面按钮即可完成复杂CFD仿真，由此催生了Fluent [7] 和COMSOL[8]等工业软件。但即便如此，开展CFD仿真仍具有高度专业性，需要专门知识和大量人工操作。近年来，自然语言处理 (NLP) 技术的快速发展，尤其是大语言模型 (LLM) 的出现[9-14], ，为CFD研究带来了新希望，有望彻底革新该领域。

LLM的出现标志着自然语言处理领域的重大突破。LLM能够理解并生成自然语言，处理海量信息并提供智能反馈。近期诸如GPT-4 [15], 、Llama 2 [16], 和ChatGLM [17], 等进展，已在翻译、文本生成和问答等任务中展现出强大能力。然而，尽管在许多任务中表现优异，单个LLM在解决需要大量文本生成的复杂问题[10, 13, 14,18-20]时仍存在局限。例如，计算流体动力学问题通常涉及复杂的几何建模、物理建模和数值方法，这些已超出当前单个LLM的能力范围。为充分发挥LLM在计算流体动力学领域的潜力，需要新方法来增强其处理复杂问题的能力。

为解决单一LLM在解决复杂问题时的局限性，多智能体系统(MAS)的发展已成为一种颇具前景的解决方案[9-14, 20]。MAS通过多个智能体协作完成复杂任务，每个智能体专注于不同的子任务或领域，从而提升整体系统的效率与准确性。值得注意的是，MAS可通过迭代式对抗生成实现无监督学习——某些智能体对其他智能体生成的文本进行评估，从而帮助优化生成内容的精确度

text to better meet user requirements. In CFD simulation tasks, MAS can assign different agents to handle CFD task division, input file writing, CFD simulation, and simulation result evaluation. The evaluating agents can provide feedback to other agents, optimizing their outputs. Once implemented, this natural language-based MAS approach to CFD simulation can significantly lower the barrier to conduct CFD simulations and reduce the workload for researchers, making CFD studies more efficient and accessible.

Currently, various LLM-based tools have been developed to facilitate the collaboration and integration of multi-agent systems. MetaGPT [11] and Langchain [21] are two notable tools in this field. MetaGPT is a framework designed for multi-agent collaboration, enabling the coordination of multiple LLM agents to tackle complex CFD problems. Through MetaGPT, researchers can chain different LLM agents together, each playing a specific role, to accomplish tasks collectively. Langchain, on the other hand, is a tool for Retrieval-Augmented Generation (RAG) technology. By effectively integrating information from multiple documents, Langchain can provide more professional support for CFD research. Building on the MetaGPT and Langchain frameworks, this paper develops MetaOpenFOAM, a natural language-based CFD simulation framework. MetaOpenFOAM takes user requirements as input, generates OpenFOAM input files through LLM, and returns simulation results that meet user requirements after automatically running OpenFOAM.

The structure of this paper is as follows: first, the basic framework of MetaOpenFOAM and the implementation method of RAG are introduced. Next, a series of test datasets and evaluation methods for CFD simulations with natural language input are presented. Following this, the results of MetaOpenFOAM are quantitatively introduced, along with an ablation study and parameter sensitivity analysis. Finally, the results of MetaOpenFOAM are qualitatively analyzed.

2. Methodology

2.1 MetaOpenFOAM Framework

As shown in Figure 1, MetaOpenFOAM is architected to interpret user requirements, decompose them into manageable subtasks, and execute these subtasks through a series of specialized agents. The framework leverages the MetaGPT assembly line paradigm to assign distinct roles to each agent, ensuring efficient task execution and error handling.

The framework is divided into four primary roles, each with specific responsibilities and actions:

文本以更好地满足用户需求。在CFD模拟任务中，MAS可分配不同智能体分别处理CFD任务划分、输入文件编写、CFD模拟及仿真结果评估。评估智能体可向其他智能体提供反馈，优化其输出结果。实施过程中，

这种基于自然语言的多智能体系统（MAS）CFD模拟方法，能显著降低开展CFD仿真的门槛，减轻研究人员的工作负担，使CFD研究更高效且易于普及。

目前，已开发出多种基于大语言模型（LLM）的工具来促进多智能体系统的协作与集成。MetaGPT [11] 和Langchain [21] 是该领域两个值得关注的工具。MetaGPT是一个专为多智能体协作设计的框架，可协调多个LLM智能体共同解决复杂CFD问题。通过MetaGPT，研究人员能将不同LLM智能体串联起来，每个智能体扮演特定角色，协同完成任务。而Langchain则是检索增强生成（RAG）技术的工具，通过有效整合多份文档信息，能为CFD研究提供更专业的支持。本文基于MetaGPT和Langchain框架，开发了基于自然语言的CFD模拟框架MetaOpenFOAM。该框架以用户需求为输入，通过LLM生成OpenFOAM输入文件，并在自动运行OpenFOAM后返回符合用户需求的模拟结果。

本文结构如下：首先介绍MetaOpenFOAM的基本框架和RAG的实现方法；其次展示针对自然语言输入CFD仿真的一系列测试数据集与评估方法；随后定量呈现MetaOpenFOAM的结果，包括消融研究和参数敏感性分析；最后对MetaOpenFOAM的结果进行定性分析。

2. 方法论

2.1 MetaOpenFOAM框架

如图1所示，MetaOpenFOAM被设计用于解析用户需求，将其分解为可管理的子任务，并通过一系列专业智能体执行这些子任务。该框架采用MetaGPT流水线范式为每个智能体分配特定角色，确保任务高效执行与错误处理。

该框架划分为四个主要角色，每个角色都有特定的职责和动作：

Architect (Role): This role is responsible for the initial interpretation of the user's natural language requirements. The Architect converts user requirements into a specified format, finds similar cases from the database, creates the input architecture, and oversees the overall workflow to ensure that the user's specifications are accurately translated into actionable tasks.

Actions of Architect:

Find similar cases in the database of OpenFOAM cases and tutorials.

Create the input architecture according to the similar case.

InputWriter (Role): The InputWriter role focuses on generating and refining the necessary input files for the CFD simulation. This involves writing initial input files, rewriting them as necessary based on feedback or errors, and ensuring that all files conform to OpenFOAM's requirements.

Actions of InputWriter:

Write input files for OpenFOAM based on the architecture provided by the Architect.

Rewrite input files for OpenFOAM if modifications or corrections are needed.

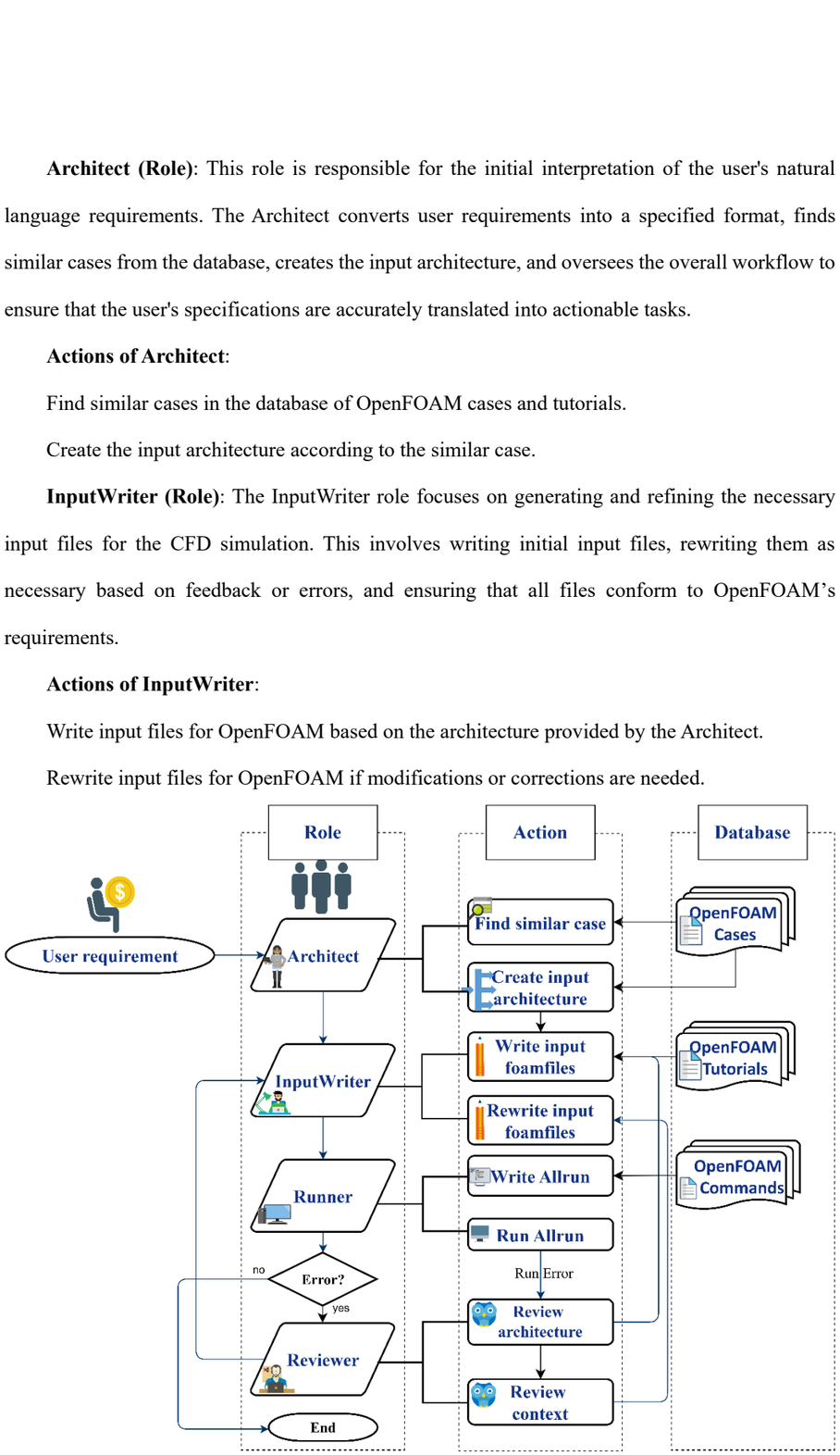


Figure 1. The framework of MetaOpenFOAM

Runner (Role): The Runner executes the CFD simulation using OpenFOAM. This role ensures

架构师 (角色): 该角色负责对用户自然语言的初始解读

语言要求。架构师将用户需求转换为指定格式，查找

数据库中的相似案例，创建输入架构，并监督整体工作流程以确保

用户需求被准确转化为可执行任务。

架构师的动作:

在OpenFOAM案例及教程的数据库中查找相似案例。

根据相似案例创建输入架构。

输入写入器 (角色): 输入写入器角色专注于为CFD模拟生成和完善必要的输入文件。这包括编写初始输入文件、根据反馈或错误进行必要的重写，并确保所有文件符合OpenFOAM的要求。

输入写入器的动作:

根据架构师提供的架构，为OpenFOAM编写输入文件。

如需修改或校正，则为OpenFOAM重写输入文件。

图1. MetaOpenFOAM的框架

运行器 (角色): 运行器使用OpenFOAM执行CFD模拟。该角色确保

that the simulation runs smoothly and monitors for any possible errors during execution.

Actions of Runner:

Write the Allrun script to automate the simulation execution process.

Run the Allrun script to perform the CFD simulation.

Reviewer (Role): The Reviewer's role is to analyze any errors that occur during the simulation, identify the relevant file that caused the error, and report these findings back to the InputWriter. This role is to check the file architecture based on existing information to determine whether the error is caused by the absence of related files or the incorrect contents that need modifications.

Actions of Reviewer:

Review the input architecture.

Review error context.

Procedure of MetaOpenFOAM

First, the Architect takes the user requirements and splits them into several subtasks, which are then given to the InputWriter. The InputWriter creates the necessary OpenFOAM input files and passes them to the Runner. The Runner writes the Allrun script and executes the OpenFOAM simulation. If an error occurs, the Runner provides the execution error command and error context to the Reviewer. The Reviewer examines the file architecture and context to identify and solve the error, then returns the revised instructions to the InputWriter. The InputWriter either writes new files following the revised architecture or corrects the error in the input files. This loop repeats until no errors occur, or it reaches a user-defined maximum number of iterations or maximum investment (based on tokens). The prompts of the main action in each role are shown in Appendix A.

2.2 Retrieval-Augmented Generation (RAG) based on Langchain

Langchain's RAG technology is a critical component that supports MetaOpenFOAM by integrating a searchable database of OpenFOAM official documents and tutorials. This system enhances the agents' ability to perform their tasks by providing relevant information and contextual guidance, ensuring that the framework can handle a wide range of CFD scenarios with minimal user intervention.

模拟过程平稳运行，并监控执行期间可能出现的任何错误。

运行器动作:

编写Allrun脚本以自动化模拟执行流程。

运行Allrun脚本以执行CFD模拟。

评审员（角色）：评审员的职责是分析模拟过程中出现的任何错误，识别导致错误的相关文件，并将这些发现反馈给输入写入器。该角色需基于现有信息检查文件架构，以确定错误是由相关文件缺失还是需要修改的错误内容引起的。

评审员动作:

审查输入架构。

审查错误上下文。

MetaOpenFOAM流程

首先，架构师接收用户需求并将其拆分为若干子任务，随后将这些子任务交给输入写入器。输入写入器创建必要的OpenFOAM输入文件并传递给运行器。运行器编写Allrun脚本并执行OpenFOAM模拟。若发生错误，运行器将执行错误命令及错误上下文提供给评审员。评审员通过检查文件架构和上下文来识别并解决错误，随后将修订指令返回给输入写入器。输入写入器根据修订后的架构编写新文件或直接在输入文件中修正错误。该循环将持续至无错误发生，或达到用户定义的迭代次数上限或最大资源投入（基于令牌）。各角色主要动作的提示详见附录A。

2.2 基于Langchain的检索增强生成（RAG）

Langchain的RAG技术是一个关键组件，它通过整合OpenFOAM官方文档和教程的可搜索数据库来支持MetaOpenFOAM。该系统通过提供相关信息及上下文指导，增强了智能体执行任务的能力，确保该框架能够以最少的用户干预处理广泛的CFD场景。

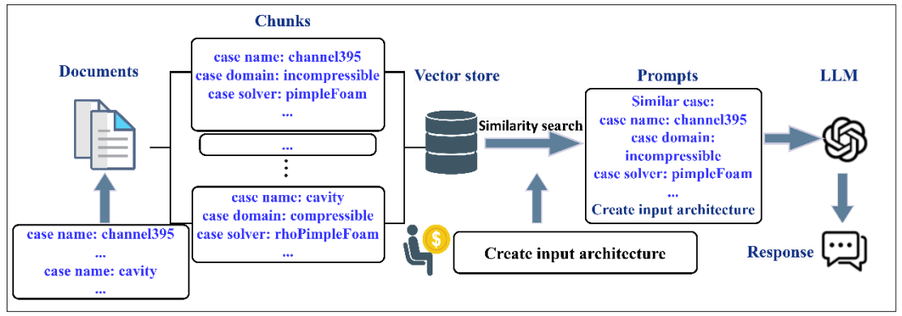


Figure 2. The procedure of Retrieval-Augmented Generation (RAG)

Figure 2 shows the procedure of RAG in action “Find similar case” of MetaOpenFOAM. Firstly, based on the tutorials provided by OpenFOAM, a database containing file structures is constructed. These documents are then split into chunks. Each individual case is divided into separated chunks. A vector store is then created using these chunks. After saving the database, it only requires querying to retrieve the most similar chunks. These chunks are then combined with the user message as input for the LLM, completing the entire RAG process. And for other actions of MetaOpenFOAM, the procedure of RAG is similar, and only the database should be changed into file context or file command, corresponding to the OpenFOAM tutorials and OpenFOAM commands in Figure 1. During the actions "Write input OpenFOAM files" and "Rewrite input OpenFOAM files," documents containing tutorial file contents are required. For the action "Write the Allrun file," documents containing the collection of OpenFOAM execution commands are needed to ensure that the written commands comply with the standards. Specific examples of these documents can be found in Appendix B.

The more detailed document segmentation facilitates more accurate matching during retrieval. The necessity of the above data-enhanced retrieval (RAG) method will be validated in Section 4.1.

3. Experiment

3.1 Setup

MetaGPT v0.8.0 [11] was selected to integrate various LLMs, while OpenFOAM 10 [2] was utilized for CFD computations due to its stability and reliability as an open-source CFD solver. GPT-4o [15] was chosen as the representative LLM because of its outstanding performance. The temperature, as a parameter of LLM controlling randomness of generated text, was set to 0.01 to ensure highly focused and deterministic outputs, which results in low randomness of the generated

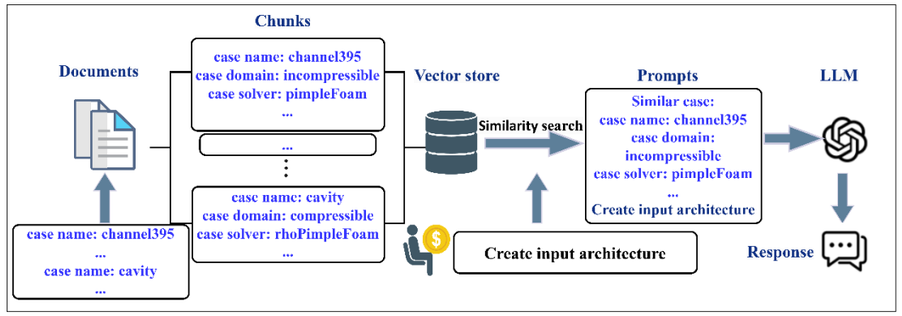


图2. 检索增强生成（RAG）流程

图2展示了MetaOpenFOAM在“查找相似案例”动作中RAG的流程。首先，基于OpenFOAM提供的教程，构建一个包含文件结构的数据库。这些文档随后被分割成分块，每个独立案例被划分为单独的分块。接着利用这些分块创建向量存储。保存数据库后，仅需通过查询即可检索最相似的分块。这些分块随后与用户消息结合作为LLM的输入，从而完成整个RAG过程。对于MetaOpenFOAM的其他动作，RAG流程类似，只需将数据库替换为与图1中OpenFOAM教程和OpenFOAM命令相对应的文件上下文或文件命令。在执行“编写输入OpenFOAM文件”和“重写输入OpenFOAM文件”动作时，需要包含教程文件内容的文档。而对于“编写Allrun文件”动作，则需要包含OpenFOAM执行命令集合的文档，以确保编写的命令符合标准。这些文档的具体示例可参见附录B。

更详细的文档分割有助于在检索过程中实现更精准的匹配。

上述数据增强检索（RAG）方法的必要性将在第4.1节得到验证。

3. 实验

3.1 实验设置

MetaGPT v0.8.0 [11] 被选用来集成各种大语言模型，而OpenFOAM 10 [2] 则因其作为开源CFD求解器的稳定性和可靠性被用于CFD计算。GPT-4o [15] 因其出色的性能被选为代表大语言模型。温度作为控制生成文本随机性的大语言模型参数，被设置为0.01以确保高度聚焦和确定性输出，从而使生成内容具有较低的随机性。

text. The influence of temperature on performance is evaluated in Section 4.2.

For RAG technology, LangChain v0.1.19 [21] was employed to link the LLM and the database. The FAISS vector store [22], known for its efficiency and user-friendliness, was used as the database vector store, and OpenAIEmbeddings were selected for embedding the data chunks. The “similarity” method was utilized for matching similar chunks. The combination of retrieved documents and user messages represents the simplest form of stacking. More details could be found in the code: https://github.com/Terry-cyx/MetaOpenFOAM

3.2 Benchmarking Natural Language Input for CFD Solvers

Currently, there are no public benchmarks for CFD user requirements to validate CFD solvers that take natural language as input. Therefore, this paper derives several common simulation requirements from OpenFOAM tutorials and lists several relevant simulations needs.

It is important to note that, from the user's perspective, natural language-based CFD user requirements are generally incomplete. They cannot cover all the necessary information required for input files and can only provide the most essential details, such as the case name, case category, solver, mesh, boundary conditions, and initial conditions. Therefore, in the constructed cases below, only partial information is provided as user requirements, aligning with the usage habits of natural language.

Eight cases, covering a range of multidimensional flow problems including both 2D and 3D flows, various compressible and incompressible flows with various physical processes such as turbulence, heat transfer, and combustion, were tested. These cases involve Direct Numerical Simulation (DNS) and the turbulence models of Reynolds-Averaged Navier-Stokes (RANS) and Large Eddy Simulation (LES). Both compressible and incompressible solvers are included, along with description for Newtonian and non-Newtonian fluids. All these cases were modified from OpenFOAM tutorials, with specific parameters adjusted accordingly.

① HIT: do a DNS simulation of incompressible forcing homogeneous isotropic turbulence with Grid 32^3 using dnsFoam
② PitzDaily: do a LES simulation of incompressible pitzDaily flow using pisoFoam with inlet velocity = 5 m/s
③ Cavity: do a 2D RANS simulation of incompressible cavity flow using pisoFoam, with RANS model: RNGkEpsilon, grid 15*15*1
④ LidDrivenCavity: do an incompressible lid driven cavity flow simulation with the top wall moves in the x direction at a speed of 1 m/s while the other three are stationary

文本。 温度对性能的影响将在第4.2节进行评估。

在RAG技术中，我们采用LangChain v0.1.19 [21] 来连接LLM和数据库。选用以高效易用著称的FAISS向量存储[22], 作为数据库向量存储，并采用OpenAI嵌入对数据块进行向量化处理。“相似性”

方法被用于匹配相似分块。检索到的文档与用户消息的组合代表了最基础的堆叠形式。更多细节可参阅代码：https://github.com/Terry-cyx/MetaOpenFOAM

3.2 CFD求解器的自然语言输入基准测试

目前尚未有公开的CFD用户需求基准测试来验证以自然语言为输入的CFD求解器。因此，本文从OpenFOAM教程中提取若干常见模拟需求，并列举相关模拟要求。

需注意的是，从用户视角看，基于自然语言的CFD用户需求通常是不完整的。这些需求无法涵盖输入文件所需的全部信息，只能提供最关键的要素，如案例名称、案例类别、求解器、网格、边界条件和初始条件。因此下文构建的案例中，仅提供部分信息作为用户需求，以符合自然语言的使用习惯。

测试了八个案例，涵盖了一系列多维流动问题，包括二维和三维流动、各种可压缩和不可压缩流动，涉及湍流、传热和燃烧等不同物理过程。这些案例包括直接数值模拟（DNS）以及雷诺平均纳维-斯托克斯（RANS）和大涡模拟（LES）的湍流模型。同时包含可压缩和不可压缩求解器，并对牛顿流体和非牛顿流体进行了描述。所有案例均基于OpenFOAM教程修改，并相应调整了特定参数。

1 HIT：使用dnsFoam对不可压缩强迫均匀各向同性湍流进行直接数值模拟，网格32^3

2 PitzDaily: 使用pisoFoam对不可压缩pitzDaily流动进行大涡模拟，入口速度= 5 m/s

3 Cavity: 使用pisoFoam对不可压缩空腔流动进行二维雷诺平均模拟，RANS模型：RNGkEpsilon，网格15*15*1

4 LidDrivenCavity：进行不可压缩顶盖驱动空腔流动模拟，顶壁以1米/秒的速度沿x方向移动，其余三面保持静止

⑤ SquareBendLiq: do a compressible simulation of squareBendLiq of using rhoSimpleFoam with endTime = 100, deltaT = 1, and writeInterval = 10

⑥ PlanarPoiseuille: do a laminar simulation of incompressible planar Poiseuille flow of a non-Newtonian fluid with grid 1*20*1, modelled using the Maxwell viscoelastic laminar stress model, initially at rest, constant pressure gradient applied from time zero

⑦ CounterFlowFlame: do a 2D laminar simulation of counterflow flame using reactingFoam in combustion with grid 50*20*1

⑧ BuoyantCavity: do a RANS simulation of buoyantCavity using buoyantFoam, which investigates natural convection in a heat cavity with a temperature difference of 20K is maintained between the hot and cold; the remaining patches are treated as adiabatic.

3.3 Evaluation Metrics

It is essential to establish evaluation metrics to assess the performance of natural language-based CFD solvers. These metrics need to consider common indicators in the CFD domain, such as successful mesh generation and convergence of calculations, as well as computational costs and generation success rates (pass@k in [23]) relevant to the LLM domain. Therefore, we evaluate the practical performance of natural language-based CFD solvers using the following five metrics: the first four metrics A, B, C, D for single experiments and the last metric E being added for multiple experiments.

Single experiments can be evaluated by the following metrics:

(A) Executability: This metric rates input files on a scale from 0 (failure) to 4 (flawless). A score of '0' indicates grid generation failure, '1' indicates grid generation success but running failure, '2' indicates that the case is runnable but does not converge, '3' indicates that the case runs to the endTime specified in the controlDict, and '4' indicates flawless input foam files, which not only run to the endTime but also meet all user requirements. A single experiment is considered to have passed the test if its executability score reaches '4'. Among them, ‘1’ to ’3’ can be judged automatically by the program, but ‘4’ requires human participation.

(B) Cost: The cost evaluation includes (1) running time, (2) number of iterations, (3) token usage, and (4) expenses, which are proportional to token usage.

(C) Code Statistics: This metric includes (1) the number of input files, (2) the number of lines per input file, and (3) the total number of lines in the input files.

(D) Productivity: This metric is defined as the number of tokens used divided by the number of lines in the input files, representing token consumption per input line.

For multiple experiments, a new metric needs to be added:

5 SquareBendLiq: 使用rhoSimpleFoam对方形弯管液体进行可压缩模拟，结束时间= 100，时间步长= 1，写入间隔= 10

6 平面泊肃叶: 对非牛顿流体进行不可压缩平面泊肃叶流的二维层流模拟，使用网格1*20*1，采用麦克斯韦粘弹性层流应力模型建模，初始静止，从时间零点施加恒定压力梯度

7 逆流火焰: 使用反应泡沫在燃烧中进行二维逆流火焰的层流模拟，网格为50*20*1

8 浮力腔: 使用浮力泡沫进行浮力腔的RANS模拟，研究热腔内维持20K温差的自然对流现象，热和冷壁面之间保持温差，其余壁面视为绝热

3.3 评估指标

建立评估指标对衡量基于自然语言的CFD求解器性能至关重要。这些指标需兼顾CFD领域的常见指标（如成功网格生成和计算收敛）以及与LLM领域相关的计算成本和生成成功率（[23]中的通过率@k）。因此，我们通过以下五项指标评估基于自然语言的CFD求解器实际性能：前四项指标A、B、C、D针对单次实验，最后新增指标E用于多次实验。

单个实验可通过以下指标进行评估：

(A) 可执行性：该指标以0（失败）到4（完美）的等级对输入文件进行评分。'0'分表示网格生成失败，'1'分表示网格生成成功但运行失败，'2'分表示案例可运行但不收敛，'3'分表示案例运行至控制字典中指定的结束时间，'4'分表示完美的输入foam文件，不仅运行至结束时间且满足所有用户需求。当可执行性评分达到'4'时，该实验即被视为通过测试。其中，'1'至'3'可由程序自动判定，但'4'分需人工参与评估。

(B) 成本：成本评估包含(1)运行时间、(2)迭代次数、(3)令牌使用量及(4)费用（与令牌使用量成正比）。

(C) 代码统计: 该指标包括 (1) 输入文件数量, (2) 每个输入文件的行数以及 (3) 输入文件总行数。

(D) 生产力: 该指标定义为使用的令牌数量除以输入文件的行数，表示每输入行的令牌消耗。

针对多项实验，需要新增一个指标：

(E) Pass@k [23]: This metric represents the probability that at least one of the k generated input file samples passes the unit tests. It measures the model's ability to generate correct input files within k attempts. We follow the unbiased version of pass@k as presented by Chen et al. (2021a) and Dong et al. (2023) to evaluate the pass@k of MetaOpenFOAM

$$pass@k := \underset{problems}{E}[1 - \frac{\binom{n-c}{k}}{\binom{n}{k}}],$$

where n represents the number of input sets generated for each user requirement, and c represents the number of these samples that pass the test, i.e., achieve an executability score of 4. To evaluate pass@k, we generate $n \geqslant k$ samples per task (with n = 10 and k = 1 in this paper), count the number of correct samples $c \leqslant n$ which pass unit tests, and calculate the unbiased estimator.

3.4 Main results

Table 1 Performance of MetaOpenFOAM

	Executability	**Token Usage**	**Iteration**	**Productivity**	**Pass@1(%)**
HIT	4	12667	2.4	36.3	100
PitzDaily	4	18083	2.1	32.1	100
Cavity	4	12863	0	28.3	100
LidDrivenCavity	2.8	52090	12.5	149.7	60
SquareBendLiq	4	16385	0	27.6	100
PlanarPoiseuille	3.7	35532	5.2	81.3	90
CounterFlowFlame	3.7	47927	7.2	20.3	90
BuoyantCavity	2.4	156812	16.3	161.3	40
Average	3.6	44045	5.7	67.1	85

Table 1 presents the performance of MetaOpenFOAM on eight test cases from the benchmark proposed in Section 3.2. The complete flow chart for the individual examples is described in detail in Appendix C, using HIT as an example. Only a selection of key metrics from Section 3.3 is displayed, while the rest are provided in Appendix D. Each metric for single experiments is the average of n tests (n=10). For the iteration metric in the cost, maximum iteration is set to 20 in the program to prevent infinite iterations. If executability does not reach 3 or above after 20 iterations, the test is automatically marked as failed and the iteration is terminated.

(E) 通过率@k [23]：该指标表示在k次生成的输入文件样本中至少有一个通过单元测试的概率。它衡量模型在k次尝试内生成正确输入文件的能力。我们采用Chen等人(2021a)和Dong等人(2023)提出的无偏通过率@k版本来评估MetaOpenFOAM的通过率@k

$$pass@k := \underset{problems}{E}[1 - \frac{\binom{n-c}{k}}{\binom{n}{k}}],$$

其中n表示针对每个用户要求生成的输入集数量，c表示
这些样本中通过测试（即获得可执行性评分4分）的数量。为评估
通过率@k，我们为每项任务生成$n \geqslant k$个样本（本文中n = 10 和k = 1），统计
通过单元测试的正确样本数量$c \leqslant n$，并计算无偏估计量。

3.4 主要结果

表1 MetaOpenFOAM的性能

	可执行性	令牌使用量	迭代	生产力	通过率@1(%)
HIT	4	12667	2.4	36.3	100
PitzDaily	4	18083	2.1	32.1	100
腔体	4	12863	0	28.3	100
盖驱动腔	2.8	52090	12.5	149.7	60
方形弯管液体	4	16385	0	27.6	100
平面泊肃叶	3.7	35532	5.2	81.3	90
逆流火焰	3.7	47927	7.2	20.3	90
浮力腔	2.4	156812	16.3	161.3	40
平均值	3.6	44045	5.7	67.1	85

表1展示了MetaOpenFOAM在基于第3.2节提出的基准测试中八个测试用例上的性能表现。附录C以HIT为例详细描述了各案例的完整流程图。仅展示了第3.3节中的部分关键指标，其余指标详见附录D。单项实验的各指标均为n次测试（n=10）的平均值。在成本项的迭代指标中，程序将最大迭代次数设为20以防止无限迭代。若可执行性评分在20次迭代后仍未达到3分及以上，则测试自动标记为失败并终止迭代。

Overall, the average pass rate (pass@1) of 85% and high executability score 3.6 demonstrate the outstanding performance of MetaOpenFOAM. On average, each test case requires 44,045 tokens. Given a cost of $5 per 1M tokens, generating one test case costs only $0.22, and each line of input file consumes an average of 67.1 tokens, costing just $0.0003, which is significantly lower than manual labor costs. Therefore, in terms of lowering the barrier to use, reducing labor costs, and increasing efficiency, MetaOpenFOAM is revolutionary.

For different test cases, it can be seen that HIT, PitzDaily, Cavity, and SquareBendLiq have an executability score of 4, meaning flawless results that satisfy all user requirements. These cases require fewer iterations and fewer tokens. However, for cases with lower executability scores, such as BuoyantCavity and LidDrivenCavity, the LLM fails to correctly modify errors, resulting in more iterations and higher token usage, and consequently, a lower pass@1.

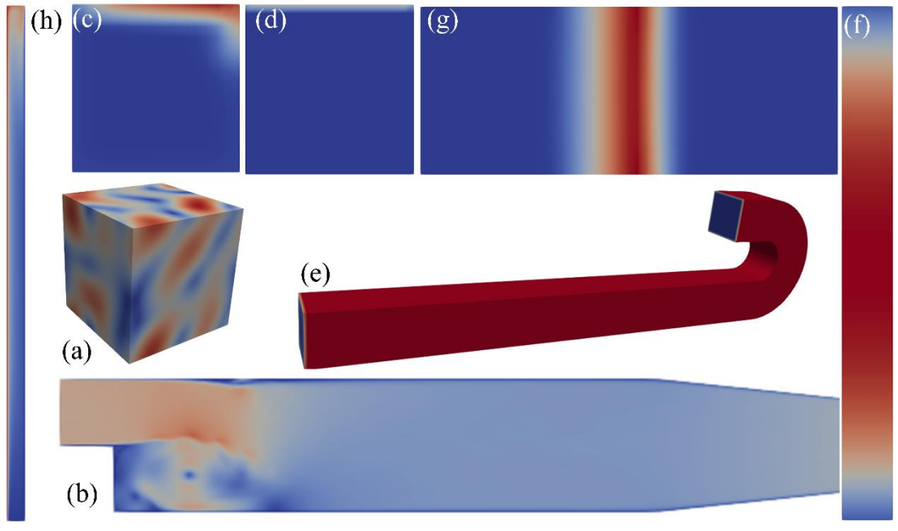


Figure 3. Demo simulation results simulated by MetaOpenFOAM. (a) Homogeneous Isotropic Turbulence (HIT) (b) PitzDaily (c) Cavity (d) Lid-Driven Cavity (e) Square Bend Liquid (SquareBendLiq) (f) Planar Poiseuille (g) Counter Flow Flame (h) Buoyant Cavity.

Analyzing the correlation between iteration and token usage, we find a Pearson correlation coefficient of 0.89 with a p-value of 0.0013. This indicates a strong positive correlation between iteration and token usage, and this correlation is statistically significant (p-value much less than 0.05). The reason is evident: more iterations mean more input to the LLM. Therefore, in the subsequent discussion, we will focus only on iteration and not token usage, due to their strong

总体而言，85%的平均通过率（pass@1）和3.6的高可执行性评分彰显了MetaOpenFOAM的卓越性能。平均每个测试用例需要消耗44,045个令牌。

假设每100万令牌的成本为5美元，生成一个测试用例仅需0.22美元，而输入文件的每一行平均消耗67.1个令牌，成本仅为0.0003美元，远低于人工成本。因此，在降低使用门槛、减少人工成本和提高效率方面，MetaOpenFOAM具有革命性意义。

对于不同的测试用例，可以看出HIT、皮茨日报、腔体和方形弯管液体的可执行性评分为4，意味着结果完美满足所有用户需求。这些案例需要较少的迭代次数和令牌。然而，对于可执行性评分较低的案例，如浮力腔和盖驱动腔，LLM未能正确修正错误，导致更多迭代次数和更高的令牌使用量，从而降低了通过率@1。

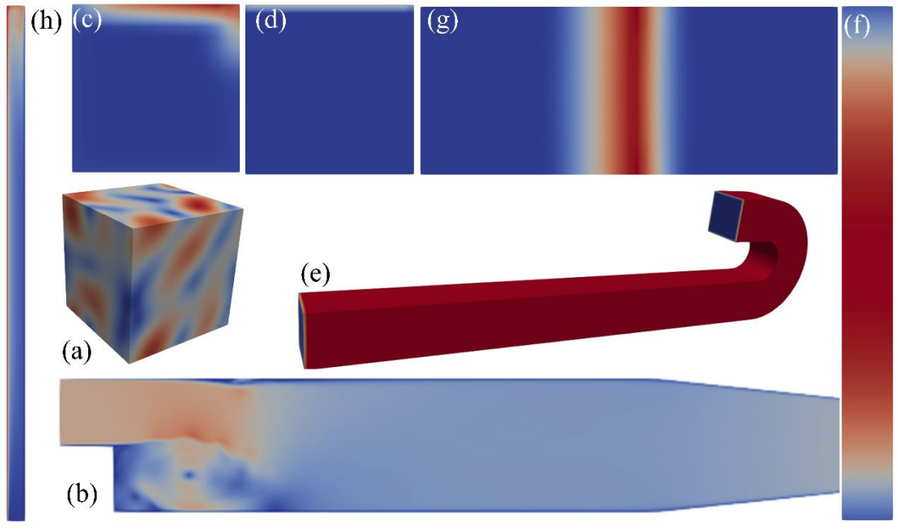


图3. MetaOpenFOAM模拟的演示结果。(a) 均匀各向同性湍流（HIT） (b) 皮茨日报 (c) 腔体 (d) 顶盖驱动空腔 (e) 方形弯管液体（ SquareBendLiq ） (f) 平面泊肃叶 (g) 逆流火焰 (h) 浮力空腔。

通过分析迭代次数与令牌使用量之间的相关性，我们发现皮尔逊相关系数为0.89，p值为0.0013。这表明迭代次数与令牌使用量之间存在强烈的正相关性，且该相关性具有统计学意义（p值远小于0.05）。原因显而易见：更多的迭代次数意味着向LLM输入更多内容。因此，在后续讨论中，我们将仅关注迭代次数而非令牌使用量，因为它们之间存在强

correlation.

As shown in Figure 3, we visualized MetaOpenFOAM's contour plots for eight test examples using paraview [24]. Detailed results about performance and test examples are provided in Appendix D.

4. Discussion

This section will discuss the necessity of each component in MetaOpenFOAM as well as sensitivity analysis of key parameters related to large language models and qualitative analysis of some MetaOpenFOAM results.

4.1 Ablation Analysis

To validate the necessity of each component in MetaOpenFOAM, we removed some components to quantitatively discuss how their removal influences the model outputs. Specifically, since MetaOpenFOAM primarily consists of Roles, Actions, and RAG, the ablation study focused on these three aspects.

A. Remove “Reviewer” Role

As shown in section 2.1, there are 4 roles in MetaOpenFOAM where the Architect, InputWriter, and Runner are essential roles. Removing any of these would result in an executability of 0, making ablation analysis meaningless. However, the Reviewer is not strictly necessary, allowing for comparison with and without it. It is important to note that the Reviewer is the key component of MetaOpenFOAM that distinguishes automated multi-agent collaboration from a simple aggregation of single agents. Therefore, examining how the Reviewer affects model outputs is vital.

B. Remove “Review architecture” Action

Each role involves several actions. Some actions, like creating the input architecture, writing OpenFOAM input files, and running OpenFOAM, are essential. Omitting these actions would obviously prevent the tests from passing. Therefore, we only considered the non-essential action of “Review architecture”. If “Review architecture” is missing, errors related to wrong file architecture become difficult to resolve.

C. Remove RAG

RAG technology is another key component of MetaOpenFOAM, providing professional knowledge missing from general LLMs. To verify the importance of RAG, we tested the performance of MetaOpenFOAM without it.

Table 2: The ablation study on role, action and RAG. ‘#’ denotes ‘Remove’, ‘#Reviewer’ means

相关性。

如图3所示，我们使用paraview [24]可视化了MetaOpenFOAM在八个测试案例中的等高线图。关于性能与测试案例的详细结果见附录D。

4. 讨论

本节将探讨MetaOpenFOAM中各组件的必要性，以及与大型语言模型相关的关键参数的敏感性分析，并对部分MetaOpenFOAM结果进行定性分析。

4.1 消融分析

为验证MetaOpenFOAM各组件的必要性，我们移除了部分组件以定量讨论其移除对模型输出的影响。由于MetaOpenFOAM主要由角色、动作和RAG构成，消融研究聚焦于这三个方面。

A. 移除“评审员”角色

如第2.1节所示，MetaOpenFOAM中有4个角色，其中架构师、输入写入器和运行器是核心角色。移除其中任何一个都会导致可执行性降为0，使得消融分析失去意义。然而评审员并非绝对必要，因此可以对比其存在与否的影响。需特别指出的是，评审员是MetaOpenFOAM的关键组件，正是它使得自动化多智能体协作区别于单一智能体的简单聚合。因此，研究评审员如何影响模型输出至关重要。

B. 移除“评审架构”动作

每个角色都涉及若干动作。创建输入架构、编写OpenFOAM输入文件和运行OpenFOAM等动作是必不可少的，省略这些动作显然会导致测试无法通过。因此我们仅考虑非必要的“评审架构”动作。若缺失“评审架构”动作，与错误文件架构相关的错误将难以解决。

C. 移除RAG

RAG技术是MetaOpenFOAM的另一关键组件，可补充通用大语言模型所缺乏的专业知识。为验证RAG的重要性，我们测试了未集成该技术时MetaOpenFOAM的性能。

表2: 关于角色、动作和RAG的消融研究。‘#’表示‘移除’，‘#评审员’意味着

remove Reviewer, ‘#Nothing’ means remove nothing, i.e., the complete and original MetaOpenFOAM, ‘Arc.’ denotes the input architecture.

Statistical Index	#Reviewer	#Review Arc.	#RAG	#Nothing
(A) Executability	1.7	3.0	0.8	**3.6**
(B) Cost: Running time (s)	**126**	292	332	271
(B) Cost: Iteration (max: 20)	**0**	7.7	19.2	5.7
(B) Cost: Token Usage	**15661**	69336	81346	44045
(C) Code Statistic: Input Files	12.8	13.2	**10.1**	13.7
(C) Code Statistic: Lines per Input File	**42.0**	42.5	124.5	49.9
(C) Code Statistic: Total lines of Input Files	**540**	572	1491	760
(D) Productivity	**29.8**	143.0	158.2	67.1
(E) pass@1 (%)	27.5	70	0	**85**

Table 2 shows the performance of MetaOpenFOAM when the Reviewer (role), Review input architecture (action), RAG, or nothing is removed. After removing the Reviewer role, the pass@1 of MetaOpenFOAM dropped from 85% to 27.5%. Without the Reviewer, no iterations occurred, resulting in lower running time and cost. However, the significantly lower pass@1 and executability indicate a substantial decline in MetaOpenFOAM's applicability and utility. As shown in Figure 4, apart from the cavity and squareBendLiq cases, which require no iterations, all other cases failed. This demonstrates that without the Reviewer role, MetaOpenFOAM cannot handle CFD simulation tasks with moderate complexity. When the “Review architecture” action was removed, the pass@1 also decreased from 85% to 70%. This decline mainly stemmed from the HIT and LidDrivenCavity cases, where file architecture-related errors (for example, "cannot find file XX") occurred. Without the “Review architecture” action, MetaOpenFOAM could not create a new file architecture to resolve these issues, resulting in repeated modifications to existing files. Thus, the “Review architecture” action is verified as necessary.

For the RAG module, another key component of MetaOpenFOAM, removing RAG resulted in a pass@1 of 0, indicating that MetaOpenFOAM without RAG could not pass any CFD simulation

移除评审员， ‘#无’表示不进行移除，即完整保留原始MetaOpenFOAM中， ‘架构’表示输入架构。

统计指标	#评审员	#评审架构	#RAG	#无
(A) 可执行性	1.7	3.0	0.8	**3.6**
(B) 成本：运行时间（秒）	**126**	292	332	271
(B) 成本：迭代（最大：20）	**0**	7.7	19.2	5.7
(B) 成本：令牌使用量	15661	69336	81346	44045
(C) 代码统计：输入文件	12.8	13.2	**10.1**	13.7
(C) 代码统计：每个输入文件的行数	**42.0**	42.5	124.5	49.9
(C) 代码统计：输入文件总行数	**540**	572	1491	760
(D) 生产率	**29.8**	143.0	158.2	67.1
(E) pass@1 (%)	27.5	70	0	**85**

表2展示了当移除评审员（角色）、评审输入架构（动作）、RAG或无任何操作时MetaOpenFOAM的性能表现。移除评审员角色后，MetaOpenFOAM的通过率@1从85%骤降至27.5%。由于缺乏评审员，系统未进行任何迭代，导致运行时间和成本降低。但通过率@1与可执行性的大幅下降表明MetaOpenFOAM的适用性和实用性显著减弱。如图4所示，除无需迭代的腔体和方形弯管液体案例外，其余案例全部失败。这证明没有评审员角色时，MetaOpenFOAM无法处理中等复杂度的CFD模拟任务。当移除“评审架构”动作时，通过率@1也从85%降至70%。该下降主要源于HIT和盖驱动腔案例中出现的文件架构相关错误（例如“找不到XX文件”）。缺乏“评审架构”动作时，系统无法创建新的文件架构来解决这些问题，导致对现有文件进行反复修改。由此验证“评审架构”动作的必要性。

对于RAG模块——MetaOpenFOAM的另一关键组件，移除RAG导致通过率@1为0，这表明没有RAG的MetaOpenFOAM无法通过任何CFD模拟

tasks in the database. However, executability was not 0 because some simulation tasks could complete mesh generation (Executability=1), run but not converge (Executability=2), or run successfully but not meet user requirements (Executability=3). For instance, in the PitzDaily test, MetaOpenFOAM without RAG could even achieve an executability of 3, but the simulated case used a randomly generated 20*20*20 cubic mesh instead of the PitzDaily mesh. This shows that without RAG, MetaOpenFOAM loses the ability to complete CFD simulation tasks, likely due to the lack of training data for constructing OpenFOAM input files in LLM.

Detailed data for the entire ablation study are presented in table form in Appendix E.

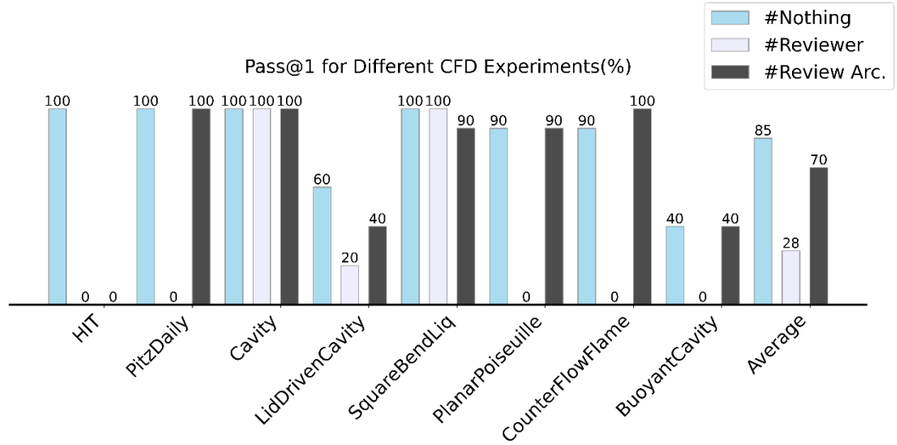


Figure 4 pass@1 of ablation study on role, action. '#' denotes 'Remove', '#Reviewer' means remove Reviewer, '#Nothing' means remove nothing, i.e., the complete and original MetaOpenFOAM, 'Arc.' denotes the input architecture.

4.2 Influence of parameter "temperature"

In LLM, the "temperature" is a key parameter, which controls the randomness and creativity of the generated text. When the temperature is high (for example near 1), the probability distribution becomes flatter, making the model more likely to choose words with less probability. The generated text becomes more diverse and creative, but it may also be less coherent or sensible. When the temperature is low (for example near 0), the probability distribution becomes sharper, making the model more likely to choose the most probable words. The generated text becomes more conservative and coherent but less diverse and creative.

Table 3. The sensitivity analyzes of parameter 'temperature' of LLM, where 'temp.' means

数据库中的任务。然而，可执行性并未归零，因为部分模拟任务能够完成网格生成（可执行性=1）、运行但未收敛（可执行性=2）、或成功运行但未满足用户需求（可执行性=3）。例如在PitzDaily测试中，无RAG的MetaOpenFOAM甚至能达到3的可执行性，但模拟案例使用了随机生成的20*20*20立方体网格而非PitzDaily网格。这表明没有RAG时，MetaOpenFOAM丧失了完成CFD模拟任务的能力，这很可能源于LLM中缺乏构建OpenFOAM输入文件的训练数据。

完整的消融研究详细数据以表格形式呈现在附录E中。

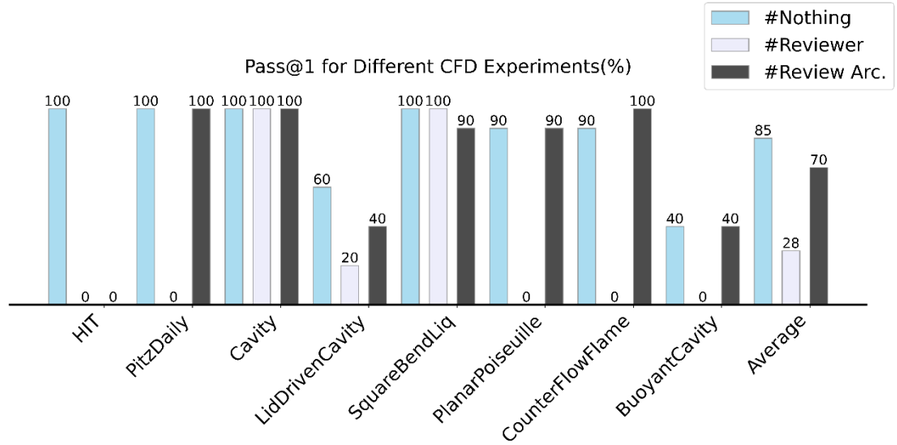


图4展示了角色和动作消融研究的通过率@1。'#'表示'移除'，'#评审员'意为移除评审员，'#无'表示不进行任何移除，即完整保留原始MetaOpenFOAM，'架构'代表输入架构。

4.2 参数"温度"的影响

在LLM中，"温度"是关键参数，它控制生成文本的随机性和创造性。当温度较高（例如接近1）时，概率分布趋于平缓，模型更倾向于选择概率较低的词汇。生成的文本更具多样性和创造性，但也可能降低连贯性或合理性。当温度较低（例如接近0）时，概率分布更为集中，模型更倾向于选择最高概率的词汇。生成的文本更为保守和连贯，但多样性和创造性会减弱。

表3. LLM参数'温度'的敏感性分析，其中'温度'表示

‘temperature’.

Statistical Index	temp.=0.01	temp.=0.5	temp.=0.99
(A) Executability	**3.6**	3.4	2.3
(B) Cost: Running time (s)	**271**	357	350
(B) Cost: Iterations (max: 20)	**5.7**	7.4	13.5
(B) Cost: Token Usage	**44045**	58419	109195
(C) Code Statistic: Input Files	**13.7**	14.4	14.1
(C) Code Statistic: Lines per Input File	49.9	59.9	**43.3**
(C) Code Statistic: Total lines of Input Files	760	874	**614**
(D) Productivity	**67.1**	87.5	179.3
(E) pass@1 (%)	**85**	83	48

Table 3 shows the average evaluation metrics after running the database of test cases 10 times at three different temperatures (0.01, 0.5, 0.99). It is evident that when the temperature is set to 0.01, both (A) Executability and (B) Cost, as well as (D) Productivity and (E) pass@1, are optimal on average. This suggests that conservative and coherent generated text leads to more stable and accurate results.

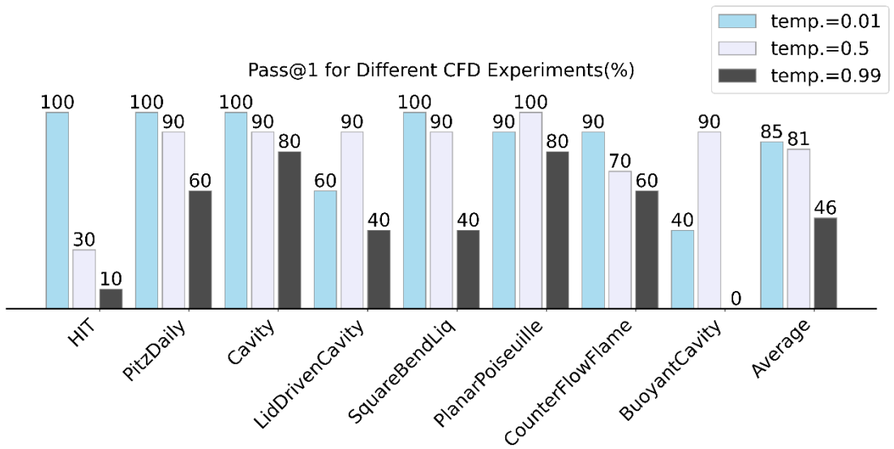


Figure 5 Average pass@1 for different CFD experiments with 3 different temperatures (0,01, 0.5, 0.99) of LLM, where ‘temp.’ means ‘temperature’.

Figure 5 shows the average pass@1 for the eight test cases under three different temperatures

‘温度’.

统计指标	温度=**0**.01	温度=**0**.5	温度=**0**.99
(A) 可执行性	**3.6**	3.4	2.3
(B) 成本：运行时间（秒）	**271**	357	350
(B) 成本：迭代次数（最大值：20）	**5.7**	7.4	13.5
(B) 成本：令牌使用量	44045	58419	109195
(C) 代码统计：输入文件	**13.7**	14.4	14.1
(C) 代码统计：每个输入文件的行数	49.9	59.9	**43.3**
(C) 代码统计：输入文件总行数	760	874	**614**
(D) 生产率	**67.1**	87.5	179.3
(E) 通过率@1 (%)	**85**	83	48

表3展示了在三种不同温度(0.01、0.5、0.99)下运行测试用例数据库10次后的平均评估指标。显然，当温度设置为0.01时，(A)可执行性和(B)成本，以及(D)生产力和(E)通过率@1在平均值上均达到最优。这表明保守且连贯的生成文本能带来更稳定且准确的结果。

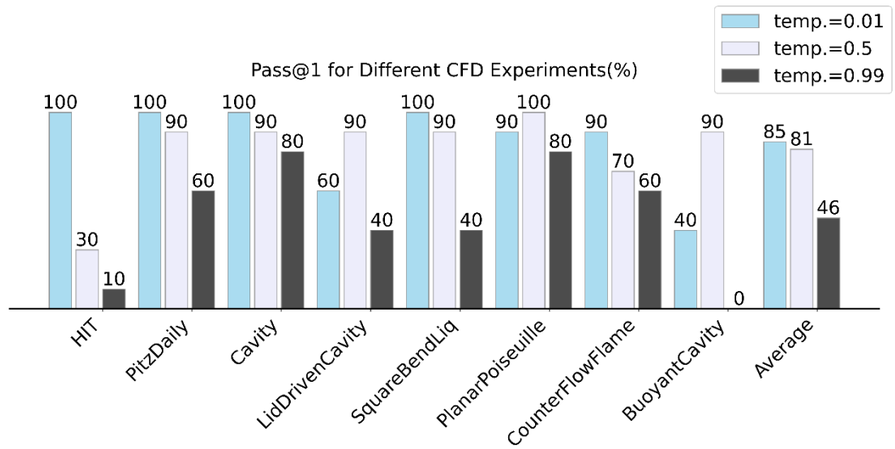


图5 三种不同温度（0.01、0.5、

0.99）下LLM的平均通过率@1，其中'temp.'表示'温度'。

图5展示了八个测试用例在三种不同温度下的平均通过率@1

(0.01, 0.5, 0.99). We found that for most case, lower temperature leads to higher pass@1 due to the more stable and accurate generated inputs. However, for a few cases, such as LidDrivenCavity and BuoyantCavity, a middle temperature (temperature=0.5) achieved better pass@1 than the low temperature (temperature=0.01). But high temperature (temperature = 0.99) results in low pass@1 in all the test dataset, which means that LLM with high temperature is not suitable for CFD tasks solving.

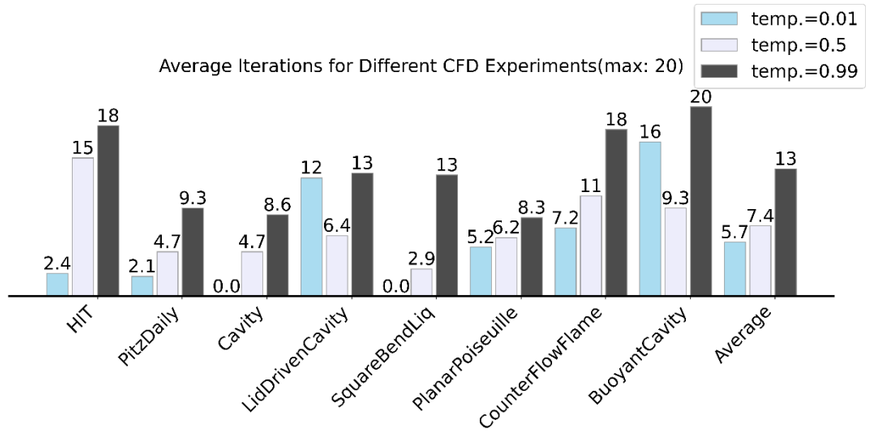


Figure 6 Average iterations for different CFD experiments with 3 different temperatures of LLM

Figure 6 illustrates the average iterations for eight test cases under three different temperatures (0.01, 0.5, 0.99). Each iteration requires running the InputWriter, Runner, and Reviewer. Iterations is a cost metric for MetaOpenFOAM, which correlates positively with tokens and expenses. For most cases, lower temperature leads to lower iterations due to the more stable and accurate generated inputs. However, for a few cases like LidDrivenCavity and BuoyantCavity, very low randomness (temperature=0.01) led to overly sharp probability distributions, resulting in repeated errors during iterations. In contrast, moderate randomness (temperature=0.5) helped overcome the fixed patterns of errors produced at low temperatures, reducing the number of iterations and thereby lowering tokens and expenses.

Therefore, a dynamic temperature model might be beneficial. It could dynamically adjust from low to middle temperatures when low-temperature generation is ineffective, improving pass rates and reducing costs in MetaOpenFOAM.

(0.01, 0.5, 0.99).我们发现，在大多数情况下，较低的temperature会导致较高的pass@1，这是由于生成的input更加稳定和准确。然而，在少数情况下，如LidDrivenCavity和BuoyantCavity，中等temperature（temperature=0.5）比低

temperature（temperature=0.01）获得了更好的pass@1。但高temperature（temperature = 0.99）在所有测试数据集中导致低pass@1

，这意味着具有高temperature的LLM不适合用于CFD任务求解。

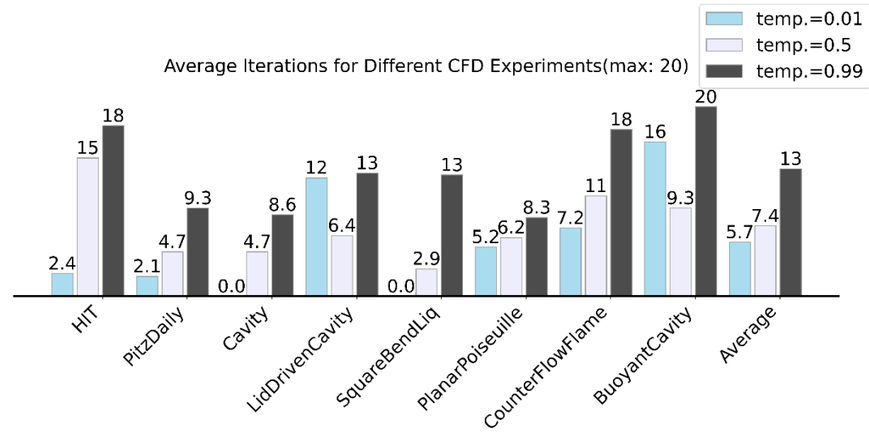


图6 采用3种不同LLM温度进行CFD实验的平均迭代次数

图6展示了在三种不同温度（0.01、0.5、0.99）下八个测试用例的平均迭代次数。每次迭代都需要运行输入写入器、运行器和评审员。迭代次数是MetaOpenFOAM的成本指标，与令牌和费用呈正相关。大多数情况下，较低温度会因生成更稳定准确的输入而减少迭代次数。然而，对于盖驱动腔和浮力腔等少数案例，极低的随机性（温度0.01）导致概率分布过于尖锐，从而在迭代过程中产生重复错误。相比之下，适度随机性（温度0.5）有助于克服低温产生的固定错误模式，减少迭代次数，进而降低令牌消耗和费用。

因此，采用动态温度模型可能更为有利。该模型能够根据{v3}动态调整从U到p的温度参数。在低温生成无效时，采用中低温可提高通过率。

并降低MetaOpenFOAM的成本。

4.3 Generalizability Study

In this section, we will qualitatively analyze the performance of MetaOpenFOAM in some special situations such as the ones requiring modification of key parameters, matching with less similar cases, or requiring human participation to improve performance.

A. Key data identification and modification

One of the most important capabilities in a natural language based CFD simulation framework is the ability to identify changes to key data in the user requirement and modify those changes when writing the input file. This capability is the foundation of CFD numerical optimization based on natural language and is a key differentiator from executability 3 to 4 in evaluation metrics.

Key data can be divided into two categories. One is can be directly corrected in the input, such as grid size, model parameters, and control parameters; the other category cannot be directly modified in the input, such as the Reynolds number.

For key data that can be directly corrected in the input, MetaOpenFOAM can identify and modify the corresponding parameters in the input. Here lists the dataset where the key data changed:

① HIT: do a DNS simulation of incompressible forcing homogeneous isotropic turbulence with Grid 20^3 using dnsFoam. [32->20]

② PitzDaily: do a LES simulation of incompressible pitzDaily flow using pisoFoam with inlet velocity = 8 m/s. [5->8]

③ Cavity: do a 2D RANS simulation of incompressible cavity flow using pisoFoam, with RANS model: KEpsilon, grid 15*15*1. [RNGkEpsilon-> KEpsilon]

④ LidDrivenCavity: do an incompressible lid driven cavity flow simulation with the top wall moves in the x direction at a speed of 2 m/s while the other 3 are stationary. [1->2]

⑤ SquareBendLiq: do a compressible simulation of squareBendLiq of using rhoSimpleFoam with endTime = 1000, deltaT = 1, and writeInterval = 100. [100->1000, 10->100]

⑥ PlanarPoiseuille: do a laminar simulation of incompressible planar Poiseuille flow of a Newtonian fluid with grid 1*20*1, initially at rest, constant pressure gradient applied from time zero. [non-Newtonian -> Newtonian]

⑦ CounterFlowFlame: do a 2D laminar simulation of counterflow flame using reactingFoam in combustion with grid 50*20*1. [50*20*1->40*20*1]

⑧ BuoyantCavity: do a RANS simulation of buoyantCavity using buoyantFoam, which investigates natural convection in a heat cavity with a temperature difference of 15K is maintained between the hot and cold; the remaining patches are treated as adiabatic. [20->15]

Table 4 presents the performance test results of MetaOpenFOAM on the original dataset (dataset1) and the modified dataset (dataset2). It was found that the two datasets show similar performance in terms of Executability and pass@1, indicating that MetaOpenFOAM can identify

4.3 泛化性研究

在本节中，我们将定性分析MetaOpenFOAM在一些场景中的性能

特殊情况下，例如需要修改关键参数、匹配相似度较低的案例，或需要人工参与以提升性能时。

类似案例，或需要人工参与以提升性能。

A. 关键数据识别与修改

基于自然语言的CFD模拟框架最重要的能力之一，是能够识别用户需求中关键数据的变更，并在编写输入文件时修改这些变更。这一能力是自然语言驱动的CFD数值优化的基础，也是评估指标中可执行性3到4的关键差异点。

关键数据可分为两类：一类可直接在输入中修正，如网格尺寸、模型参数和控制参数；另一类无法直接在输入中修改，如雷诺数。

对于可直接在输入中修正的关键数据，MetaOpenFOAM能够识别并修改输入中的相应参数。以下列出关键数据变更的数据集：

1 HIT：使用dnsFoam对不可压缩强迫均匀各向同性湍流进行直接数值模拟，网格20^3 。[32->20]

2 PitzDaily: 使用pisoFoam对不可压缩pitzDaily流动进行大涡模拟，入口速度= 8 m/s。[5->8]

3 Cavity: 使用pisoFoam对不可压缩空腔流动进行二维雷诺平均模拟，雷诺平均模型：KEpsilon，网格15*15*1。[RNGkEpsilon-> KEpsilon]

4 LidDrivenCavity：进行不可压缩顶盖驱动空腔流动模拟，顶壁以2 m/s速度沿x方向移动，其余三壁静止。[1->2]

5 SquareBendLiq: 使用rhoSimpleFoam对方形弯管液体进行可压缩模拟，结束时间= 1000，时间步长= 1，写入间隔= 100。[100->1000, 10->100]

6 平面泊肃叶: 对静止状态下、从零时刻开始施加恒定压力梯度的牛顿流体不可压缩平面泊肃叶流进行层流模拟，使用网格1*20*1。[非牛顿-> 牛顿]

7 逆流火焰: 使用反应泡沫在燃烧中对逆流火焰进行二维层流模拟，采用网格50*20*1。[50*20*1->40*20*1]

8 浮力腔体: 使用浮力泡沫对浮力腔体进行RANS模拟，研究热腔内维持15K温差的热和冷之间的自然对流；其余壁面按绝热处理。[20->15]

表4展示了MetaOpenFOAM在原始数据集（数据集1）和修改后的数据集（数据集2）上的性能测试结果。研究发现，这两个数据集在可执行性和通过率@1方面表现出相似的性能，表明MetaOpenFOAM能够识别

and modify key parameters in the input for key data that can be directly modified, achieving similar performance as the original dataset.

Figure 7 displays the executability of each case before and after modification. The differences between dataset1 and dataset2 are minimal for most cases, except for SquareBendLiq, which shows a significant difference. This discrepancy arises because the user requirements for SquareBendLiq demand a longer simulation time. During this extended simulation time, the case encountered convergence issues, resulting in an Executability of 2 and a pass@1 of 0.

Table 4. Performance of MetaOpenFOAM in two different datasets (dataset1: original dataset, dataset2: modified dataset)

Statistical Index	**Dataset1**	**Dataset2**
(A) Executability	3.6	3.7
(B) Cost: Running time (s)	271	294
(B) Cost: Iteration (max: 20)	5.7	7.2
(B) Cost: Token Usage	44045	48416
(C) Code Statistic: Input Files	13.7	13.5
(C) Code Statistic: Lines per Input File	50	42
(C) Code Statistic: Total lines of Input Files	760	581
(D) Productivity	67.1	79.2
(E) pass@1 (%)	85	85

However, for some data that cannot be directly modified in the input, such as Reynolds number, the current version of MetaOpenFOAM cannot directly recognize such data that require secondary calculation.

并修改输入中的关键参数，针对可直接修改的关键数据，实现相似

性能与原始数据集相同。

图7展示了修改前后各案例的可执行性。除方形弯管液体外，数据集1与数据集2在大多数案例中差异极小。方形弯管液体出现显著差异的原因是用户需求要求更长的模拟时间。在延长的模拟时间内，该案例遇到收敛问题，导致可执行性为2且通过率@1为0。

表4. MetaOpenFOAM在两个不同数据集中的性能（数据集1：原始数据集，

数据集2：修改后的数据集）

统计指标	数据集1	数据集2
(A) 可执行性	3.6	3.7
(B) 成本：运行时间（秒）	271	294
(B) 成本：迭代（最大：20）	5.7	7.2
(B) 成本：令牌使用量	44045	48416
(C) 代码统计：输入文件	13.7	13.5
(C) 代码统计：每个输入文件的行数	50	42
(C) 代码统计：输入文件总行数	760	581
(D) 生产率	67.1	79.2
(E) pass@1 (%)	85	85

然而，对于某些无法在输入中直接修改的数据，例如雷诺数，

当前版本的MetaOpenFOAM无法直接识别这类需要二次处理的数据

计算。

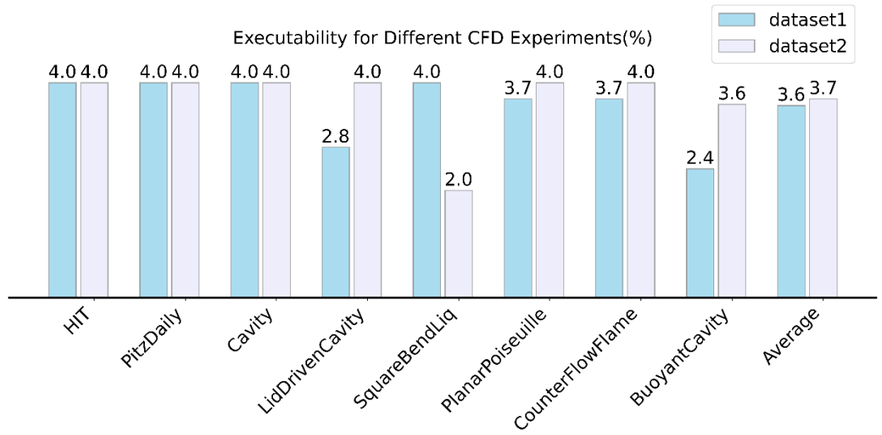


Figure 7 Average executability for different CFD experiments in two different datasets (dataset1: original dataset, dataset2: modified dataset)

For example, in the HIT case, if the user requests a simulation with Reynolds number at 20, MetaOpenFOAM cannot produce a simulation that meets user requirement. This issue is not insurmountable. In the future, by introducing additional agents and a comprehensive library of functions to convert key parameters to input parameters, MetaOpenFOAM could handle a broader range of CFD simulation tasks.

B. Failure similarity match

Similarity match, as a part of RAG technology is performed during the "Find similar cases" action by the Architect. However, it may fail to match the most similar case due to the inaccurate user requirement, or the inadequate database, which may not contain similar cases. In such scenarios, both pass@1 and executability are significantly reduced. However, even in these cases, using RAG technology still yields better results than not using it.

For example, in the lidDrivenCavity case, the most similar case found in 10 tests was sloshingCylinder, which clearly does not have a high similarity match. Despite this, shown as in Table 1 and Table 8, the pass@1 60% and executability 2.8 with RAG technology were still much better than without it, whose pass@1 is 0% and executability is 1.0.

C. Human participation

While MetaOpenFOAM aims to perform successful simulations using natural language inputs alone, for cases with complex geometries and boundary conditions, natural language inputs alone are insufficient. Therefore, human participation is needed to enhance the performance of

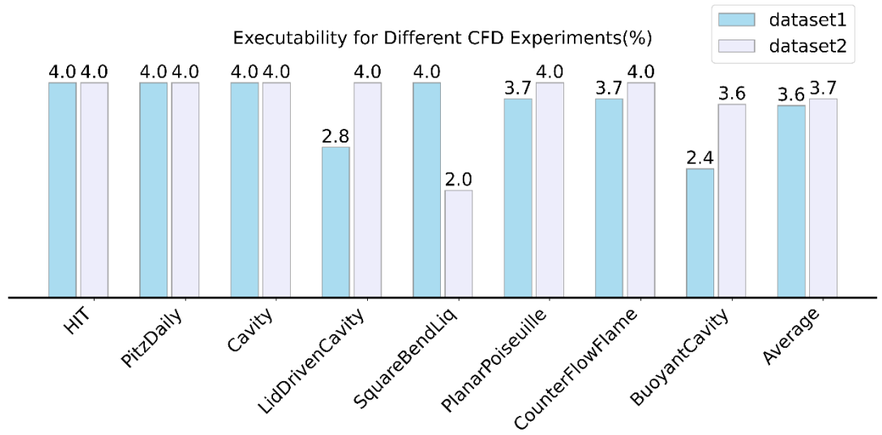


图7 两个不同数据集（数据集1:

原始数据集，数据集2：修改后的数据集）中不同CFD实验的
平均可执行性

例如，在HIT案例中，若用户请求雷诺数为20的模拟，MetaOpenFOAM当前无法生成符合用户要求的模拟结果。这一问题并非不可克服。未来通过引入更多智能体及完备的函数库将关键参数转化为输入参数，MetaOpenFOAM将能处理更广泛的计算流体动力学模拟任务。

B. 失败相似性匹配

作为RAG技术的一部分，相似性匹配由架构师在“查找相似案例”动作中执行。然而，由于用户要求不准确或数据库不足（可能未包含相似案例），它可能无法匹配到最相似的案例。在此类场景中，

通过率@1和可执行性均显著降低。但即使在这些情况下，使用RAG技术仍比不使用它获得更好的结果。

例如，在lidDrivenCavity案例中，10次测试中找到的最相似案例是sloshingCylinder，显然其相似性匹配度不高。尽管如此，如表1和表8所示，使用RAG技术的通过率@1为60%、可执行性为2.8，仍远优于不使用时的0%通过率@1和1.0可执行性。

C. 人工参与

尽管MetaOpenFOAM旨在仅通过自然语言输入完成成功的模拟，但对于具有复杂几何结构和边界条件的案例，仅靠自然语言输入是不够的。因此需要人工参与来提升MetaOpenFOAM的性能。

MetaOpenFOAM. Interfaces have been included in MetaOpenFOAM to handle issues related to complex geometries and boundary conditions.

These interfaces allow for human intervention to refine the simulation setup, ensuring that MetaOpenFOAM can tackle more sophisticated CFD tasks that are beyond the current capabilities of fully automated processes. This combination of automated and human-assisted approaches aims to strike a balance between efficiency and accuracy in CFD simulations.

5. Conclusion

For the first time, an LLM-based multi-agent framework for CFD has been established, incorporating the construction of a multi-agent system based on MetaGPT and RAG technology based on Langchain. Specifically, the multi-agent system consists of four roles: Architect, InputWriter, Runner, and Reviewer. These roles work together by first decomposing the CFD simulation task into a series of input file generation subtasks (Architect), then generating the corresponding input files based on the subtasks (InputWriter), running the CFD simulation (Runner), and finally providing feedback to the InputWriter by checking the simulation results (Reviewer) until the results meet expectations or the maximum number of iterations is reached. RAG technology aids this process by converting OpenFOAM tutorials into a database, matching the most similar cases from the database whenever the LLM needs access, thereby assisting the LLM in answering queries.

To test the performance of MetaOpenFOAM in CFD simulations, a benchmark for natural language-based CFD solvers was proposed, consisting of eight CFD simulation tasks derived from OpenFOAM tutorials. Tests on the benchmark have shown that MetaOpenFOAM achieved a high pass@1 rate of 85%, with each test case costing only $0.22 on average. This demonstrates MetaOpenFOAM's ability to automate CFD simulations using only natural language input and iteratively correct errors to achieve the desired simulation at a low cost.

An ablation study was conducted to verify the roles of each component in the multi-agent system and the RAG technology. The results showed that removing the Reviewer, Action review architecture, and RAG resulted in pass@1 rates of 27.5%, 70%, and 0%, respectively, all lower than the original framework's 85%. This demonstrates the necessity of reviewing results and RAG technology. A sensitivity study on the temperature parameter in the LLM showed that compared to middle/high temperature (0.5/0.99) pass@1 rates of 83% and 48%, the LLM based on low

MetaOpenFOAM已内置相关接口以处理涉及

复杂几何结构和边界条件。

这些接口允许人工干预以优化模拟设置，确保MetaOpenFOAM能够处理当前全自动化流程无法胜任的更复杂CFD任务。这种自动化与人工辅助相结合的方式旨在CFD仿真中实现效率与精度的平衡。

5. 结论

首次建立了基于LLM的CFD多智能体框架，整合了基于MetaGPT的多智能体系统构建和基于Langchain的RAG技术。具体而言，该多智能体系统包含四个角色：架构师、输入写入器、运行器和评审员。这些角色协同工作，首先将CFD模拟任务分解为一系列输入文件生成子任务（架构师），随后基于子任务生成对应输入文件（输入写入器），运行CFD模拟（运行器），

最终通过检查模拟结果向输入写入器提供反馈（评审员），直至结果符合预期或达到最大迭代次数。RAG技术通过将OpenFOAM教程转化为数据库来辅助此过程，每当LLM需要访问时从数据库中匹配最相似案例，从而协助LLM回答查询。

为测试MetaOpenFOAM在CFD仿真中的性能，我们提出了一个基于自然语言的CFD求解器基准测试，该测试包含八个源自OpenFOAM教程的CFD模拟任务。基准测试结果表明，MetaOpenFOAM实现了85%的高通过@1率，且每个测试用例的平均成本仅为0.22美元。这证明了MetaOpenFOAM仅通过自然语言输入即可自动化CFD模拟，并通过迭代修正错误以低成本实现预期仿真的能力。

为验证多智能体系统中各组件及RAG技术的作用，我们进行了消融研究。结果显示，移除评审员、动作审查架构和RAG技术后，通过@1率分别为27.5%、70%和0%，均低于原框架85%的水平。这证明了结果评审机制与RAG技术的必要性。针对LLM温度参数的敏感性研究表明：相较于中/高温（0.5/0.99）83%与48%的通过@1率，基于低温

temperature (0.01) has a higher pass@1 rate of 85% on average. This suggests that conservative and coherent generated text leads to more stable and accurate results. However, the middle temperature (0.5) also showed good performance in specific cases.

Additionally, MetaOpenFOAM maintained an 85% pass@1 rate even after modifying keywords in the prompts, indicating its ability to identify and modify key parameters in the input corresponding to the keywords. The performance on failure similarity matches also showed that MetaOpenFOAM has good simulation capability even when the most similar case is not matched. Finally, incorporating human participation can enhance the performance of MetaOpenFOAM in handling issues related to complex geometries and boundary conditions. These three aspects demonstrate that MetaOpenFOAM possesses a certain degree of generalization ability.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. 52025062 and 52106166). The authors also acknowledge High-Performance Computing Centre at Tsinghua University for providing computational resource.

During the preparation of this work the author(s) used ChatGPT in order to improve language and readability. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the content of the publication.

References

[1] J. Blazek, Computational fluid dynamics: principles and applications, Butterworth-Heinemann, 2015.
[2] H. Jasak, A. Jemcov, Z. Tukovic, OpenFOAM: A C++ library for complex physics simulations, International workshop on coupled methods in numerical dynamics, Dubrovnik, Croatia), 2007, pp. 1-20.
[3] J. An, H. Wang, B. Liu, K.H. Luo, F. Qin, G.Q. He, A deep learning framework for hydrogen-fueled turbulent combustion simulation, International Journal of Hydrogen Energy, 45 (2020) 17992-18000.
[4] R. Mao, M. Lin, Y. Zhang, T. Zhang, Z.-Q.J. Xu, Z.X. Chen, DeepFlame: A deep learning empowered open-source platform for reacting flow simulations, Computer Physics Communications, 291 (2023) 108842.
[5] Y. Wang, P. Chatterjee, J.L. de Ris, Large eddy simulation of fire plumes, Proceedings of the Combustion Institute, 33 (2011) 2473-2480.
[6] J. Wei, J. An, N. Wang, J. Zhang, Z. Ren, Velocity nonuniformity and wall heat loss coupling effect on supersonic mixing layer flames, Aerospace Science and Technology, 141 (2023) 108545.
[7] U. Manual, ANSYS FLUENT 12.0, Theory Guide, 67 (2009).
[8] C. Multiphysics, Introduction to comsol multiphysics®, COMSOL Multiphysics, Burlington, MA, accessed Feb, 9 (1998) 32.

（0.01）的LLM平均通过@1率达85%，表明保守且连贯的生成文本能带来更稳定且准确的结果。但中温（0.5）在特定场景下也展现出良好性能。

此外，即使修改提示中的关键词，MetaOpenFOAM仍保持85%的通过@1率，表明其具备识别并修改与关键词对应的输入关键参数的能力。在失败相似性匹配方面的性能也显示，即使未匹配到最相似案例，MetaOpenFOAM仍具有良好的模拟能力。最后，引入人工参与可提升MetaOpenFOAM处理复杂几何结构和边界条件相关问题的性能。这三个方面证明MetaOpenFOAM具有一定程度的泛化能力。

致谢

本研究得到国家自然科学基金（项目编号：52025062和52106166）资助。作者同时感谢清华大学高性能计算中心提供的计算资源支持。

在准备本作品期间，作者使用ChatGPT以提升语言表达和可读性。使用该工具/服务后，作者对内容进行了必要的审阅和编辑，并对出版物的内容承担全部责任。

参考文献

[1] J. Blazek, Computational fluid dynamics: principles and applications, Butterworth-Heinemann, 2015.
[2] H. Jasak, A. Jemcov, Z. Tukovic, OpenFOAM: A C++ library for complex physics simulations, International workshop on coupled methods in numerical dynamics, Dubrovnik, Croatia), 2007, pp. 1-20.
[3] J. An, H. Wang, B. Liu, K.H. Luo, F. Qin, G.Q. He, A deep learning framework for hydrogen-fueled turbulent combustion simulation, International Journal of Hydrogen Energy, 45 (2020) 17992-18000.
[4] R. Mao, M. Lin, Y. Zhang, T. Zhang, Z.-Q.J. Xu, Z.X. Chen, DeepFlame: A deep learning empowered open-source platform for reacting flow simulations, Computer Physics Communications, 291 (2023) 108842.
[5] Y. Wang, P. Chatterjee, J.L. de Ris, Large eddy simulation of fire plumes, Proceedings of the Combustion Institute, 33 (2011) 2473-2480.
[6] J. Wei, J. An, N. Wang, J. Zhang, Z. Ren, Velocity nonuniformity and wall heat loss coupling effect on supersonic mixing layer flames, Aerospace Science and Technology, 141 (2023) 108545.
[7] U. Manual, ANSYS FLUENT 12.0, Theory Guide, 67 (2009).
[8] C. Multiphysics, Introduction to comsol multiphysics®, COMSOL Multiphysics, Burlington, MA, accessed Feb, 9 (1998) 32.

[9] E. Akata, L. Schulz, J. Coda-Forno, S.J. Oh, M. Bethge, E. Schulz, Playing repeated games with large language models, arXiv preprint arXiv:2305.16867, (2023).
[10] Y. Du, S. Li, A. Torralba, J.B. Tenenbaum, I. Mordatch, Improving factuality and reasoning in language models through multiagent debate, arXiv preprint arXiv:2305.14325, (2023).
[11] S. Hong, X. Zheng, J. Chen, Y. Cheng, J. Wang, C. Zhang, Z. Wang, S.K.S. Yau, Z. Lin, L. Zhou, Metagpt: Meta programming for multi-agent collaborative framework, arXiv preprint arXiv:2308.00352, (2023).
[12] L. Wang, C. Ma, X. Feng, Z. Zhang, H. Yang, J. Zhang, Z. Chen, J. Tang, X. Chen, Y. Lin, A survey on large language model based autonomous agents, Frontiers of Computer Science, 18 (2024) 186345.
[13] Z. Wang, S. Mao, W. Wu, T. Ge, F. Wei, H. Ji, Unleashing the emergent cognitive synergy in large language models: A task-solving agent through multi-persona self-collaboration, arXiv preprint arXiv:2307.05300, (2023).
[14] M. Zhuge, H. Liu, F. Faccio, D.R. Ashley, R. Csordás, A. Gopalakrishnan, A. Hamdi, H.A.A.K. Hammoud, V. Herrmann, K. Irie, Mindstorms in natural language-based societies of mind, arXiv preprint arXiv:2305.17066, (2023).
[15] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F.L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, Gpt-4 technical report, arXiv preprint arXiv:2303.08774, (2023).
[16] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, Llama 2: Open foundation and fine-tuned chat models, arXiv preprint arXiv:2307.09288, (2023).
[17] T. GLM, A. Zeng, B. Xu, B. Wang, C. Zhang, D. Yin, D. Rojas, G. Feng, H. Zhao, H. Lai, ChatGLM: A Family of Large Language Models from GLM-130B to GLM-4 All Tools, arXiv preprint arXiv:2406.12793, (2024).
[18] T. Cai, X. Wang, T. Ma, X. Chen, D. Zhou, Large language models as tool makers, arXiv preprint arXiv:2305.17126, (2023).
[19] G. Li, H. Hammoud, H. Itani, D. Khizbullin, B. Ghanem, Camel: Communicative agents for" mind" exploration of large language model society, Advances in Neural Information Processing Systems, 36 (2023) 51991-52008.
[20] J.S. Park, J. O'Brien, C.J. Cai, M.R. Morris, P. Liang, M.S. Bernstein, Generative agents: Interactive simulacra of human behavior, Proceedings of the 36th annual acm symposium on user interface software and technology, 2023, pp. 1-22.
[21] H. Chase, LangChain, https://github.com/ langchain-ai/langchain, 2022.
[22] M. Douze, A. Guzhva, C. Deng, J. Johnson, G. Szilvasy, P.-E. Mazaré, M. Lomeli, L. Hosseini, H. Jégou, The faiss library, arXiv preprint arXiv:2401.08281, (2024).
[23] M. Chen, J. Tworek, H. Jun, Q. Yuan, H.P.D.O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, Evaluating large language models trained on code, arXiv preprint arXiv:2107.03374, (2021).
[24] U. Ayachit, The paraview guide: a parallel visualization application, Kitware, Inc.2015.

Appendix A. Prompts of MetaOpenFOAM

In this Appendix, we will present some key action prompts for the four roles: Architect,

[9] E. Akata, L. Schulz, J. Coda-Forno, S.J. Oh, M. Bethge, E. Schulz, Playing repeated games with large language models, arXiv preprint arXiv:2305.16867, (2023).
[10] Y. Du, S. Li, A. Torralba, J.B. Tenenbaum, I. Mordatch, Improving factuality and reasoning in language models through multiagent debate, arXiv preprint arXiv:2305.14325, (2023).
[11] S. Hong, X. Zheng, J. Chen, Y. Cheng, J. Wang, C. Zhang, Z. Wang, S.K.S. Yau, Z. Lin, L. Zhou, Metagpt: Meta programming for multi-agent collaborative framework, arXiv preprint arXiv:2308.00352, (2023).
[12] L. Wang, C. Ma, X. Feng, Z. Zhang, H. Yang, J. Zhang, Z. Chen, J. Tang, X. Chen, Y. Lin, A survey on large language model based autonomous agents, Frontiers of Computer Science, 18 (2024) 186345.
[13] Z. Wang, S. Mao, W. Wu, T. Ge, F. Wei, H. Ji, Unleashing the emergent cognitive synergy in large language models: A task-solving agent through multi-persona self-collaboration, arXiv preprint arXiv:2307.05300, (2023).
[14] M. Zhuge, H. Liu, F. Faccio, D.R. Ashley, R. Csordás, A. Gopalakrishnan, A. Hamdi, H.A.A.K. Hammoud, V. Herrmann, K. Irie, Mindstorms in natural language-based societies of mind, arXiv preprint arXiv:2305.17066, (2023).
[15] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F.L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, Gpt-4 technical report, arXiv preprint arXiv:2303.08774, (2023).
[16] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, Llama 2: Open foundation and fine-tuned chat models, arXiv preprint arXiv:2307.09288, (2023).
[17] T. GLM, A. Zeng, B. Xu, B. Wang, C. Zhang, D. Yin, D. Rojas, G. Feng, H. Zhao, H. Lai, ChatGLM: A Family of Large Language Models from GLM-130B to GLM-4 All Tools, arXiv preprint arXiv:2406.12793, (2024).
[18] T. Cai, X. Wang, T. Ma, X. Chen, D. Zhou, Large language models as tool makers, arXiv preprint arXiv:2305.17126, (2023).
[19] G. Li, H. Hammoud, H. Itani, D. Khizbullin, B. Ghanem, Camel: Communicative agents for" mind" exploration of large language model society, Advances in Neural Information Processing Systems, 36 (2023) 51991-52008.
[20] J.S. Park, J. O'Brien, C.J. Cai, M.R. Morris, P. Liang, M.S. Bernstein, Generative agents: Interactive simulacra of human behavior, Proceedings of the 36th annual acm symposium on user interface software and technology, 2023, pp. 1-22.
[21] H. Chase, LangChain, https://github.com/ langchain-ai/langchain, 2022.
[22] M. Douze, A. Guzhva, C. Deng, J. Johnson, G. Szilvasy, P.-E. Mazaré, M. Lomeli, L. Hosseini, H. Jégou, The faiss library, arXiv preprint arXiv:2401.08281, (2024).
[23] M. Chen, J. Tworek, H. Jun, Q. Yuan, H.P.D.O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, Evaluating large language models trained on code, arXiv preprint arXiv:2107.03374, (2021).
[24] U. Ayachit, The paraview guide: a parallel visualization application, Kitware, Inc.2015.

附录A. MetaOpenFOAM的提示

在本附录中，我们将展示针对四个角色的关键动作提示：架构师、

InputWriter, Runner, and Reviewer. For the complete prompts, please refer directly to the source code: https://github.com/Terry-cyx/MetaOpenFOAM

Action: Create input architecture in Role: Architect

```
    User requirement:
    {requirement}
    Your task is to generate the OpenFOAM input foamfiles list following
file structure of OpenFOAM cases to meet the user requirements.
    Here is a OpenFOAM case similar to the user requirements
    The following is a case of OpenFOAM, which is similar to the user's
requirements:
    {tutorial}
    Please take this case as a reference. generate the OpenFOAM input
foamfiles list following file structure of OpenFOAM cases to meet the user
requirements.
    You should splits foamfiles list into several subtasks, and one subtask
corresponds to one input foamfile
    Return ```splits into number_of_subtasks subtasks:
    subtask1: to Write a OpenFoam specific_file_name foamfile in
specific_folder_name folder that could be used to meet user
requirement:{requirement}.
    subtask2: to Write a OpenFoam specific_file_name foamfile in
specific_folder_name folder that could be used to meet user
requirement:{requirement}.
    ...

    ``` with NO other texts,
    your subtasks:
```

Action: Write input file in Role: InputWriter

```
    Your task is {requirement}.
    The similar foamfile is provided as follows:
    {tutorial_file}
    Please take this foamfile as a reference, which may help you to finish
your task.
    According to your task, return ```your_code_here ``` with NO other
texts,
    your code:
```

Action: Write allrun file in Role: Runner

```
       Your task is to write linux execution command allrun file to meet
the user requirement: {requirement}.
       The input file list is {file_list}.
       Here is a OpenFOAM allrun file similar to the user requirements:
```

输入写入器、运行器及评审员。完整提示请直接查阅源文件

代码: https://github.com/Terry-cyx/MetaOpenFOAM

动作：在角色: 架构师中创建输入架构

```
    用户需求：
    {requirement}
    您的任务是按照OpenFOAM案例的文件结构生成OpenFOAM输入foam文件列表
以满足用户需求。
    这里有一个符合用户需求的OpenFOAM案例
    以下是一个OpenFOAM案例，与用户的
需求相似：
    {tutorial}
    请将此案例作为参考，生成OpenFOAM输入
foam文件列表，遵循OpenFOAM案例的文件结构以满足用户
要求。
    您应将foam文件列表拆分为若干子任务，每个子任务
对应一个输入foam文件
    返回    ```拆分为数量_个_子任务：
    子任务1：编写一个OpenFoam专用_文件_名称的foam文件
特定_文件夹_名称文件夹可用于满足用户
要求:{requirement}。
    子任务2：编写一个OpenFOAM特定_文件_名称的Foam文件
特定_文件夹_名称文件夹可用于满足用户
要求:{requirement}。
    ...

    ``` 无其他文本          ，
    你的子任务：
```

动作：以角色：输入写入器编写输入文件

```
    你的任务是{requirement}。
    类似的Foam文件提供如下：
    {tutorial_file}
    请将此Foam文件作为参考，可能有助于你完成
你的任务。
    根据你的任务，返回              ```你的_代码_在此  ``` 不要包含其他
文本 ，
    你的代码：
```

动作：在角色：运行器中编写Allrun文件

```
       您的任务是编写Linux执行命令Allrun文件以满足
用户需求：{requirement}。
       输入文件列表是 {file_list}。
       以下是符合用户需求的OpenFOAM Allrun文件示例：
```

```
        {tutorial}
        Please take this file as a reference.
        The possible command list is
        {commands}
        The possible run list is
        {runlists}
        Make sure the written linux execution command are coming from the
above two lists.
        According to your task, return ```your_allrun_file_here ``` with NO
other texts
```

Action: Review file architecture in Role: Reviewer

```
    {command} has been executed in OpenFOAM10, and got the following error:
    {error}
    The corresponding input file list is:
    {file_list} in folder {folder_list}
    Please analyze which files the error may be related to, and return the
related files and the corresponding folders as follows:
    ###file_name1, file_name2, ...### in ```file_folder1,
file_folder2, ...``` with NO other texts.
    where file_name1, file_name2, ..., come from {file_list}
    and file_folder1, file_folder2, ..., come from {folder_list}
```

Action: Review file context in Role: Reviewer

```
    to rewrite a OpenFoam {file_name} foamfile in {file_folder} folder that
could solve the error:
    ###ERROR BEGIN:
    {error}
    ERROR END.###
    Note that {file_list} in {folder_list} folder was found to be
associated with the error, and you need to rewrite {file_name} first,
taking into account how these files affect each other.
    the original {file_list} in {folder_list} folder encounter the error
when {command} has been executed in OpenFOAM10,
    {related_files}
    Note that you need to return the entire modified file, never return a
single modified fragment, because I want to save and run the file directly,
making sure there are no other characters
    According to your task, return ```your_code_here ``` with NO other
texts,
    your code:
```

Appendix B. OpenFOAM database

The OpenFOAM database is derived from the tutorials in the main OpenFOAM directory and

```
        {tutorial}
        请将此文件作为参考。
        可能的命令列表是
        {commands}
        可能的运行列表是
        {runlists}
        确保所写的Linux执行命令来自
上述两个列表。
        根据你的任务，返回 ```你的_allrun_文件_在此 ``` 不要包含
其他文本
```

操作：以审阅者角色审查文件架构

```
    {command} 已在OpenFOAM10中执行，并收到以下错误：
    {error}
    对应的输入文件列表为：
    {file_list} 位于文件夹{folder_list}
    请分析错误可能与哪些文件相关，并返回
相关文件及对应文件夹如下：
    ###file _name1 ，file_名称2 ，...### in ```file_文件夹1 ，
file_文件夹2 ，...``` 没有其他文本。
    其中文件_名称1, 文件_名称2, ..., 来自 {file_list}
    且文件_文件夹1, 文件_文件夹2, ..., 来自{folder_list}
```

动作: 在角色: 评审员中审查文件上下文

```
    重写一个OpenFoam{file_name} foam文件于 {file_folder} 文件夹中
可解决以下错误：
    ###错误开始：
    {error}
    错误结束。###
    注意 {file_list} 中的 {folder_list} 文件夹被发现存在
与该错误关联，您需要先重写 {file_name} ，
需考虑这些文件如何相互影响。
    the original{文件_列表} 在 {文件夹_列表} 文件夹遇到错误
当命令在OpenFOAM10中执行后，
    {related_files}
    请注意您需要返回整个修改后的文件，切勿仅返回
单个修改片段，因为我需要直接保存并运行该文件，
确保没有其他字符
    根据你的任务要求返回 ```你的_代码_在这里 ``` 不要有其他
文本 ，
    你的代码：
```

附录B. OpenFOAM数据库

OpenFOAM数据库源自主OpenFOAM目录中的教程，且

is primarily divided into three sub-databases: foamfile architecture, foamfile context, and allrun files.

Different sub-databases are used based on the specific action being performed. Below are some excerpts:

Sub-database foamfile architecture:

```
###case begin:
case name: pitzDaily
case domain: compressible
case category: LES
case solver: rhoPimpleFoam
case input name:['fvSolution', 'fvSchemes', 'fvConstraints', 'controlDict',
'momentumTransport', 'physicalProperties', 'nut', 'k', 'T', 'p', 'alphat',
'U', 'muTilda']
corresponding input folder:{'fvSolution': 'system', 'fvSchemes': 'system',
'fvConstraints': 'system', 'controlDict': 'system', 'momentumTransport':
'constant', 'physicalProperties': 'constant', 'nut': '0', 'k': '0', 'T':
'0', 'p': '0', 'alphat': '0', 'U': '0', 'muTilda': '0'}
case end.###
```

Sub-database foamfile context:

```
/*--------------------------------*- C++ -*----------------------------------
--*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://OpenFOAM.org
    \\  /    A nd           | Version:  10
     \\/     M anipulation  |
\*---------------------------------------------------------------------------
--*/
FoamFile
{
    format      ascii;
    class       dictionary;
    object      fvConstraints;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* //
limitp
{
    type       limitPressure;
    minFactor  0.5;
    maxFactor  2;
}
```

主要划分为三个子数据库：foamfile架构、foamfile上下文和allrun文件。

根据执行的具体动作不同，会使用不同的子数据库。以下是一些摘录：

子数据库 foamfile架构：

```
###案例开始:
案例名称：pitzDaily
案例领域：可压缩
案例类别：LES
案例求解器: rhoPimpleFoam
案例输入名称:['fvSolution', '有限体积格式', '有限体积约束', '控制字典',
'动量传输', '物理属性', '螺母', 'k', 'T', 'p', '阿尔法特',
'U', 'μ波浪号']
对应输入文件夹:{'fvSolution': '系统', '有限体积格式': '系统',
'有限体积约束': '系统', '控制字典': '系统', '动量传输':
'常数', '物理属性': '常数', '螺母': '0', 'k': '0', 'T':
'0', 'p': '0', 'alphat': '0', 'U': '0', 'μ波浪号': '0'}
case end.###
```

子数据库 foamfile上下文：

```
/*--------------------------------*-              C++ -*----------------------------
--*\
  =========                 |
  \\      / F ield          | OpenFOAM: 开源计算流体动力学工具箱
   \\    /   O peration     | 网站:     https://OpenFOAM.org
    \\  /    A nd           | 版本:     10
     \\/     操纵            |
\*---------------------------------------------------------------------------
--*/
Foam文件
{
    格式          ASCII;
    类            字典;
    对象          有限体积约束;
}
// ************************************
* //
限制压力
{
    type       极限压力;
    最小系数   0.5;
    最大系数 2;
}
```

```
//
*********************************************************************
//
input_file_end.```
```

Sub-database allrun files:

```
```input_file_begin: linux execution command allrun file of case pitzDaily
(domain: compressible, category: LES, solver:rhoPimpleFoam):
#!/bin/sh
cd ${0%/*} || exit 1    # Run from this directory

# Source tutorial run functions
. $WM_PROJECT_DIR/bin/tools/RunFunctions

application="$(getApplication)"

runApplication blockMesh -dict
$FOAM_TUTORIALS/resources/blockMesh/pitzDaily
runApplication $application

#-------------------------------------------------------------------------
----
input_file_end.```
```

Appendix C. DNS HIT complete flowchart

In this section, we use the HIT (Homogeneous Isotropic Turbulence) example to demonstrate the basic workflow of MetaOpenFOAM. The corresponding user requirement is to "do a DNS simulation of incompressible forcing homogeneous isotropic turbulence with Grid 32^3 using dnsFoam."

In this user requirement, we provide the case description, category, solver, model, and grid information, while omitting details such as boundary conditions, initial conditions, run time, and time steps. This approach aligns with typical user practices.

First, based on the user requirement, the Architect converts it into a standardized format, including case name, case domain, case category, and case solver. Then, it searches the database for the most similar case, retrieves its file architecture, and writes a new file architecture suitable for the current case based on the similar case's architecture and user requirements. This new file architecture is divided into multiple subtasks and passed to the InputWriter. It is important to note that each file to be generated in the architecture corresponds to a subtask.

```
//
*********************************************************************
//
输入_文件_结束。```
```

子数据库allrun文件：

```
```输入_文件_开始：案例皮茨日报的Linux执行命令allrun文件
(域：可压缩，类别：LES，求解器：rhoPimpleFoam)：
#!/bin/sh
cd ${0%/*} || exit 1        # 从当前目录运行

# 源教程运行函数
. $WM_PROJECT_DIR/bin/tools/RunFunctions

应用程序     ="$(获取应用程序)      "

运行应用程序 块网格 -字典
$FOAM_教程/resources/blockMesh/皮茨日报
运行应用程序 $application


#-------------------------------------------------------------------
----
输入_文件_结束。```
```

附录C. 直接数值模拟HIT完整流程图

在本节中，我们使用HIT（均匀各向同性湍流）示例进行演示MetaOpenFOAM的基本工作流程。对应的用户要求是"进行一个直接数值模拟使用网格 32^3 模拟不可压缩强迫均匀各向同性湍流，采用dnsFoam求解器。"

在此用户要求中，我们提供了案例描述、类别、求解器、模型和网格信息，同时省略了边界条件、初始条件、运行时间和时间步长等细节。这种方法符合典型的用户实践。

首先，架构师根据用户需求将其转换为标准化格式，包括案例名称、案例领域、案例类别和案例求解器。随后，在数据库中搜索最相似的案例，检索其文件架构，并根据相似案例的架构和用户需求编写适合当前案例的新文件架构。该新文件架构被划分为多个子任务并传递给输入写入器。需注意的是，架构中每个待生成文件都对应一个子任务。

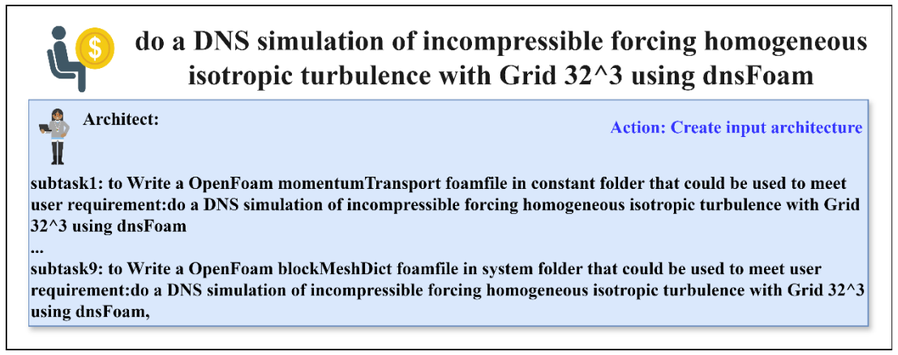


Figure 8. Sketch of Action Create input architecture in HIT case

Next, the InputWriter uses the information provided by the Architect, along with the file information from the similar case in the database, to write the input files for each subtask. Once all subtasks are completed, the work is passed to the Runner. The Runner writes an Allrun file suitable for the current requirement based on the Allrun file information from the similar case in the database and executes it in the OpenFOAM environment. If there are no errors and the Executability >= 3, the loop exits. However, when executing the HIT example, an error "No 'neighbourPatch' provided" was encountered. Therefore, the Runner passes this error information to the Reviewer for correction.

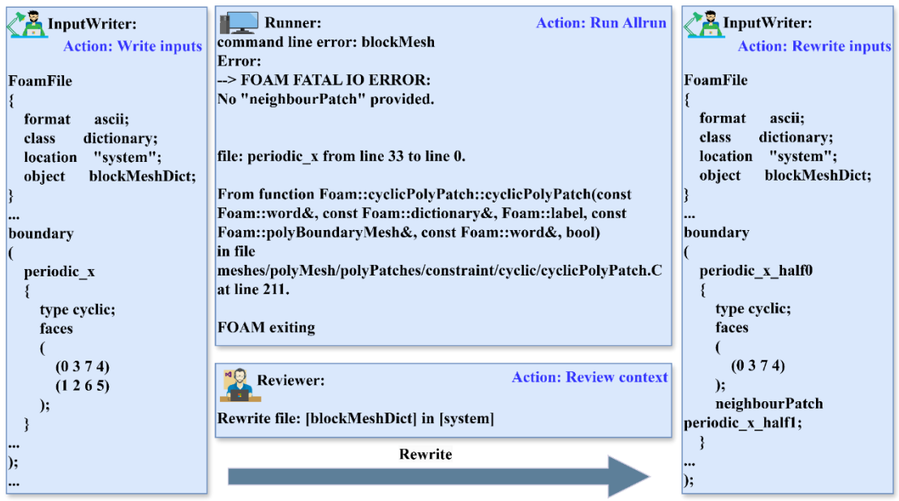


Figure 9. Sketch of the first iteration using MetaOpenFOAM in HIT case

The Reviewer first determines whether the error is related to the file architecture or the file content. In the HIT example, the Reviewer identifies the error as being related to the content of the blockMeshDict file, so it is returned to the InputWriter for modification. This entire process

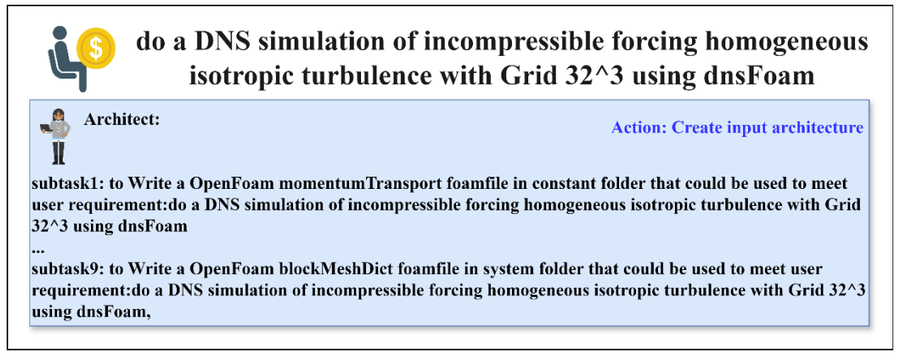


图8. HIT案例中创建输入架构操作示意图

接着，输入写入器利用架构师提供的信息，结合数据库中相似案例的文件信息，为每个子任务编写输入文件。所有子任务完成后，工作将传递给运行器。运行器根据数据库中相似案例的Allrun文件信息，编写符合当前需求的Allrun文件，并在OpenFOAM环境中执行。若未出现错误且可执行性>= 3满足，则循环退出。但在执行HIT示例时，遇到了"未提供‘neighbourPatch’"的错误，因此运行器将该错误信息传递给评审员进行修正。

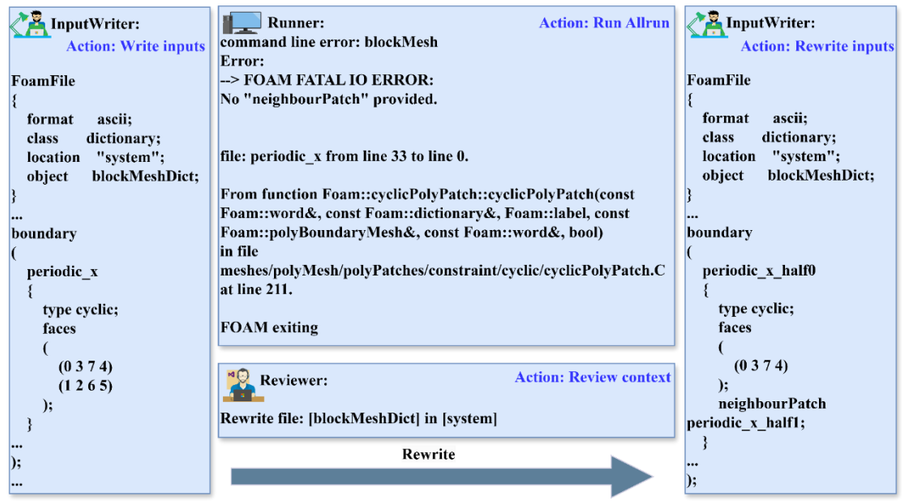


图9. HIT案例中使用MetaOpenFOAM进行第一次迭代的示意图

评审员首先判断错误是与文件架构相关还是与文件内容相关。

在HIT示例中，评审员判定该错误与

blockMeshDict文件，因此它被返回给输入写入器进行修改。整个过程

constitutes the first iteration.

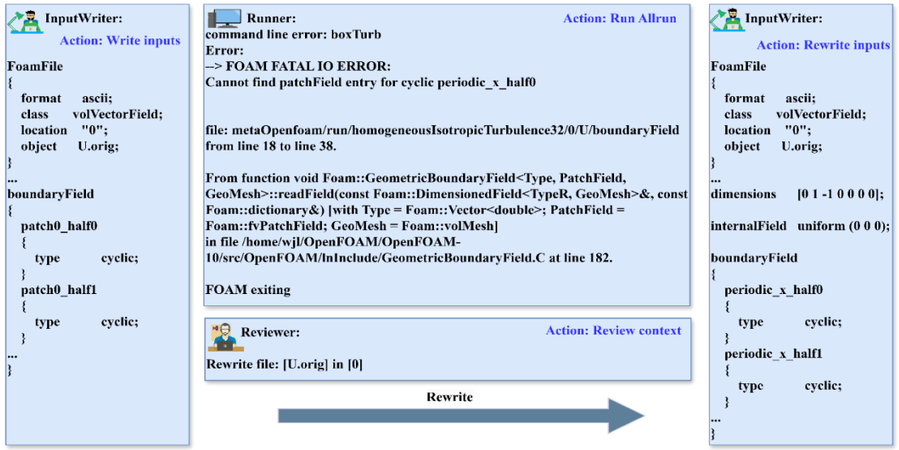


Figure 10. Sketch of the second iteration using MetaOpenFOAM in HIT case

The Runner then continues to execute the modified input files but encounters a new error: "cannot find patchField entry for cyclic periodic_x_half0," and passes this to the Reviewer. The Reviewer determines that the issue lies with the content of the U.orig file, so it instructs the InputWriter to rewrite U.orig. The Runner then runs the simulation again, ultimately achieving successful execution of the HIT example with Executability = 4.

Appendix D. Detailed performance and test examples of main results

Table 5. Performance of MetaOpenFOAM

	Running time (s)	Prompt Tokens	Completion Tokens	Number of input files	Lines per input file	Total lines of input files
HIT	189	9181	3486.4	10.8	32.3	348.7
PitzDaily	649	13426	4656.8	11	51.2	563.7
Cavity	98	8709	4154.5	12	37.8	454
LidDrivenCavity	200	45253	6837.5	9.2	38	347.9
SquareBendLiq	154	10562	5823.4	15	39.5	593
PlanarPoiseuille	181	30122	5410	11	40	437
CounterFlowFlame	260	37459	10469.2	20.4	111	2363.4
BuoyantCavity	439	142209	14602.8	19.8	49.01	972

构成第一次迭代。

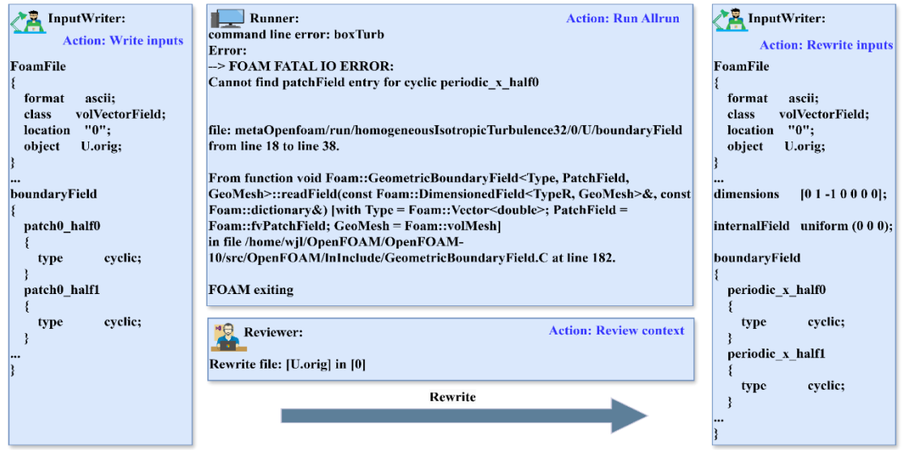


图10. HIT案例中使用MetaOpenFOAM的第二次迭代示意图

运行器随后继续执行修改后的输入文件，但遇到了一个新的错误：

"无法找到cyclic periodic_x_half0的patchField条目,"并将其传递给评审员。

评审员确定问题出在U.orig文件的内容上，因此指示

输入写入器重写U.orig文件。随后运行器再次执行模拟，最终成功完成

HIT示例成功执行，可执行性为= 4。

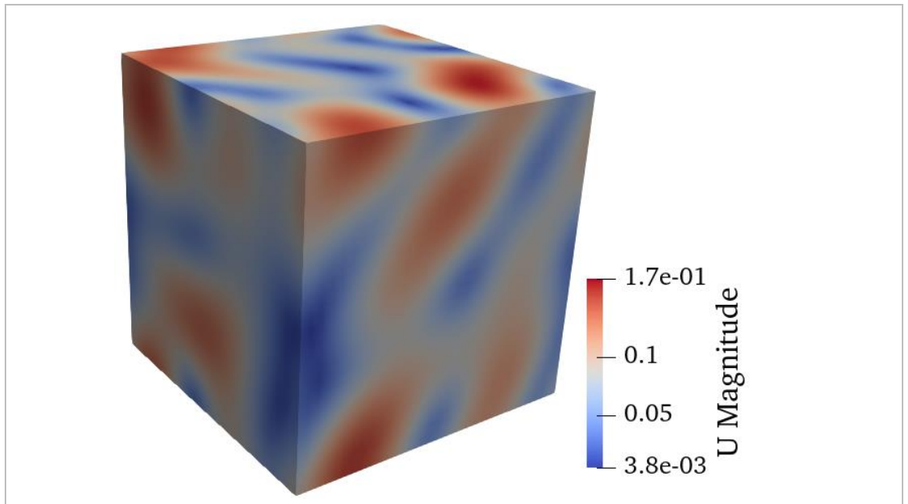
附录D. 主要结果的详细性能与测试示例

表5. MetaOpenFOAM的性能

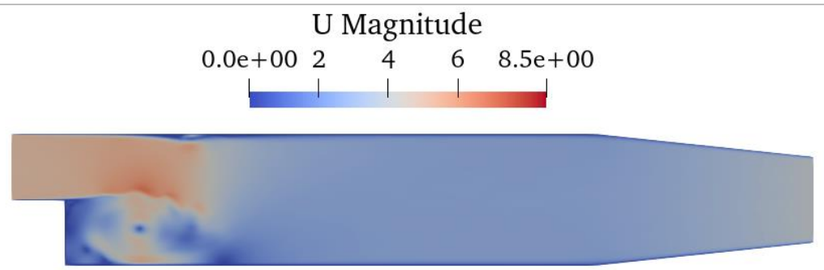
	运行时间（秒）	提示令牌	完成令牌	输入文件数量	每个输入文件的行数	输入文件总行数
HIT	189	9181	3486.4	10.8	32.3	348.7
皮茨日报	649	13426	4656.8	11	51.2	563.7
腔体	98	8709	4154.5	12	37.8	454
盖驱动腔	200	45253	6837.5	9.2	38	347.9
方形弯管液体	154	10562	5823.4	15	39.5	593
平面泊肃叶	181	30122	5410	11	40	437
逆流火焰	260	37459	10469.2	20.4	111	2363.4
浮力腔	439	142209	14602.8	19.8	49.01	972

Average	418.375	45906	7774	13.65	50	760

Where "Prompt Tokens" refer to the tokens required for the prompts sent to the LLM, while "Completion Tokens" are the tokens needed for the LLM's responses. In MetaOpenFOAM, Prompt Tokens are often more numerous than Completion Tokens. This trend becomes more pronounced with an increasing number of iterations. This is because identifying the cause of errors necessitates transmitting all input files to the LLM. The more iterations there are, the more frequently the input files are transmitted, resulting in a higher number of Prompt Tokens compared to Completion Tokens.



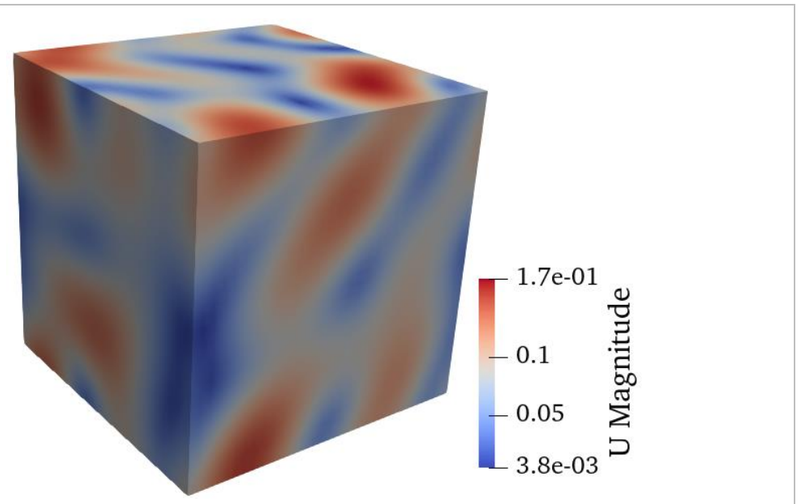
(a)



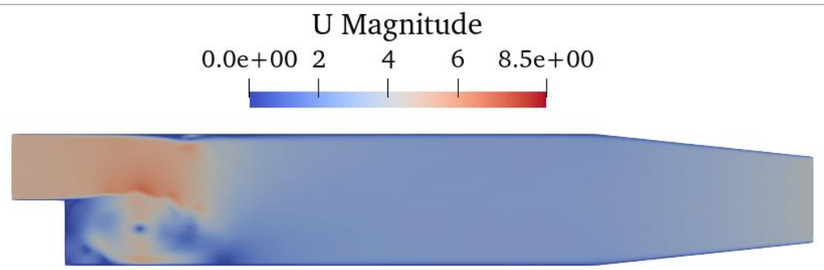
(b)

平均值	418.375	45906	7774	13.65	50	760

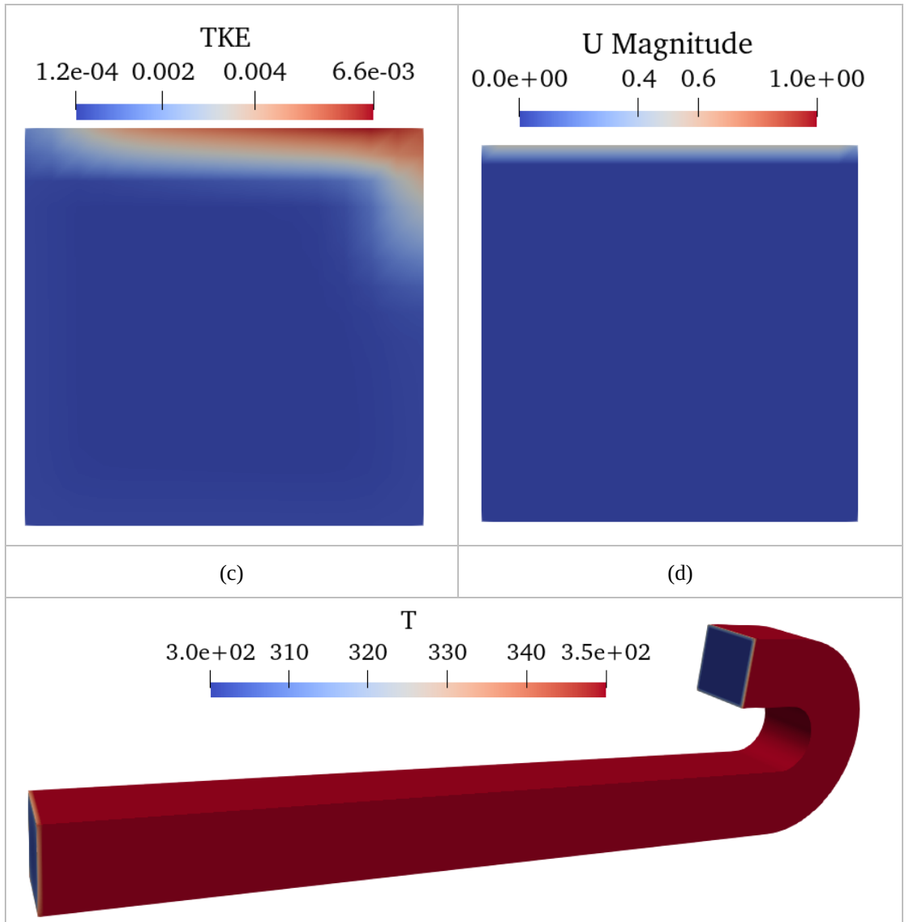
其中“提示令牌”指发送给LLM的提示所需的令牌，而“完成令牌”则是LLM响应所需的令牌。在MetaOpenFOAM中，提示令牌通常多于完成令牌。随着迭代次数的增加，这一趋势更为明显。这是因为识别错误原因需要将所有输入文件传输给LLM。迭代次数越多，输入文件传输越频繁，导致提示令牌数量相较完成令牌更高。



(a)



(b)


TKE
1.2e-04 0.002 0.004 6.6e-03
U Magnitude
0.0e+00 0.4 0.6 1.0e+00

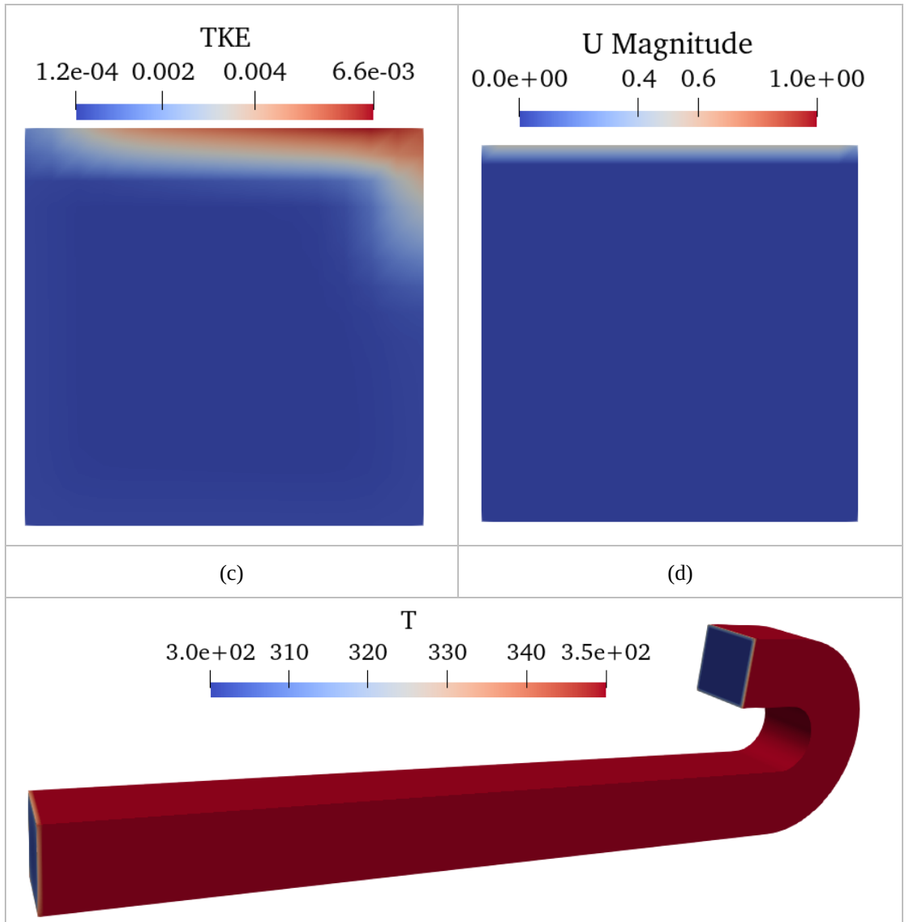
(c) (d)

T
3.0e+02 310 320 330 340 3.5e+02

(e)

U Magnitude
0.0e+00 40 60 80 100 120 1.5e+02

(f)


TKE
1.2e-04 0.002 0.004 6.6e-03
U Magnitude
0.0e+00 0.4 0.6 1.0e+00

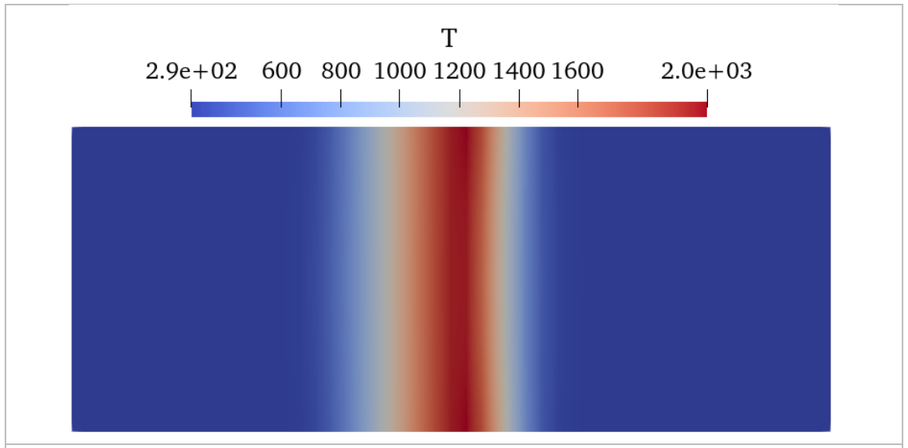
(c) (d)

T
3.0e+02 310 320 330 340 3.5e+02

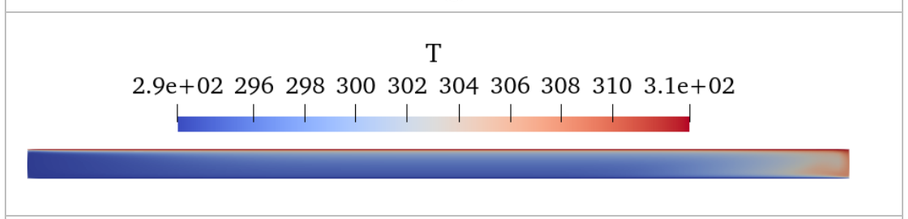
(e)

U Magnitude
0.0e+00 40 60 80 100 120 1.5e+02

(f)

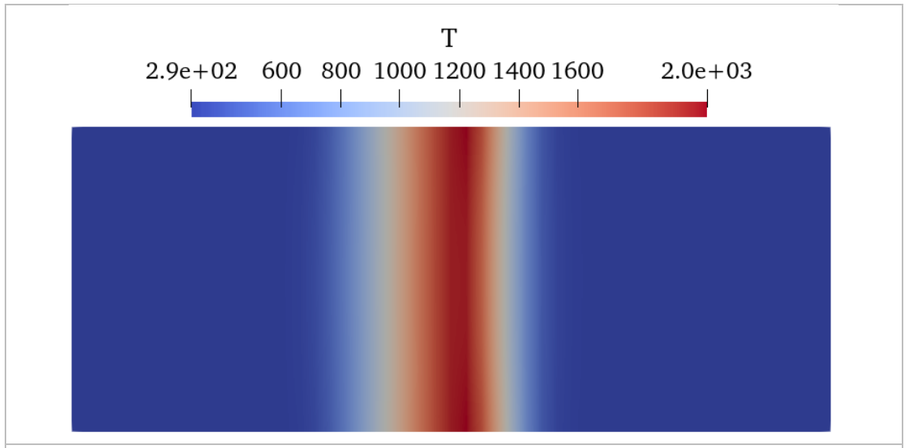


(g)

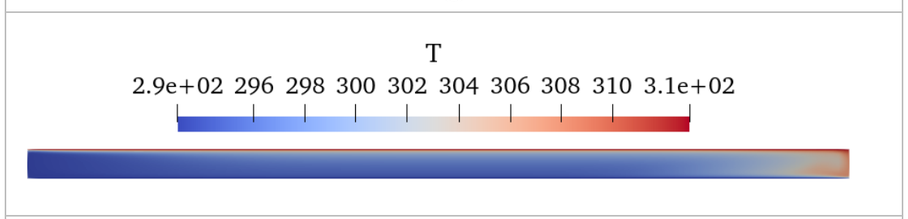


(h)

Figure 11. Simulation results simulated by MetaOpenFOAM. (a) Magnitude velocity of Homogeneous Isotropic Turbulence (HIT) (b) Magnitude velocity of PitzDaily (c) Turbulent kinetic energy of Cavity (d) Magnitude velocity of Lid-Driven Cavity (e) Temperature of Square Bend Liquid (SquareBendLiq) (f) Magnitude velocity of Planar Poiseuille (g) Temperature of Counter Flow Flame (h) Temperature of Buoyant Cavity (gravity direction to the left)



(g)



(h)

图11. MetaOpenFOAM模拟结果。(a) 均匀各向同性湍流（HIT）的速度幅值 (b) 皮茨日报的速度幅值 (c) 腔体的湍流动能 (d) 顶盖驱动空腔的速度幅值 (e) 方形弯管液体（SquareBendLiq）的温度 (f) 平面泊肃叶的速度幅值 (g) 逆流火焰的温度 (h) 浮力空腔的温度（重力方向向左）

Appendix E. Ablation study

Table 6. Performance of MetaOpenFOAM when remove Reviewer

	Executability	**Token Usage**	**Iteration**	**Productivity**	**Pass@1(%)**
HIT	0.6	11872.4	0	32.3	0
PitzDaily	1	12631.8	0	22.5	0
Cavity	4	12848	0	28.5	100
LidDrivenCavity	1.6	12681.6	0	40.8	20
SquareBendLiq	4	16371.8	0	27.5	100

附录E. 消融研究

表6. 移除评审员时MetaOpenFOAM的性能

	可执行性	令牌使用量	迭代	生产力	**通过率@1(%)**
HIT	0.6	11872.4	0	32.3	0
皮茨日报	1	12631.8	0	22.5	0
腔体	4	12848	0	28.5	100
盖驱动腔	1.6	12681.6	0	40.8	20
方形弯管液体	4	16371.8	0	27.5	100

PlanarPoiseuille	1	12578.8	0	29.9	0
CounterFlowFlame	1	20781.8	0	26.1	0
BuoyantCavity	0.2	25521.3	0	31.1	0
Average	1.7	15660.94	0	29.9	27.5

Table 7. Performance of MetaOpenFOAM when remove Action Review architecture

	Executability	**Token Usage**	**Iteration**	**Productivity**	**Pass@1(%)**
HIT	0.8	165196.0	20	488.2	0
PitzDaily	4	18083.2	2.1	32.1	100
Cavity	4	12863.6	0	28.3	100
LidDrivenCavity	2.2	73685.2	12.4	241.0	40
SquareBendLiq	3.7	21529.2	2	36.4	90
PlanarPoiseuille	2.8	32691.0	5	74.0	90
CounterFlowFlame	4	78864.4	4	85.4	100
BuoyantCavity	2.4	151771.6	15.8	158.4	40
Average	3.0	69335.5	7.7	143.0	70

Table 8. Performance of MetaOpenFOAM when remove RAG

	Executability	**Token Usage**	**Iteration**	**Productivity**	**Pass@1(%)**
HIT	0	7000.4	20	15.8	0
PitzDaily	1.4	95145.6	17.4	187.9	0
Cavity	0	118260	20	287.0	0
LidDrivenCavity	1	49751	20	164.2	0
SquareBendLiq	0.5	58698	20	229.7	0
PlanarPoiseuille	1.6	83655.8	16	232.8	0
CounterFlowFlame	1	132260	20	136.1	0
BuoyantCavity	1	105996	20	12.2	0
Average	0.8125	81345.9	19.2	158.2	0

平面泊肃叶	1	12578.8	0	29.9	0
逆流火焰	1	20781.8	0	26.1	0
浮力腔	0.2	25521.3	0	31.1	0
平均值	1.7	15660.94	0	29.9	27.5

表7. 移除Action Review架构时MetaOpenFOAM的性能

	可执行性	令牌使用量	迭代	生产力	**通过率@1(%)**
HIT	0.8	165196.0	20	488.2	0
皮茨日报	4	18083.2	2.1	32.1	100
腔体	4	12863.6	0	28.3	100
盖驱动腔	2.2	73685.2	12.4	241.0	40
方形弯管液体	3.7	21529.2	2	36.4	90
平面泊肃叶	2.8	32691.0	5	74.0	90
逆流火焰	4	78864.4	4	85.4	100
浮力腔	2.4	151771.6	15.8	158.4	40
平均值	3.0	69335.5	7.7	143.0	70

表8. 移除RAG时MetaOpenFOAM的性能

	可执行性	令牌使用量	迭代	生产力	**通过率@1(%)**
HIT	0	7000.4	20	15.8	0
皮茨日报	1.4	95145.6	17.4	187.9	0
腔体	0	118260	20	287.0	0
盖驱动腔	1	49751	20	164.2	0
方形弯管液体	0.5	58698	20	229.7	0
平面泊肃叶	1.6	83655.8	16	232.8	0
逆流火焰	1	132260	20	136.1	0
浮力腔	1	105996	20	12.2	0
平均值	0.8125	81345.9	19.2	158.2	0

Appendix F. Detailed discussion of temperature

Table 9. Performance of MetaOpenFOAM when temperature=0.5 in LLM.

	Executability	Token Usage	Iteration	Productivity	Pass@1(%)
HIT	1.8	128321.8	14.6	300.4	30
PitzDaily	3.7	36209.7	4.7	58.6	90
Cavity	3.7	30568.6	4.7	19.4	90
LidDrivenCavity	3.7	41296.1	6.4	44.0	90
SquareBendLiq	3.6	16449	2.9	27.6	90
PlanarPoiseuille	4	40384.9	6.2	92.0	100
CounterFlowFlame	3.1	66518.5	10.6	46.3	70
BuoyantCavity	3.7	107599.4	9.3	111.6	90
Average	3.4	58418.5	7.4	87.5	81.3

Table 10. Performance of MetaOpenFOAM when temperature=0.99 in LLM

	Executability	Token Usage	Iteration	Productivity	Pass@1(%)
HIT	1.3	161316	18	407.9	10
PitzDaily	3	61516.3	9.3	100.0	60
Cavity	3.2	38278.4	8.6	70.3	80
LidDrivenCavity	2.2	78580.8	13	199.1	40
SquareBendLiq	2	48812.4	12.8	75.8	40
PlanarPoiseuille	3.3	55319.4	8.3	126.0	80
CounterFlowFlame	2.8	165198.2	17.6	178.0	60
BuoyantCavity	1	264538	20	277.6	0
Average	2.3	109194.9	13.5	179.3	46.3

Appendix G. Modified dataset study

Table 11. Performance of MetaOpenFOAM in modified dataset

	Executability	Token Usage	Iteration	Productivity	Pass@1(%)

附录 F. 温度的详细讨论

表 9. LLM中温度=0.5时MetaOpenFOAM的性能

	可执行性	令牌使用量	迭代	生产力	通过率@1(%)
HIT	1.8	128321.8	14.6	300.4	30
皮茨日报	3.7	36209.7	4.7	58.6	90
腔体	3.7	30568.6	4.7	19.4	90
盖驱动腔	3.7	41296.1	6.4	44.0	90
方形弯管液体	3.6	16449	2.9	27.6	90
平面泊肃叶	4	40384.9	6.2	92.0	100
逆流火焰	3.1	66518.5	10.6	46.3	70
浮力腔	3.7	107599.4	9.3	111.6	90
平均值	3.4	58418.5	7.4	87.5	81.3

表10. LLM中温度=0.99时MetaOpenFOAM的性能

	可执行性	令牌使用量	迭代	生产力	通过率@1(%)
HIT	1.3	161316	18	407.9	10
皮茨日报	3	61516.3	9.3	100.0	60
腔体	3.2	38278.4	8.6	70.3	80
盖驱动腔	2.2	78580.8	13	199.1	40
方形弯管液体	2	48812.4	12.8	75.8	40
平面泊肃叶	3.3	55319.4	8.3	126.0	80
逆流火焰	2.8	165198.2	17.6	178.0	60
浮力腔	1	264538	20	277.6	0
平均值	2.3	109194.9	13.5	179.3	46.3

附录G. 修改后的数据集研究

表11. MetaOpenFOAM在修改后数据集中的表现

	可执行性	令牌使用量	迭代	生产力	通过率@1(%)

HIT	4	31386.8	4.6	91.8	100
PitzDaily	4	18507.2	2.2	33.2	100
Cavity	4	14697.8	1.4	33.3	100
LidDrivenCavity	4	31256.6	4.4	91.9	100
SquareBendLiq	2	16442.4	20	27.2	0
PlanarPoiseuille	4	59351	7	138.2	100
CounterFlowFlame	4	35274.8	2.8	38.0	100
BuoyantCavity	3.6	180419	15	180.0	80
Average	3.7	48417.0	7.2	79.2	85

HIT	4	31386.8	4.6	91.8	100
皮茨日报	4	18507.2	2.2	33.2	100
腔体	4	14697.8	1.4	33.3	100
盖驱动腔	4	31256.6	4.4	91.9	100
方形弯管液体	2	16442.4	20	27.2	0
平面泊肃叶	4	59351	7	138.2	100
逆流火焰	4	35274.8	2.8	38.0	100
浮力腔	3.6	180419	15	180.0	80
平均值	3.7	48417.0	7.2	79.2	85