



## **CFDAgent: A language-guided, zero-shot multi-agent system for complex flow simulation**

Downloaded from: <https://research.chalmers.se>, 2025-12-05 06:31 UTC

Citation for the original published paper (version of record):

Xu, Z., Wang, L., Wang, C. et al (2025). CFDAgent: A language-guided, zero-shot multi-agent system for complex flow simulation. *Physics of Fluids*, 37(11). <http://dx.doi.org/10.1063/5.0294696>

N.B. When citing this work, cite the original published paper.

RESEARCH ARTICLE | NOVEMBER 11 2025

# CFDAgent: A language-guided, zero-shot multi-agent system for complex flow simulation

Zhaoyue Xu (许昭越) ; Long Wang (王笼) ; Chunyu Wang (王春雨) ; Yixin Chen (陈一鑫) ;  
Qingyong Luo (罗清勇) ; Hua-Dong Yao (姚华栋)  ; Shizhao Wang (王士召)  ;  
Guowei He (何国威) 

*Physics of Fluids* 37, 117124 (2025)<https://doi.org/10.1063/5.0294696>

## Articles You May Be Interested In

OpenFOAMGPT: A retrieval-augmented large language model (LLM) agent for OpenFOAM-based computational fluid dynamics

*Physics of Fluids* (March 2025)

Direct numerical simulation of multiscale flow physics of binary droplet collision

*Physics of Fluids* (June 2020)

Water exit dynamics of jumping archer fish: Integrating two-phase flow large-eddy simulation with experimental measurements

*Physics of Fluids* (January 2020)



Physics of Fluids

## Special Topics Open for Submissions

[Learn More](#)

# CFDAgent: A language-guided, zero-shot multi-agent system for complex flow simulation

Cite as: Phys. Fluids **37**, 117124 (2025); doi: [10.1063/5.0294696](https://doi.org/10.1063/5.0294696)

Submitted: 5 August 2025 · Accepted: 20 October 2025 ·

Published Online: 11 November 2025










View Online



Export Citation



CrossMark

Zhaoyue Xu (许昭越),<sup>1,2,a)</sup>  Long Wang (王笼),<sup>1,3,a)</sup>  Chunyu Wang (王春雨),<sup>1,3,a)</sup>  Yixin Chen (陈一鑫),<sup>1,3,a)</sup>   
 Qingyong Luo (罗清勇),<sup>1,3,a)</sup>  Hua-Dong Yao (姚华栋),<sup>1,2,b)</sup>  Shizhao Wang (王士召),<sup>1,3,b)</sup>   
 and Guowei He (何国威)<sup>1,3,a)</sup> 

## AFFILIATIONS

<sup>1</sup>The State Key Laboratory of Nonlinear Mechanics, Institute of Mechanics, Chinese Academy of Sciences, Beijing 100190, China

<sup>2</sup>Department of Mechanics and Maritime Sciences, Chalmers University of Technology, Gothenburg 41296, Sweden

<sup>3</sup>School of Engineering Sciences, University of Chinese Academy of Sciences, Beijing 100049, China

<sup>a)</sup>Electronic addresses: [zhaoyue@chalmers.se](mailto:zhaoyue@chalmers.se); [wanglong@imech.ac.cn](mailto:wanglong@imech.ac.cn); [wangchunyu@imech.ac.cn](mailto:wangchunyu@imech.ac.cn); [chenyixin@imech.ac.cn](mailto:chenyixin@imech.ac.cn); [luoqingyong@imech.ac.cn](mailto:luoqingyong@imech.ac.cn); and [hgw@lnm.imech.ac.cn](mailto:hgw@lnm.imech.ac.cn)

<sup>b)</sup>Authors to whom correspondence should be addressed: [huadong.yao@chalmers.se](mailto:huadong.yao@chalmers.se) and [wangsz@lnm.imech.ac.cn](mailto:wangsz@lnm.imech.ac.cn)

## ABSTRACT

We introduce CFDAgent, a zero-shot, language-guided multi-agent framework that enables fully autonomous computational fluid dynamics (CFD) simulations from natural language prompts. CFDAgent integrates three specialized large language model driven agents: (i) the Preprocessing Agent generates Lagrangian surface meshes from inputs including natural language descriptions, two-dimensional images, or three-dimensional geometry; (ii) the Solver Agent configures and executes an immersed boundary flow solver; and (iii) the Postprocessing Agent analyzes and visualizes the results, including quantitative plots, flow field visualizations, and photorealistic renderings. These agents operate through an interactive, dialogue-based process guided by a generative pretrained transformer (GPT-4o), enabling intuitive user interaction. We validate CFDAgent against canonical benchmark cases—flow past spheres, cubes, and cylinders—at various Reynolds numbers, demonstrating excellent agreement with established literature data. The framework successfully simulates flows around complex real-world geometries, all through simple text prompts. This zero-shot capability represents a fundamental shift from traditional CFD workflows, eliminating the need for extensive preprocessing expertise, mesh generation skills, and case-specific setup. By bridging generative artificial intelligence with CFD simulations, CFDAgent significantly lowers technical barriers to expert-level CFD, unlocking broad opportunities in education, scientific research, and engineering applications while making advanced flow simulations accessible to nonspecialists.

© 2025 Author(s). All article content, except where otherwise noted, is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 4.0 International (CC BY-NC-ND) license (<https://creativecommons.org/licenses/by-nc-nd/4.0/>). <https://doi.org/10.1063/5.0294696>

## I. INTRODUCTION

Large Language Models (LLMs), particularly OpenAI's GPT series,<sup>1–4</sup> have significantly expanded the capabilities of artificial intelligence (AI), driving transformative advances in natural language processing, reasoning, and zero-shot learning. Rooted fundamentally in the Transformer architecture,<sup>5</sup> these models have revolutionized diverse fields by enhancing tasks related to text generation, knowledge extraction, and generative functionalities.<sup>3,4</sup> Beyond natural language applications, GPT models also exhibit strong capabilities in code generation and computational physics, achieving promising results in various programming-related benchmarks.<sup>4,6</sup>

Despite their impressive capabilities, current standalone LLMs remain limited in tasks requiring extensive, precise, and systematic

content generation—particularly in computational physics<sup>7</sup> due to persistent challenges such as hallucination.<sup>8</sup> As elucidated by Wolfram,<sup>9</sup> fundamental constraints inherent in these models restrict their ability to autonomously “solve science.” In particular, although LLMs can effectively assist in accelerating the development of numerical solvers in fluid dynamics or other physical problems,<sup>6</sup> they remain insufficient as replacements for simulations, especially in tasks involving complex geometry handling, rigorous physical modeling, and sophisticated numerical methods such as Computational Fluid Dynamics (CFD). Despite significant research integrating AI into CFD workflows,<sup>10–15</sup> current AI technologies—including LLMs—remain unable to function as direct solvers for the underlying governing equations.<sup>9</sup>

To mitigate these limitations, integrating LLM capabilities into simulation frameworks has emerged as an increasingly promising direction.<sup>16</sup> Recent studies<sup>16–20</sup> have proposed hybrid approaches leveraging LLM-driven techniques for enhancing simulations and engineering design optimization. Notably, these approaches commonly involve prompt-based interactions, automatic equation formulation, and assisted simulation frameworks. Specifically, the combination of LLMs and CFD has garnered substantial research attention and demonstrated initial success. For example, Dong *et al.*<sup>20</sup> complement this by fine-tuning a 7B-parameter Qwen model on 28k-NL2FOAM pairs and orchestrating a multi-agent pipeline that translates natural language descriptions into executable OpenFOAM cases with 88.7% accuracy and markedly fewer correction iterations than much larger general-purpose LLMs. Wang *et al.*<sup>16</sup> systematically benchmark reasoning-augmented and vanilla LLMs on tasks ranging from canonical lid-driven cavity flow to ill-conditioned Hilbert systems, showing that tailored prompts let reasoning models excel even without fine-tuning. Zhang *et al.*<sup>21</sup> propose LLM-PSO, an evolutionary strategy-based framework that leverages large language models for parametric shape optimization through in-context learning, demonstrating competitive performance against traditional genetic algorithms across multiple fluid dynamics applications including airfoil design, axisymmetric body optimization, and heat exchanger fin configuration, while achieving rapid convergence and computational efficiency without requiring model retraining, thus opening new avenues for AI-assisted engineering optimization. Pandey *et al.*<sup>22</sup> introduce OpenFOAMGPT, a retrieval-augmented LLM agent that automates CFD workflows in OpenFOAM through natural language interactions, refining code, minimizing syntax errors, and providing a logical framework for high-fidelity simulations with domain-specific guidance, thereby substantially lowering technical barriers for nonspecialists. Wang *et al.*<sup>23</sup> further optimize the OpenFOAMGPT framework by benchmarking different LLMs, significantly reducing token costs by up to two orders of magnitude while maintaining performance. These examples collectively highlight the potential of fine-tuned, zero-shot, and agentic LLM approaches. Consequently, the integration of advanced LLM agent-based frameworks with CFD tools represents a promising direction to bridge the gap between natural language understanding and computational simulation, specifically pointing toward the feasibility of zero-shot, language-guided execution of complex flow simulations without extensive manual intervention. Motivated by these advancements and challenges, in this work, we propose an LLM-driven multi-agent framework that coordinates preprocessing, solver execution, and post analysis entirely through natural language interaction to achieve fully automated, end-to-end simulations in CFD workflows.

In this work, we refer to this multi-agent framework as CFDAgent, which enables fully language-guided, zero-shot, end-to-end simulations of complex flows. Through natural language interaction, the framework is designed to manage three core stages that constitute the workflow of a typical complex flow simulation: (1) preprocessing, which includes geometry and mesh generation; (2) flow solving, which involves numerically solving the governing fluid dynamics equations under specified initial and boundary conditions; and (3) postprocessing, which encompasses the analysis and visualization of simulation results. Correspondingly, CFDAgent leverages LLM-driven agents to overcome the substantial demands on human and computational resources typically required in these stages.

The Preprocessing Agent integrates a hybrid text-to-three-dimensional (3D) diffusion model, Point-E, to autonomously generate point cloud and transforms it into the mesh. Point-E generates 3D point clouds from natural language prompts by first synthesizing images via GLIDE<sup>24</sup> and then reconstructing geometry, achieving 10 to 100 times faster sampling than earlier text-to-3D approaches.<sup>25–32</sup> In the second stage, the Solver Agent enables the LLM to guide users in configuring simulation parameters and launching the flow solver. Complementing this, our Immersed Boundary (IB) method, rooted in Peskin's foundational work<sup>33–35</sup> and leveraging direct-forcing methods,<sup>36–44</sup> imposes boundary conditions on complex geometries without body-fitted meshes by solving small, localized linear systems.<sup>45</sup> Enhanced through parallel domain decomposition, our implementation delivers both high accuracy and computational efficiency, as validated by extensive benchmarks.<sup>45–49</sup> Finally, the Postprocessing Agent provides the LLM with scripts for analyzing physical quantities, visualizing flow fields, and using simulation results to generate realistic imagery of physical objects.

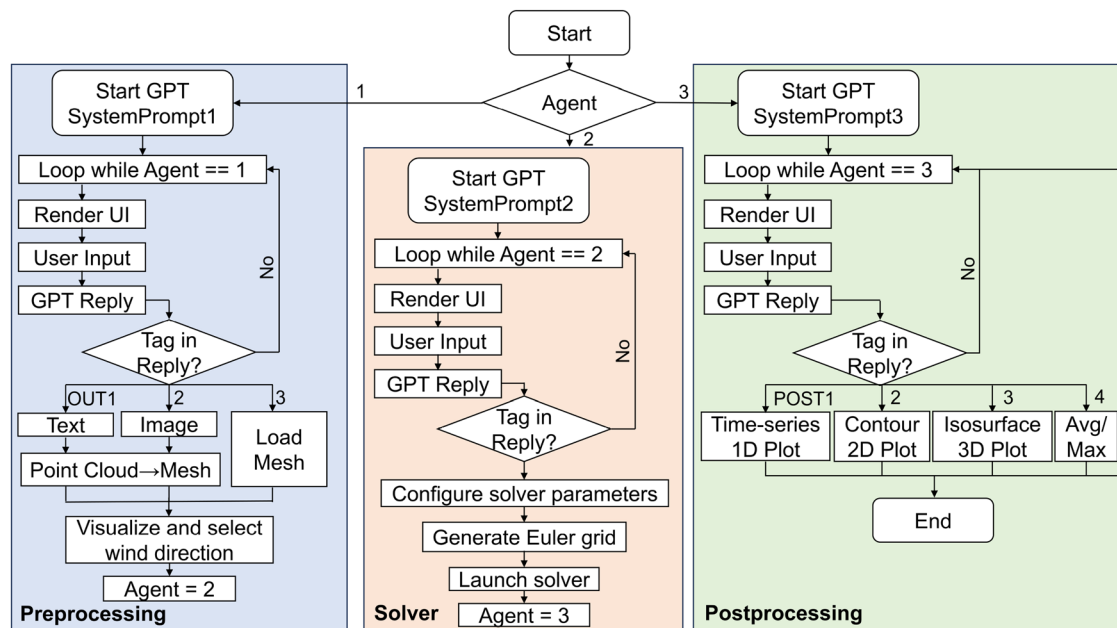
## II. METHODS

### A. Zero-shot multi-agent system

Implemented as an interactive web application using Streamlit,<sup>50</sup> the framework incorporates an OpenAI GPT-based assistant<sup>4</sup> to enable user-agent dialogue and task coordination. The proposed three-agent framework, whose workflow is presented in Algorithm 1 in Appendix C and visualized in Fig. 1, enables the end-to-end transformation from high-level natural language specifications to fully configured simulations. Algorithm 1 presents a simplified finite-state implementation where a single variable  $\text{agent} \in \{1, 2, 3\}$  coordinates the sequential progression through the three agents. While certain implementation details are omitted for clarity and brevity, the algorithm captures the essential conversation-driven mechanism.

The Preprocessing Agent initiates dialogue and determines if the user intends to run a fluid dynamics simulation. In the first stage, the user engages with a GPT-driven conversational interface. If the user expresses interest in a fluid simulation, GPT starts to assist in the simulation. Through a dialogue, the user is asked to specify the object of interest either by name or description (e.g., “dog” or “sphere”), by uploading a reference image of the object or by providing a preexisting 3D mesh file. The user also specifies the flow conditions, notably the desired Reynolds number for the simulation—a dimensionless quantity that characterizes the nature of the flow. If the object is provided only as a text description or an image, the framework generates a corresponding 3D model along with its surface mesh. This is accomplished using OpenAI's Point-E, which can produce a three-dimensional point cloud representation of an object from either a textual description or a single image. Once a sufficient point cloud is obtained, a signed distance function (SDF) regression model is applied to convert the point cloud into a continuous volumetric representation, and a Marching Cubes algorithm extracts a surface mesh from this field. The resulting surface mesh, also referred to as the Lagrangian mesh, approximates the target object's geometry. If the user instead uploads an existing 3D mesh, the generative modeling step is skipped, and the provided mesh is used directly. Through an agent-provided interface, the user defines the incoming flow direction with respect to the object, ensuring correct geometric orientation within the simulation. To accomplish this, the agent initially assumes the role of a





**FIG. 1.** Workflow diagram of CFDAgent. The system employs three specialized agents—Preprocessing, Solver, and Postprocessing—each configured with tailored system prompts to handle specific stages of the CFD simulation pipeline. Users interact with the system through natural language, with LLM responses containing tagged instructions (e.g., OUT1 and POST1) that trigger corresponding operations. The preprocessing agent handles geometry import and mesh generation. The solver agent configures simulation parameters and executes the IB simulation. The postprocessing agent generates comprehensive visualizations and quantitative analyses.

preprocessing expert by invoking system prompt. Following this initialization, the agent analyzes the response to identify one of three designated output tags:

- **###OUT1###:** Generate a point cloud from textual geometry descriptions and construct the corresponding surface mesh;
- **###OUT2###:** Generate a point cloud from uploaded images and construct the corresponding surface mesh;
- **###OUT3###:** Import a user-provided surface mesh directly.

Within the Solver Agent, the simulation parameters are first configured, followed by the launching of the simulation. Upon initial entry, the agent is reconfigured through system prompt to function as a CFD solver assistant. The user is prompted to provide essential CFD settings: the total simulation duration (end time), the Courant–Friedrichs–Lewy (CFL) number for numerical stability, the time step, boundary conditions, and the output frequency for saving simulation data. The GPT-based assistant suggests default or recommended values for these inputs. Utilizing GPT, users are able to ask questions regarding any unfamiliar or unclear physical parameters at any stage. The Eulerian grid has a default configuration, but a new grid can also be specified by setting the grid size and computational domain. With the geometry and simulation parameters specified, the framework automatically generates the necessary configuration files for the CFD solver. In particular, it writes out a Para.txt file containing the solver settings and global parameters, and a Bc Ic.txt file specifying the boundary and initial conditions as well as the Eulerian grid file. Once all the necessary files are properly prepared, the solver proceeds to simulate the flow around the object using the IB method with the given mesh and parameters. A detailed description of the IB method is

provided in Sec. III. From an implementation perspective, the agent maintains dialogue with the user until it generates the tag **###OUTT###**, signaling that all necessary information has been collected. Upon detecting this tag, the program parses the solver parameters, launches the CFD solver, and transitions to the Postprocessing Agent.

The Postprocessing Agent is used to visualize the simulation results and extract quantitative metrics for analysis. Initially, the agent is configured via SYSTEM PROMPT 3 to act as a postprocessing analyst. Postprocessing is executed through a collaborative workflow where the agent interprets user requests to configure and run specialized scripts. The user can ask for various visualizations or statistics, and each request is mapped to a specific command tag generated by the model. This framework supports a range of automated analyses, including:

- **###POST1###:** Time-series Plots. Extracts and plots time-dependent values from specified spatial points to monitor transient behavior at key locations.
- **###POST2###:** Two-dimensional (2D) Contour Plots. Renders scalar or vector fields on 2D planar slices extracted from the 3D domain (e.g.,  $x$ - $y$  or  $y$ - $z$  planes) to visualize spatial variations.
- **###POST3###:** 3D Iso-surfaces. Generates volumetric renderings of a scalar field's iso-surfaces, allowing for the visualization of coherent flow structures in three dimensions.
- **###POST4###:** Max or Average Coefficients. Computes integral quantities on selected surfaces and reports key metrics, such as time-averaged or peak coefficients ( $C_l$ ,  $C_d$ ), based on the user's query.

The strict tag grammar enables deterministic machine parsing while keeping human dialogue natural and uncluttered. All agent

prompts use a hierarchical, slot-filling design with three elements: a system prompt that states the fluid dynamics objective defines a single trigger condition, enumerates required slots, and forbids redundant loops; command blocks that provide validated tags showing correct extraction and confirmation patterns; and a self-check that verifies slot completeness and input plausibility before emitting any block, otherwise asking exactly one concise follow-up question. This structured approach minimizes errors while enabling intuitive interactions. Consequently, CFDAgent lowers the entry barrier for nonexpert users while preserving the flexibility demanded by seasoned CFD practitioners.

## B. Relevant technologies

### 1. Large language models

LLMs such as GPT-4 series<sup>4</sup> have demonstrated strong zero-shot reasoning and multimodal understanding capabilities. In the context of CFD workflows, these strengths translate into (i) robust intent recognition from noisy user queries, (ii) automatic extraction and validation of simulation parameters, and (iii) context-aware tutoring for novice users. Unlike traditional rule-based chatbots, LLMs can generalize to unforeseen phrasing while enforcing domain-specific constraints through carefully designed prompts.

To ensure reliable parsing, strictly formatted command blocks can be used in a slot-filling structure for the system message and self-check, with automatic prompts for any missing slots. After benchmarking several candidates on manually curated conversational test cases, we selected GPT-4o and retained GPT-4o-mini as a fallback. GPT-4o-mini is more cost-effective than GPT-4o, maintaining a pass@1 accuracy of  $\geq 91\%$  on our task-oriented test suite. Switching the GPT model only requires editing a single line in the Streamlit service, minimizing deployment friction. A detailed analysis of our benchmark results is provided in [Appendix B](#).

### 2. 3D geometry generation based on Point-E

Within the CFDAgent framework, the preprocessing agent leverages Point-E<sup>32</sup> as its core mechanism for autonomously generating 3D geometries from either natural language prompts or 2D reference images. This capability is pivotal for enabling an end-to-end, language-guided simulation workflow, as it allows users to define complex objects without requiring preexisting CAD models or specialized 3D modeling skills. Point-E is not a monolithic model but a sophisticated, multi-stage generative pipeline that decomposes the challenging task of text/image-to-3D generation into two sequential, diffusion-based

processes. This design prioritizes computational efficiency and practical deployability, enabling the generation of a 3D point cloud in approximately 1–2 min on a single GPU. The complete Point-E pipeline, as illustrated in [Fig. 2](#), proceeds in three stages.

Before delving into the stages of Point-E, it is essential to understand the foundational generative framework upon which it is built: the Denoising Diffusion Probabilistic Model (DDPM).<sup>51</sup> DDPMs are a class of latent variable models inspired by non-equilibrium thermodynamics. They define a generative process by learning to reverse a gradual, Markovian noising process that transforms data into pure noise. Given a clean data sample  $\mathbf{x}_0$  (e.g., an image or point cloud) drawn from the true data distribution  $q(\mathbf{x}_0)$ , the forward process adds Gaussian noise over  $T$  discrete timesteps. This is defined as a fixed Markov chain

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1 - \beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}), \quad (1)$$

where  $\beta_t \in (0, 1)$  is a small, predefined variance schedule that controls the amount of noise added at each step  $t$ . The schedule is typically chosen such that by the final step  $T$ ,  $\mathbf{x}_T$  is nearly an isotropic Gaussian, i.e.,  $q(\mathbf{x}_T|\mathbf{x}_0) \approx \mathcal{N}(\mathbf{0}, \mathbf{I})$ . A key property of this process is that any noisy sample  $\mathbf{x}_t$  can be sampled directly from  $\mathbf{x}_0$  without running the entire chain,

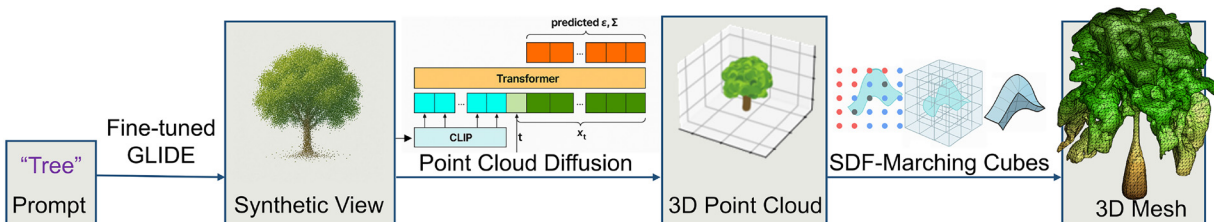
$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon, \quad \text{where } \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \quad (2)$$

Here,  $\alpha_t = 1 - \beta_t$  and  $\bar{\alpha}_t = \prod_{s=1}^t \alpha_s$ .

The first stage generates a synthetic 2D image that serves as an intermediate representation. This is achieved using a variant of the GLIDE model,<sup>52</sup> which is a powerful text-conditional diffusion probabilistic model built upon the DDPM framework described earlier. Crucially, GLIDE extends the standard DDPM to become a conditional diffusion model. It generates an image  $\mathbf{I}$  conditioned on a text prompt by modifying the reverse process  $p_\theta$  to be dependent on the text embedding. A key innovation in GLIDE is the use of classifier-free guidance to dramatically improve the alignment between the generated image and the text prompt. During training, the text conditioning is randomly dropped 10% of the time, replacing it with an unconditional token  $\emptyset$ . This allows the model to learn both a conditional distribution  $\epsilon_\theta(\mathbf{x}_t|\text{text})$  and an unconditional distribution  $\epsilon_\theta(\mathbf{x}_t|\emptyset)$ . During inference, the model's noise prediction is guided by extrapolating away from the unconditional prediction toward the conditional prediction,

$$\hat{\epsilon}_\theta(\mathbf{x}_t|\text{text}) = \epsilon_\theta(\mathbf{x}_t|\emptyset) + s(\epsilon_\theta(\mathbf{x}_t|\text{text}) - \epsilon_\theta(\mathbf{x}_t|\emptyset)), \quad (3)$$

where  $s \geq 1$  is the guidance scale. A higher  $s$  increases the influence of the text prompt, leading to outputs that more closely match text but



**FIG. 2.** Text-to-3D mesh generation pipeline. A natural language prompt is processed by a fine-tuned GLIDE model to generate a synthetic image, which is then converted into a 3D point cloud using point cloud diffusion with CLIP and a transformer. The resulting point cloud is transformed into a 3D mesh using SDF estimation and marching cubes.

potentially at the cost of sample diversity. For the Point-E's GLIDE stage, a guidance scale of  $s = 3.0$  is typically used for optimal results.<sup>32</sup> The mean of the reverse process is then parameterized using this guided noise prediction,

$$\mu_\theta(\mathbf{x}_t, t, \text{text}) = \frac{1}{\sqrt{\alpha_t}} \left( \mathbf{x}_t - \frac{\beta_t}{\sqrt{1 - \alpha_t}} \hat{\epsilon}_\theta(\mathbf{x}_t, t, \text{text}) \right). \quad (4)$$

The GLIDE model employs a U-Net architecture, which is specifically designed for processing grid-structured data like images. This is a critical distinction from the subsequent stage. For its use in Point-E, the GLIDE model is fine-tuned on a dataset of 3D object renderings. This fine-tuning biases the model to generate images that are plausible, single-view projections of 3D objects, rather than arbitrary 2D compositions. This ensures that the synthetic views it produces are suitable for the subsequent 3D reconstruction stage. During inference, a special token is appended to the user's text prompt to signal to GLIDE that it should generate a 3D-style render. If the user provides a 2D image instead of text, this first stage is bypassed, and the provided image is used directly as the conditioning input for the next stage.

The second stage takes the synthetic (or user-provided) 2D image  $\mathbf{I}$  and generates a corresponding 3D point cloud  $\mathcal{P} = \{(\mathbf{p}_i, \mathbf{c}_i)\}_{i=1}^N$ , where  $\mathbf{p}_i \in \mathbb{R}^3$  represents the 3D coordinates of a point and  $\mathbf{c}_i \in [0, 1]^3$  represents its RGB color. This stage employs a Transformer-based conditional diffusion model. Unlike GLIDE (Stage 1), which uses a U-Net, Point-E's point cloud model leverages a permutation-invariant Transformer. This choice is deliberate: since a point cloud is an unordered set of points, a Transformer that treats its input as a set of tokens without inherent positional meaning is a more natural and inductive-bias-aligned architecture than a U-Net, which is designed for grid-structured data. The conditioning image  $\mathbf{I}$  is processed by a frozen, pretrained CLIP model. Crucially, Point-E uses the spatial feature grid from CLIP, which is projected into a sequence of tokens and prepended to the Transformer's input context alongside the tokens for the noised point cloud  $\mathbf{x}_t$  and the time step  $t$ . This rich, spatial conditioning allows the model to understand the structure and semantics of the image far better than using a single global CLIP embedding. To ensure the generated point cloud  $\mathcal{P}$  is semantically aligned with the conditioning image  $\mathbf{I}$ , Point-E's point cloud diffusion model employs classifier-free guidance, following the same mathematical principle introduced in GLIDE [Eq. (3)]. During training, the image conditioning is randomly dropped 10% of the time. During inference, the model uses the same guidance formula, replacing text with  $\mathbf{I}$ , to sharpen the alignment between the point cloud and the image. The model is trained to predict both the mean  $\mu_\theta$  and the variance  $\Sigma_\theta$  of the reverse diffusion process  $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{I}) = \mathcal{N}(\mu_\theta(\mathbf{x}_t, t, \mathbf{I}), \Sigma_\theta(\mathbf{x}_t, t, \mathbf{I}))$ . Following the improvements detailed in Ref. 24, the variance  $\Sigma_\theta$  is not fixed but is learned by the model. It is parameterized as an interpolation between the upper and lower bounds  $\beta_t$  and  $\tilde{\beta}_t$ ,

$$\Sigma_\theta(\mathbf{x}_t, t) = \exp(v \log \beta_t + (1 - v) \log \tilde{\beta}_t), \quad (5)$$

where  $v$  is a vector output by the model for each dimension. This parameterization stabilizes training and improves sample quality.

For use in CFD simulations, the generated point cloud  $\mathcal{P}$  must be converted into a watertight surface mesh. Point-E achieves this

through an implicit surface representation. A separate neural network  $\phi_\psi: \mathbb{R}^3 \rightarrow \mathbb{R}$  is trained to regress a Signed Distance Function (SDF) from the point cloud. For any query point  $\mathbf{x}$  in 3D space,  $\phi_\psi(\mathbf{x})$  predicts its signed distance to the object's surface (negative outside, positive inside). The object's surface is defined as the zero-level set  $\mathcal{S} = \{\mathbf{x} \in \mathbb{R}^3 | \phi_\psi(\mathbf{x}) = 0\}$ . A standard Marching Cubes algorithm<sup>53</sup> is then applied to this implicit field to extract a triangular surface mesh. Finally, Laplacian smoothing is applied to the extracted mesh to reduce any voxelization artifacts and produce a smoother surface suitable for numerical simulation.

### 3. Immersed boundary method for incompressible flows

The IB method offers an efficient framework for simulating incompressible flows around bodies with complex geometries on non-conformal, typically Cartesian, grids. This is achieved by introducing a forcing into the Navier–Stokes equations to enforce the boundary conditions at the fluid–body interface. In this formulation, the fluid domain is discretized on a fixed Eulerian grid, while the immersed boundaries are represented by Lagrangian markers that are either stationary or in motion.

For the momentum, the Navier–Stokes equations with IB forcing take the form:

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla p + \frac{1}{Re} \nabla^2 \mathbf{u} + \mathbf{f}, \quad (6)$$

$$\nabla \cdot \mathbf{u} = 0, \quad (7)$$

where  $\mathbf{u}(\mathbf{x}, t)$  is the Eulerian velocity field,  $p(\mathbf{x}, t)$  is the pressure, and  $Re$  is the Reynolds number. The term  $\mathbf{f}$  represents the body force exerted on the fluid by the immersed surface. Within the IB framework, one defines a Lagrangian force  $\mathbf{F}(\mathbf{X}, t)$  on the fluid–body interface  $\mathcal{S}$  (with Lagrangian coordinate  $\mathbf{X}$ ). This force is spread to the Eulerian grid via a convolution with a regularized delta function  $\delta_h$ ,

$$\mathbf{f}(\mathbf{x}, t) = \int_{\mathbf{X} \in \mathcal{S}} \mathbf{F}(\mathbf{X}, t) \delta_h(\mathbf{x} - \mathbf{X}) d\mathbf{X}. \quad (8)$$

Here,  $\delta_h$  ensures a smooth transfer of force from the Lagrangian markers to the Eulerian grid. The delta function  $\delta_h$  used for force interpolation is a piecewise function ensuring smoothness. In 3D cases, it is defined as

$$\delta_h(\mathbf{x} - \mathbf{X}) = \frac{1}{h^3} \phi\left(\frac{x - X}{h}\right) \phi\left(\frac{y - Y}{h}\right) \phi\left(\frac{z - Z}{h}\right), \quad (9)$$

where  $\phi(r)$  is the kernel function

$$\phi(r) = \begin{cases} \frac{1}{8} (3 - 2|r| + \sqrt{1 + 4|r| - 4r^2}), & |r| \leq 1, \\ \frac{1}{8} (5 - 2|r| - \sqrt{-7 + 12|r| - 4r^2}), & 1 \leq |r| \leq 2, \\ 0, & |r| \geq 2. \end{cases} \quad (10)$$

This kernel function ensures second-order accuracy in force interpolation and velocity reconstruction.

Conversely, the fluid velocity at the Lagrangian markers is obtained by interpolating the Eulerian field,

$$\mathbf{U}^*(\mathbf{X}, t) = \int_{\mathbf{x} \in \Omega} \mathbf{u}(\mathbf{x}, t) \delta_h(\mathbf{x} - \mathbf{X}) d\mathbf{x}. \quad (11)$$

The Lagrangian force  $\mathbf{F}$  is then chosen so as to enforce the no-slip and no-penetration conditions. In practice, one computes  $\mathbf{F}(\mathbf{X}, t)$  to drive the interpolated velocity  $\mathbf{U}^*(\mathbf{X}, t)$  toward the prescribed boundary velocity  $\mathbf{U}_b(\mathbf{X}, t)$  (for example, via a penalty or direct-forcing formulation), thereby closing the coupling between fluid and body.

The spatial discretization is performed using a second-order finite volume scheme, and temporal integration is carried out with a three-step, second-order, low-storage Runge–Kutta method. To address large-scale three-dimensional simulations efficiently, the solver employs parallelization based on domain decomposition and MPI protocols. Flow variables are distributed across worker processors, whereas the IB equations are assembled and solved centrally on the root processor using a gather-scatter communication strategy, thus demonstrating excellent scalability. More numerical details and validations can be found in our previous publications.<sup>45</sup>

### III. RESULTS AND DISCUSSION

#### A. CFD<sub>Agent</sub> flow simulation around canonical geometries: Sphere, cube, and cylinder

By leveraging the capabilities of the proposed LLM-driven CFD framework, we conducted three-dimensional numerical simulations of unsteady flow around three canonical geometries—a sphere, a cube, and a cylinder—at representative Reynolds numbers to validate the framework's performance and accuracy.

##### 1. Flow around a sphere

The entire simulation workflow is guided by CFD<sub>Agent</sub> through an interactive, dialogue-based process. As shown in Table I, when the user expresses interest in CFD, CFD<sub>Agent</sub> autonomously orchestrates its core agents to initiate the procedure. The Preprocessing Agent requests the geometry and Reynolds number, supporting three distinct input modes: (1) direct generation from a natural language prompt (e.g., sphere), (2) reconstruction from an input image of the object, and (3) direct import of a pre-generated surface mesh. All three modes ultimately produce a Lagrangian surface mesh representing the object. Subsequently, the Solver Agent prompts the user for detailed numerical simulation settings and initiates the solver.

Unless otherwise specified, all simulations in this section utilize the default configuration settings. For external flow cases studied in this work, except the infinitely long cylinder, we employ a default and consistent set of boundary conditions: uniform velocity  $u = 1$  with freestream conditions at the inlet, zero-gradient pressure boundary condition at the outlet, far-field conditions at the outer computational domain boundaries, and no-slip wall condition on the object surface. As illustrated in Fig. 3, the computational domain extends  $[-4D, 8D] \times [-5D, 5D] \times [-5D, 5D]$  in the  $x$ ,  $y$ , and  $z$  directions, respectively, where  $D$  is the characteristic length, here is the sphere diameter, with a core refinement region centered at the origin extending over a  $2D$  area. The Eulerian grid is locally refined near the sphere surface, with a minimum size of  $h = D/40$ , selected to match the average Lagrangian mesh size. The total number of Eulerian grid cells is approximately  $3.6 \times 10^6$ , and the surface consists of approximately 18 000 Lagrangian points. Grid convergence was rigorously validated through multiple tests, confirming that variations in grid resolution

**TABLE I.** An example interaction between the user and CFD<sub>Agent</sub> during a simulation. The detailed dialogue example can be found in Appendix A.

User Prompt	CFD <sub>Agent</sub>
Compute the flow around a sphere at $Re = 300$	We have the object as sphere and the Reynolds number as “300.” Please confirm these before I initiate the simulation.
Yes, I confirm	Starting point cloud generation for sphere. Please wait... Creating SDF and generating mesh... Please select a wind direction.
[Click a direction] 200 should be enough	What is the end time of the simulation? Great! You've set the end time of the simulation to 200.0. Now, what is the CFL number? (The default is 0.8)
Use default.	You've set the CFL number to 0.8. Now, what is the maximum time step? (The default is 0.1)
I prefer 0.2.	You've set the maximum time step to 0.2. Finally, what is the output interval time? (The default is 1.0)
10	You've set the output interval time to 10.0. <i>Here's a summary of all values.</i> Do you confirm these values?
Yes!	Use the default Euler grid and boundary conditions, or customize them?
Default.	Parameters saved to Para.txt and Bc_Ic.txt. The Euler grid is generated with default settings. Simulation has started!

and domain size have negligible influence on the flow characteristics.<sup>45,46,49,54</sup> If the default Eulerian grid is not used, Agent Solver automatically switches to an interactive Eulerian grid-generation mode for user definition. It queries the user for each of the Eulerian grid parameters described earlier, confirms the selections, and then executes the Python script to build a new grid.

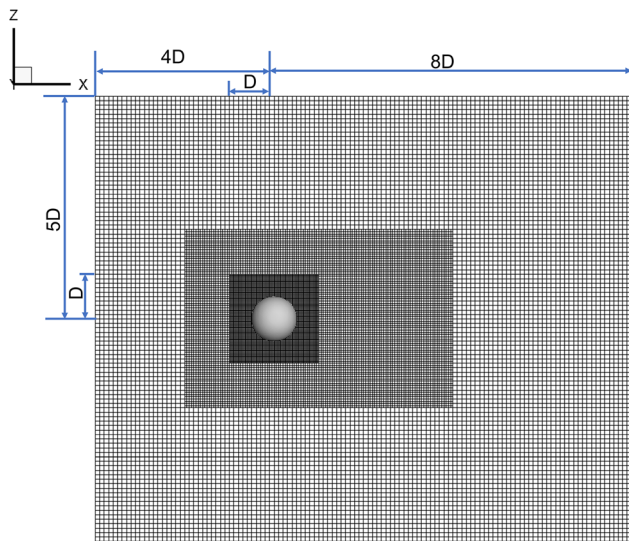
The Postprocessing Agent generates figures to visualize the simulation results as specified by the user prompt. In the postprocessing step, we compute the dimensionless drag and lift coefficients,  $C_d$  and  $C_l$ , by normalizing the drag ( $D$ ) and lift ( $L$ ) forces by the dynamic pressure,  $q = \frac{1}{2} \rho U^2$ , and a reference area,  $S$ ,

$$C_d = D/(qS), C_l = L/(qS). \quad (12)$$

Here,  $\rho$  is the fluid density and  $U$  is the freestream velocity.

To ensure the reliability of our numerical results, a grid independence study was conducted using three different grid resolutions. Table II presents the drag coefficient  $C_d$  computed with varying grid densities. The results demonstrate that the solution is grid-independent, with the drag coefficient converging for the medium and fine grids. The medium grid with  $3.6 \times 10^6$  cells was therefore selected for subsequent simulations as it provides an optimal balance between computational accuracy and efficiency.





**FIG. 3.** Computational domain and grid configuration showing the  $x$ - $z$  plane view. The domain extends from  $-4D$  to  $8D$  in the  $x$ -direction and symmetrically spans  $\pm 5D$  in both  $y$  and  $z$  directions. A sphere of diameter  $D$  is positioned at the origin with local mesh refinement in the surrounding  $2D \times 2D$  region. Due to symmetry, only a two-dimensional slice is presented for clarity.

At  $Re = 100$ , the flow around a sphere is steady and axisymmetric, with a well-defined separation bubble forming downstream. The lift coefficient  $C_l$  under this condition is nearly zero, consistent with the axisymmetric nature of the flow. Table III provides a comparison of the drag coefficient  $C_d$  from numerical simulations, employing different geometry-inputting approaches as well as previously reported literature values. As shown in Fig. 4, the computed streamlines exhibit reattachment on the symmetry axis, forming a closed separation bubble and a toroidal vortex whose center lies within the bubble. The  $z$ -component of vorticity,  $\omega_z$ , exhibits the expected pair of vortices with equal magnitude and opposite sign, as shown in Fig. 5. Peak magnitudes occur just downstream of separation along the sphere's flanks, with monotonic decay in the far wake, consistent with a steady, axisymmetric regime without vortex shedding. Both results are consistent with the steady regime and match the features shown in Ref. 55.

The comparison indicates good agreement, thus validating the accuracy and reliability of the current approaches.

At  $Re = 300$ , the flow transitions to an unsteady state characterized by pronounced vortex shedding and associated fluctuations in  $C_d$  and  $C_l$ . This represents a fundamental shift from the steady, axisymmetric regime observed at  $Re = 100$  to a fully three-dimensional time-dependent flow pattern. Figure 6 presents instantaneous streamlines

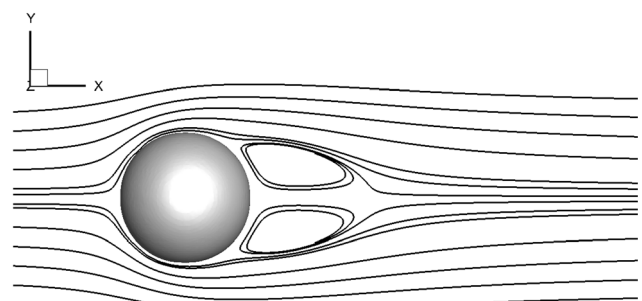
**TABLE II.** Grid independence study for the sphere at  $Re = 100$  and  $Re = 300$ .

Grids	Number of cells	$C_d$ ( $Re = 100$ )	$C_d$ ( $Re = 300$ )
Coarse	$0.45 \times 10^6$	1.09	0.65
Medium	$3.6 \times 10^6$	1.12	0.68
Fine	$28.8 \times 10^6$	1.12	0.68

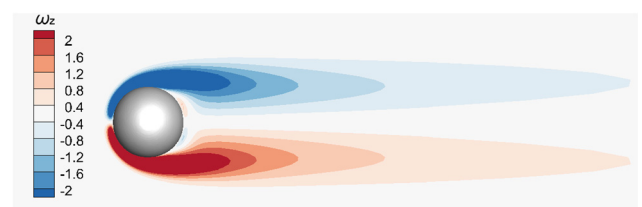
**TABLE III.** Comparison of drag coefficients ( $C_d$ ) of the sphere at  $Re = 100$ .

Case	$C_d$
Prompt-based geometry (sphere)	1.11
Image-based geometry	1.11
Imported mesh geometry	1.12
Johnson and Patel <sup>55</sup>	1.10
Fadlun <i>et al.</i> <sup>37</sup>	1.08

and vorticity in the  $x$ - $z$  plane, revealing the complex wake structure at this Reynolds number. The streamlines highlight the unsteady perturbations that disrupt axisymmetry. At the higher Reynolds number, the flow instability amplifies, leading to periodic shedding of vortices from the sphere's rear. A representative instantaneous vortical structure under this condition is illustrated in Fig. 7, where vortices are visualized and identified based on the  $Q$ -criterion. The resulting vortex shedding induces the time-averaged  $C_d$  slightly above the steady-state value while introducing  $C_l$  fluctuations, as corroborated by the literature. The observed vortex shedding patterns are consistent with those previously documented by Wang and Zhang<sup>45</sup> and Johnson and Patel.<sup>55</sup> Table IV compares the computed time-averaged drag and lift coefficients against literature benchmarks. The results obtained from the various input approaches for the CFD Agent, including both geometry-descriptive (image and text) and mesh-based inputs, all show excellent quantitative agreement with the reference values. This consistency across different methods not only validates the accuracy of each approach but also underscores the overall robustness and reliability of the simulation methodology employed.



**FIG. 4.** Streamlines for flow past a sphere at  $Re = 100$ .



**FIG. 5.** Contours of the  $z$ -component of vorticity  $\omega_z$  for flow past a sphere at  $Re = 100$ .

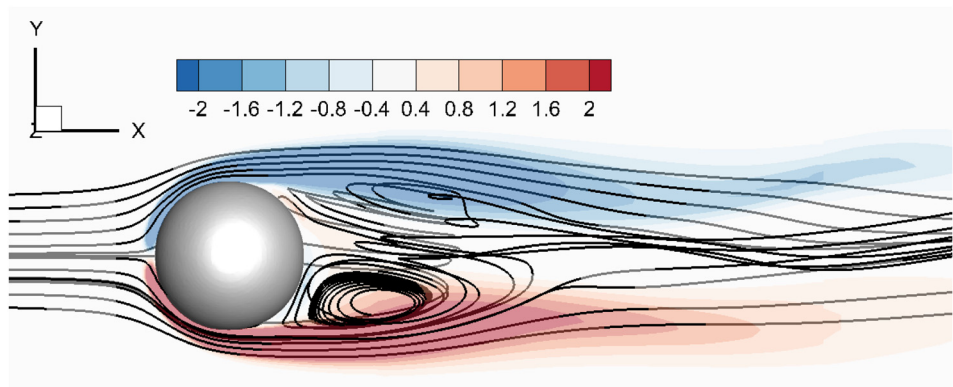


FIG. 6. Instantaneous streamlines and vortex for flow past a sphere at  $Re = 300$ .

2. Flow around a cube

Following the sphere benchmark, we examine the second test case of flow around a cube. Based on our previous findings that different geometry input methods (image, mesh, and prompt) yield modest differences in results, we employ only the prompt-based input “cube” for brevity in this section. All other computational settings remain identical to those used in the sphere simulations.

To ensure the rigor of our numerical approach, we conducted a grid independence study with three different grid resolutions at  $Re = 50$  and  $Re = 300$ . As shown in Table V, the results demonstrate excellent grid convergence, with the drag coefficient remaining virtually unchanged between the medium and fine grids. The computed values show strong agreement with the reference results of Saha,<sup>56</sup> validating both the accuracy of our simulations and the adequacy of the chosen grid resolution. Consequently, the medium grid configuration

with  $3.6 \times 10^6$  cells is adopted for all subsequent simulations involving geometries of similar size and Reynolds number, as it provides an optimal balance between computational efficiency and numerical accuracy.

At  $Re = 300$ , the flow exhibits pronounced three-dimensional characteristics with complex wake dynamics. Figure 8 illustrates the instantaneous three-dimensional vortical structures, revealing the intricate wake topology characterized by strong three-dimensional vortex shedding patterns. The two-dimensional visualization in Fig. 9 presents the z-component of vorticity  $\omega_z$  in the wake region, demonstrating the asymmetric vortex shedding with alternating positive and negative vorticity regions. These flow features, including the formation of counter-rotating vortex pairs and the subsequent downstream evolution of the wake structure, are consistent with the unsteady flow characteristics documented by Saha.<sup>56</sup> The observed flow patterns validate the capability of our framework to accurately capture complex bluff body flows across different geometries.

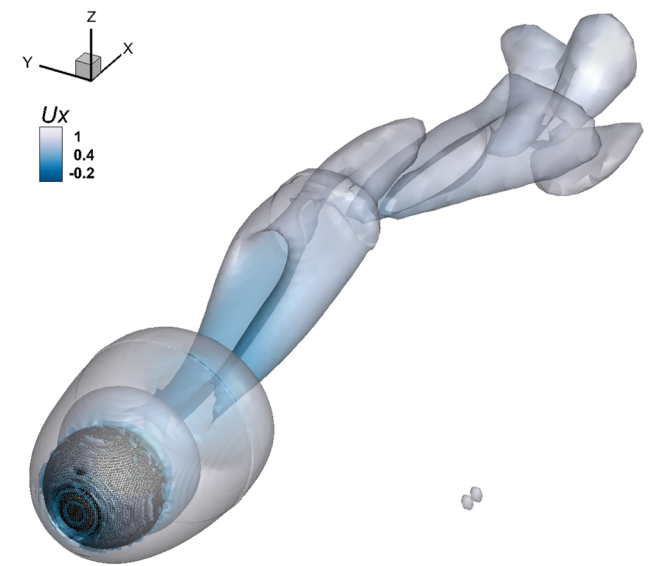


FIG. 7. Instantaneous vortical structure for flow past a sphere at  $Re = 300$ . The vortical structures are visualized using iso-surfaces of the Q-criterion, highlighting regions with a high ratio of vorticity to strain. The iso-surfaces are colored by the streamwise velocity, ranging from  $-0.2$  to  $1$ , with colors transitioning from blue to white, illustrating the variation of the flow speed around the vortex structures.

TABLE IV. Comparison of drag and lift coefficients ( $C_d$  and  $C_l$ ) of the sphere at  $Re = 300$ .

Case	$C_d$	$C_l$
Prompt-based geometry (sphere)	0.68	0.065
Image-based geometry	0.68	0.065
Imported mesh geometry	0.68	0.071
Johnson and Patel <sup>55</sup>	0.66	0.069
Kim <i>et al.</i> <sup>39</sup>	0.66	0.067

TABLE V. Grid independence study for the cube at  $Re = 50$  and  $Re = 300$ .

Grids	Number of cells	$C_d$ ( $Re = 50$ )	$C_d$ ( $Re = 300$ )
Coarse	$0.45 \times 10^6$	1.71	0.76
Medium	$3.6 \times 10^6$	1.80	0.81
Fine	$28.8 \times 10^6$	1.80	0.81
Saha <sup>56</sup>	...	1.78	0.80



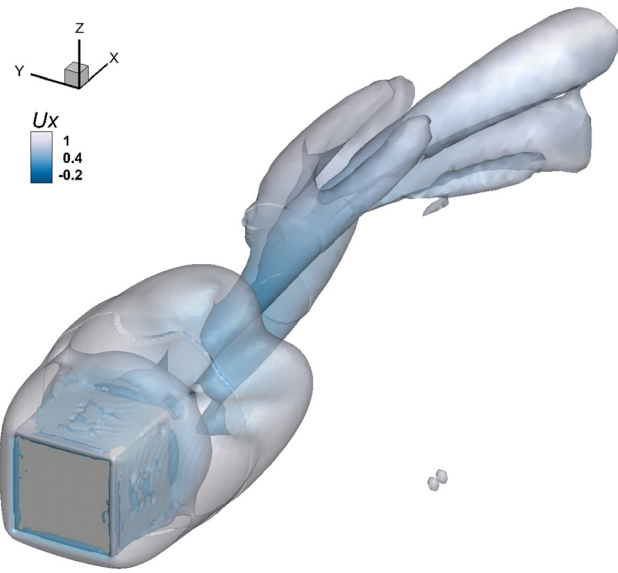


FIG. 8. Instantaneous vortical structure for flow past a cube at  $Re = 300$ .

### 3. Flow around an infinite circular cylinder

To further validate the framework's accuracy, we apply it to a canonical problem in fluid dynamics: flow around an infinite circular cylinder. This scenario is crucial for benchmarking numerical methods due to its well-documented flow phenomena, such as vortex shedding, which are highly sensitive to numerical parameters and boundary conditions. The simulation is designed to represent a cylinder that is infinite in the spanwise ( $z$ -direction). In the  $z$ -direction, the grids are extruded with uniform refinement extending along the span. The other parts of the grids are consistent with the strategies adopted for other benchmarks. The boundary conditions consist of a uniform inlet velocity, a zero-pressure outlet, symmetry conditions on the lateral boundaries to approximate infinite domain extent, and a no-slip condition on the cylinder surface. Following the simulation, the computed flow field in the  $z = 0$  ( $xy$ -plane) is extracted and analyzed. This reference plane allows for direct comparison with extensive results from the literature. A grid independence study was conducted; and the current results are consistent with previous findings, as listed in Table VI.

At  $Re = 40$ , the flow remains steady with a fully developed recirculation zone formed in the wake of the cylinder. This recirculation

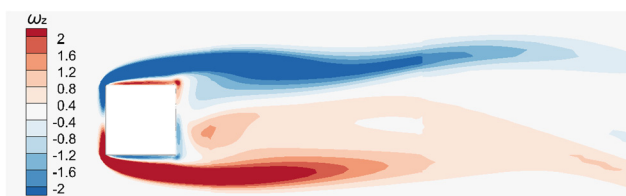


FIG. 9. Contours of the  $z$ -component of vorticity  $\omega_z$  for flow past a cube at  $Re = 300$ .

region can be described by several characteristic parameters: the streamwise extent  $l$ , the horizontal distance  $a$  from the cylinder's trailing edge to the vortex center, and the vertical spacing  $b$  between the twin vortex centers, all of which are illustrated in the streamline pattern shown in Fig. 10. Table VII presents the computed results from this study alongside corresponding values reported in previous literature.

At  $Re = 200$ , the flow is characterized by alternating vortex shedding from the cylinder. The instantaneous vorticity contours at  $Re = 200$  are shown in Fig. 11. The comparisons of the present results (drag and lift coefficients and Strouhal number) with those from the literature are summarized in Table VIII. The numerical results from the present simulations are in good agreement with those from the literature.

### B. Zero-shot simulation of complex flows over arbitrary real-world objects

The Preprocessing Agent of the proposed LLM-driven CFD framework is capable of generating complex geometries based solely on natural language prompts, without requiring any preexisting CAD models or case-specific training. This capability allows for the simulation of arbitrary objects, significantly enhancing the versatility and applicability of the framework.

We demonstrate the framework's generalization capabilities by examining simulations prompted by diverse textual inputs describing various complex geometries. Specifically, eight different geometries—an image, “cat,” “dog,” “motorcycle,” “pot,” “side mirror,” “tower,” and “tree”—are autonomously generated and simulated at  $Re = 300$ . The Preprocessing Agent automatically generates the corresponding geometric models and computational meshes using the integrated image/text-to-3D model and mesh conversion methods. All simulations employ consistent settings to ensure comparability across scenarios, using the same computational parameters and boundary conditions as those established in the benchmark sphere and cube flow cases presented earlier.

Figure 12 presents comprehensive postprocessing results for each geometry, systematically visualizing velocity fields, streamlines, and three-dimensional vortex structures through natural language prompts to the Postprocessing Agent. The velocity distributions on the  $x$ - $z$  plane ( $y = 0$ ) reveal distinct flow characteristics for each geometry. Sharp edges and corners, particularly evident in the pot and tower cases, generate localized flow acceleration and separation regions. The motorcycle geometry exhibits complex flow interactions between its multiple components (wheels, body, handlebars), while organic shapes like cat and dog show smoother velocity transitions around curved surfaces.

TABLE VI. Grid independence study for the cylinder at  $Re = 40$  and  $Re = 200$ .

Grids	Number of cells	$C_d$ ( $Re = 40$ )	$C_d$ ( $Re = 200$ )
Coarse	$0.5 \times 10^6$	1.48	1.25
Medium	$4.0 \times 10^6$	1.54	1.32
Fine	$32.0 \times 10^6$	1.54	1.32
Taira and Colonius <sup>43</sup>	...	1.54	1.35
Linnick and Fasel <sup>57</sup>	...	1.54	1.34

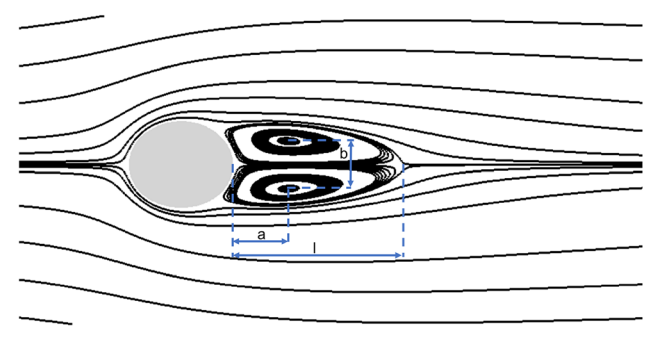


FIG. 10. Streamlines for flow past a cylinder at  $Re = 40$ .

TABLE VII. Characteristic dimensions of the recirculation zone for flow past a cylinder at  $Re = 40$ .

Case	$l/D$	$a/D$	$b/D$
Present	2.36	0.72	0.60
Linnick and Fasel <sup>57</sup>	2.28	0.72	0.60
Taira and Colonius <sup>43</sup>	2.33	0.72	0.60

The streamline visualizations highlight markedly different wake structures across geometries. The pot produces a highly three-dimensional, dispersed wake with multiple recirculation zones, reflecting flow separation from its cylindrical body and handle. In contrast, the motorcycle generates a small wake despite its geometric complexity, with distinct vortex shedding from the wheels and body. The tree geometry creates an intricate wake pattern due to flow interaction with its branching structure, showing how the framework handles highly irregular shapes. These visualizations demonstrate the framework’s ability to capture complex three-dimensional flow phenomena across drastically different geometric configurations.

Beyond traditional CFD visualization, the framework leverages GPT-4o’s multimodal synthesis capabilities to generate context-aware artistic renderings. For instance, the cat and dog cases show wind effects on fur, the motorcycle visualization captures aerodynamic flow lines during motion, and the tree rendering depicts wind-driven leaf dispersion—all inspired by the computed flow fields. These synthesized images bridge the gap between technical CFD results and intuitive physical understanding. It should be noted that these synthesized images are artistic interpretations based on the CFD results rather than rigorous physical visualizations.

Collectively, these results underscore the robustness, versatility, and generalizability of the zero-shot CFDAgent framework in autonomously simulating and visualizing complex flow fields around arbitrary geometries defined solely by natural language inputs.

IV. CONCLUSIONS

In this study, we introduce CFDAgent, a zero-shot, natural-language-driven multi-agent framework that autonomously executes



FIG. 11. Vorticity for flow past a cylinder at  $Re = 200$ .

TABLE VIII. Comparison of numerical results for flow around a circular cylinder at  $Re = 200$ .

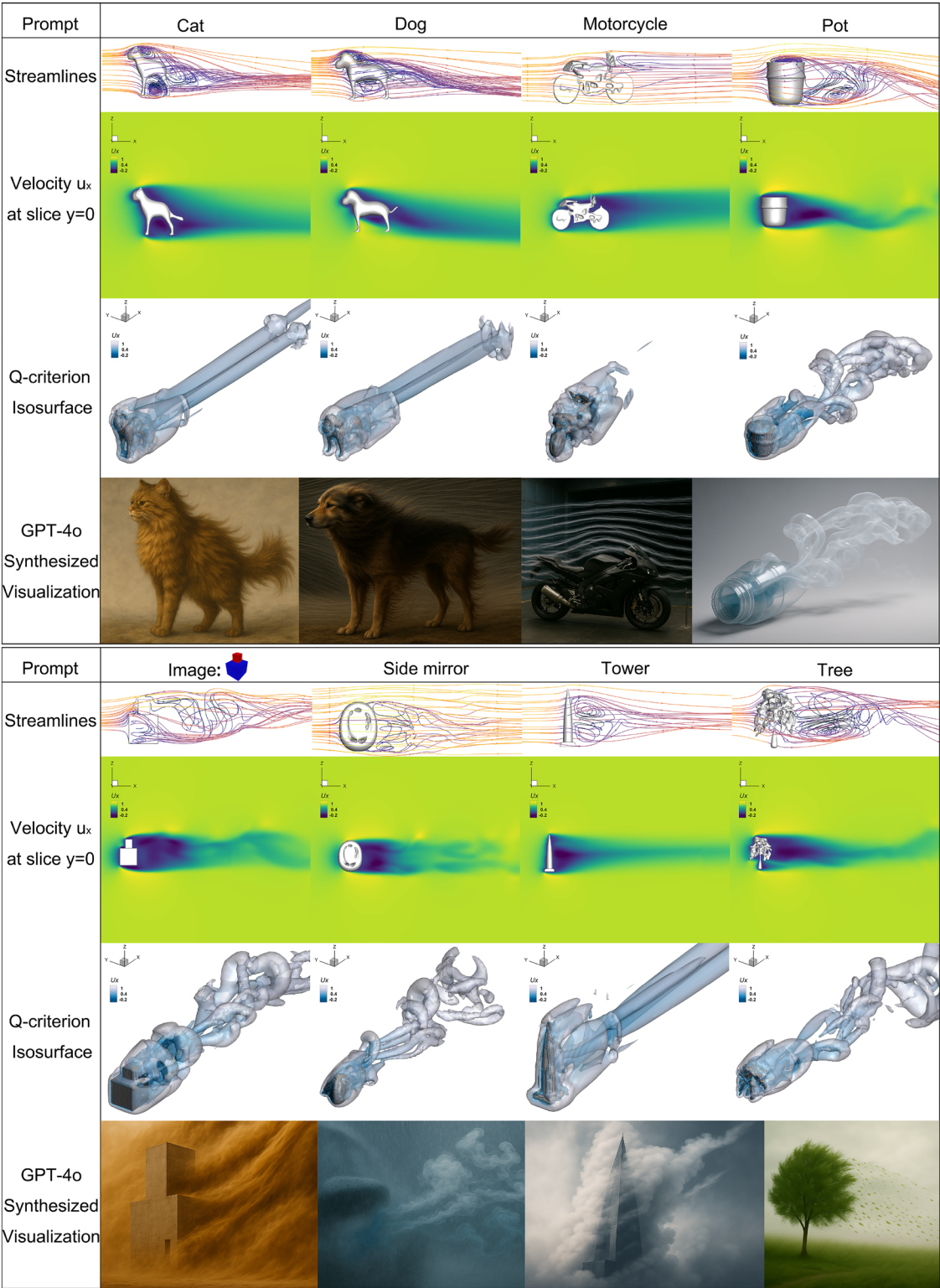
Case	$C_d$	$C_l$	$St$
Present	1.32	0.69	0.198
Linnick and Fasel <sup>57</sup>	1.34	0.69	0.197
Taira and Colonius <sup>43</sup>	1.35	0.68	0.196

complex CFD simulations without requiring domain-specific training or manual preprocessing. By integrating advanced LLM capabilities, a hybrid text-to-3D diffusion model, and an immersed boundary solver, CFDAgent enables robust geometry generation and accurate flow simulation for arbitrary, user-defined objects.

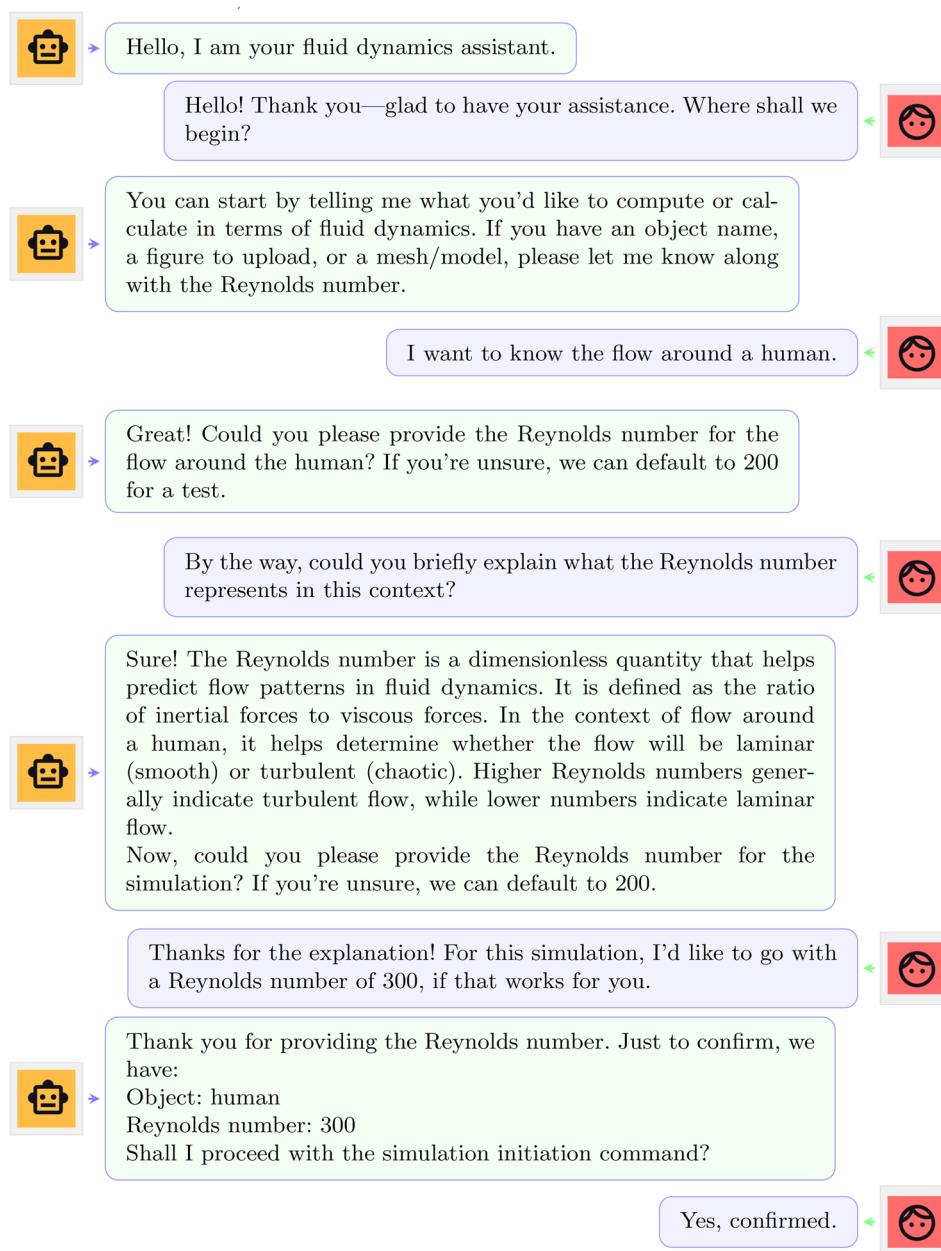
The framework demonstrates remarkable versatility through its multimodal input capabilities, accepting natural language descriptions, 2D images, and 3D geometry meshes while providing diverse outputs including quantitative plots, flow visualizations, and synthesized photorealistic images. Validation against canonical benchmark cases—flow past spheres, cubes, and cylinders—shows excellent agreement with established literature data. CFDAgent successfully simulates flows around complex real-world geometries ranging from everyday objects (pots and motorcycles) to organic shapes (animals and trees) and engineering structures (towers and vehicles), all through simple text prompts. This zero-shot capability represents a fundamental shift from traditional CFD workflows, eliminating the need for extensive preprocessing, mesh generation expertise, and case-specific setup.

Despite these advances, current limitations include geometric fidelity constraints from the Point-E model, computational costs at high Reynolds numbers, and restrictions to static boundary conditions. Future development should focus on enhancing geometry generation quality through advanced diffusion models, implementing adaptive grid refinement strategies, and extending capabilities to handle moving boundaries and multiphase flows.

By substantially lowering the barriers to CFD analysis, CFDAgent opens new possibilities for rapid design exploration, educational applications, and preliminary engineering assessments, marking a significant step toward democratizing computational fluid dynamics through artificial intelligence.



**FIG. 12.** CFDAgent postprocessing results demonstrating generalization for different prompts: Streamlines, velocity distributions, vortex structures (Q-criterion iso-surface), and GPT-4o synthesized visualizations.



## ACKNOWLEDGMENTS

This work was supported by the NSFC Excellence Research Group Program for “Multiscale Problems in Nonlinear Mechanics” (No. 12588201), the National Natural Science Foundation of China (Nos. 12425207 and 92252203), and the Chinese Academy of Sciences Project for Young Scientists in Basic Research (Grant No. YSBR-087). The work was also supported by the OCTAVE project (Grant No. P2024-01011) in the VINNOVA Strategic Vehicle Research and Innovation Program from the Swedish Energy Agency.

## AUTHOR DECLARATIONS

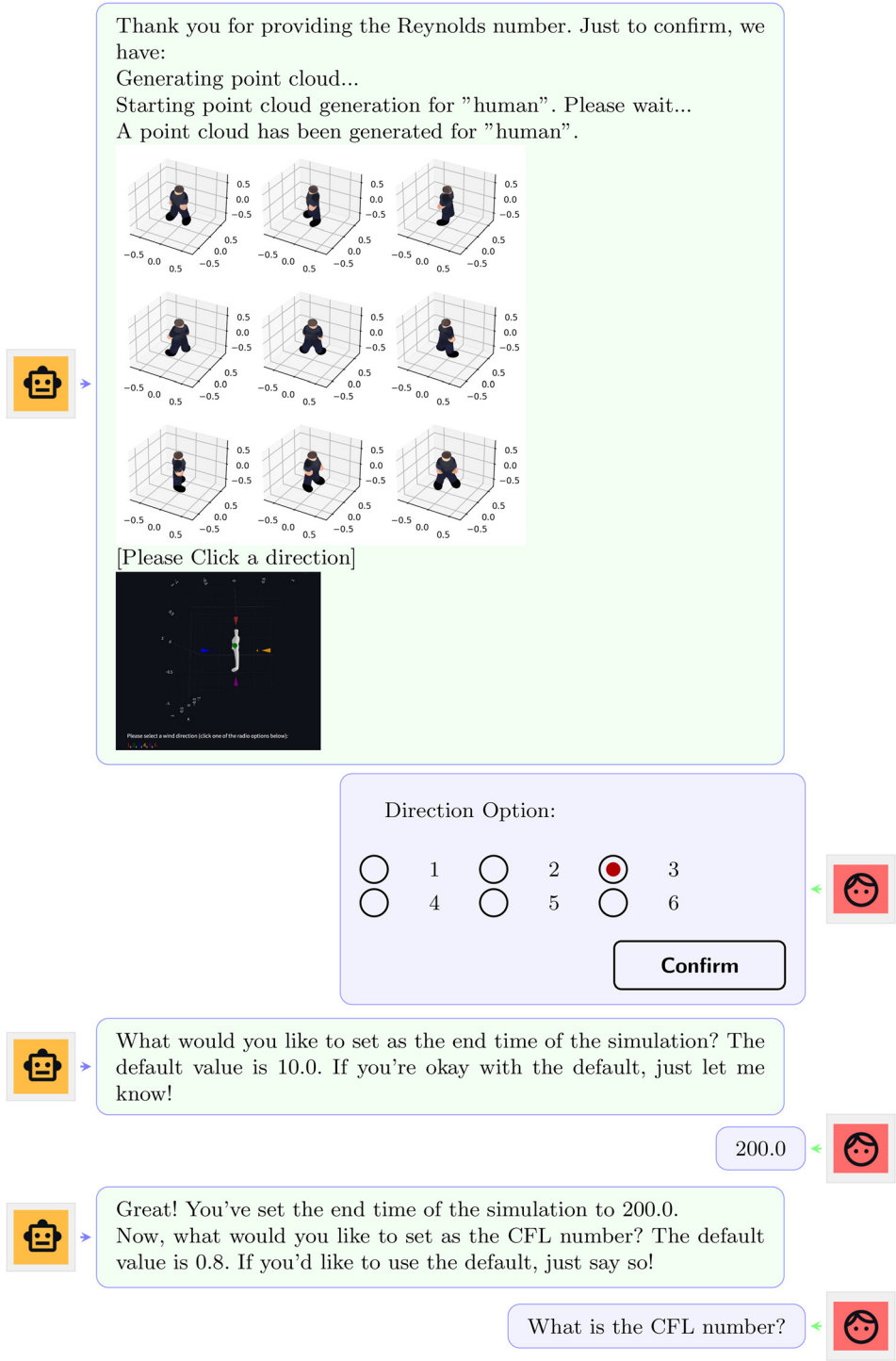
### Conflict of Interest

The authors have no conflicts to disclose.

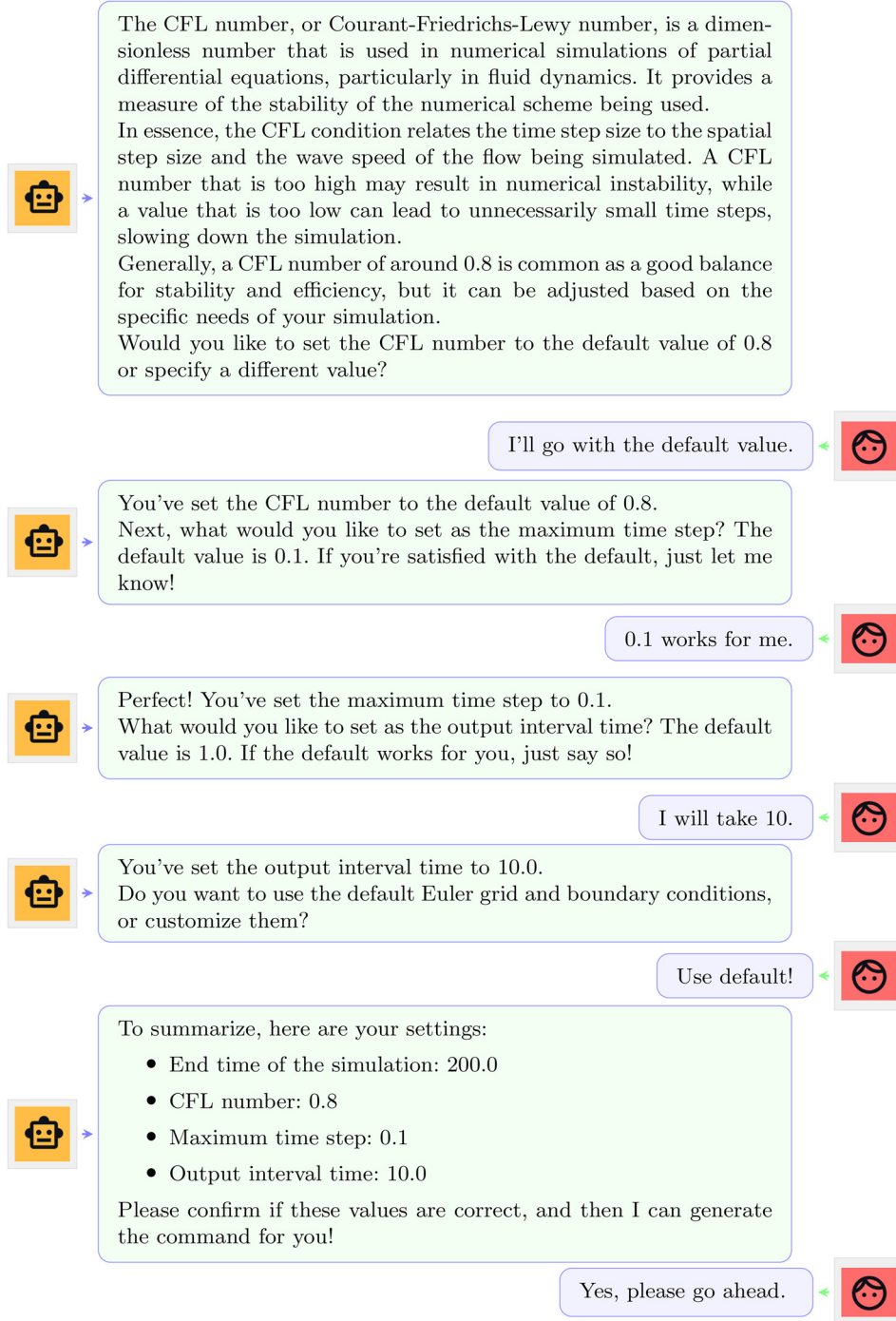
### Author Contributions

**Zhaoyue Xu:** Conceptualization (equal); Data curation (equal); Formal analysis (equal); Investigation (equal); Methodology (equal); Software (equal); Validation (equal); Visualization (equal); Writing –






original draft (equal); Writing – review & editing (equal). **Long Wang:** Validation (equal); Writing – original draft (equal); Writing – review & editing (equal). **Chunyu Wang:** Software (equal); Writing – original draft (equal); Writing – review & editing (equal). **Yixin Chen:** Software (equal); Writing – review & editing (equal). **Qingyong Luo:** Formal analysis (equal); Methodology (equal); Writing – original draft (equal); Writing – review & editing (equal). **Hua-Dong Yao:** Conceptualization (equal); Funding acquisition (equal); Methodology



(equal); Project administration (equal); Resources (equal); Supervision (equal); Validation (equal); Visualization (equal); Writing – original draft (equal); Writing – review & editing (equal). **Shizhao Wang:** Formal analysis (equal); Investigation (equal); Methodology (equal);

Resources (equal); Software (equal); Supervision (equal); Validation (equal). **Guowei He:** Conceptualization (equal); Funding acquisition (equal); Methodology (equal); Project administration (equal); Resources (equal); Supervision (equal); Validation (equal).






Current Computational Control Parameter Configuration


Parameter	Value
START_TIME	0.0
END_TIME	200.0
CFL	0.8
MAX_DT	0.1
print_freq	10.0

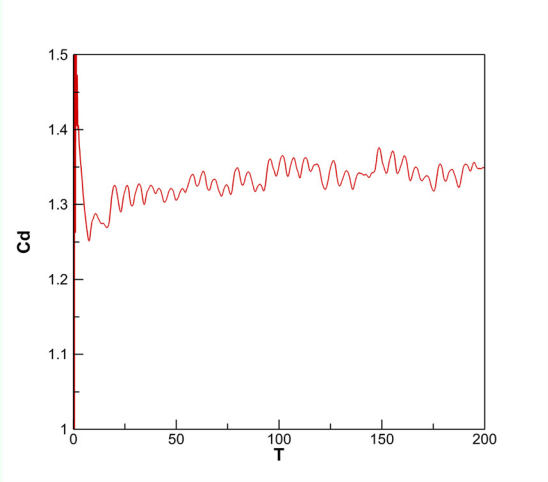
```
### mesh_name #####
ORGmesh
### lag_name #####
man.lag
### Reynolds number #####
300.
### IB\_key, No = 0, Yes = 1 #####
1
```

Parameters saved to 'Para.txt' and 'Bc-Ic.txt'.  
Euler grid generated with default settings.  
Simulation has started! The simulation results are now available.  
Would you like to proceed with any specific analysis or next steps?


Plot the instantaneous drag coefficient over time.









What is the time-averaged drag coefficient?

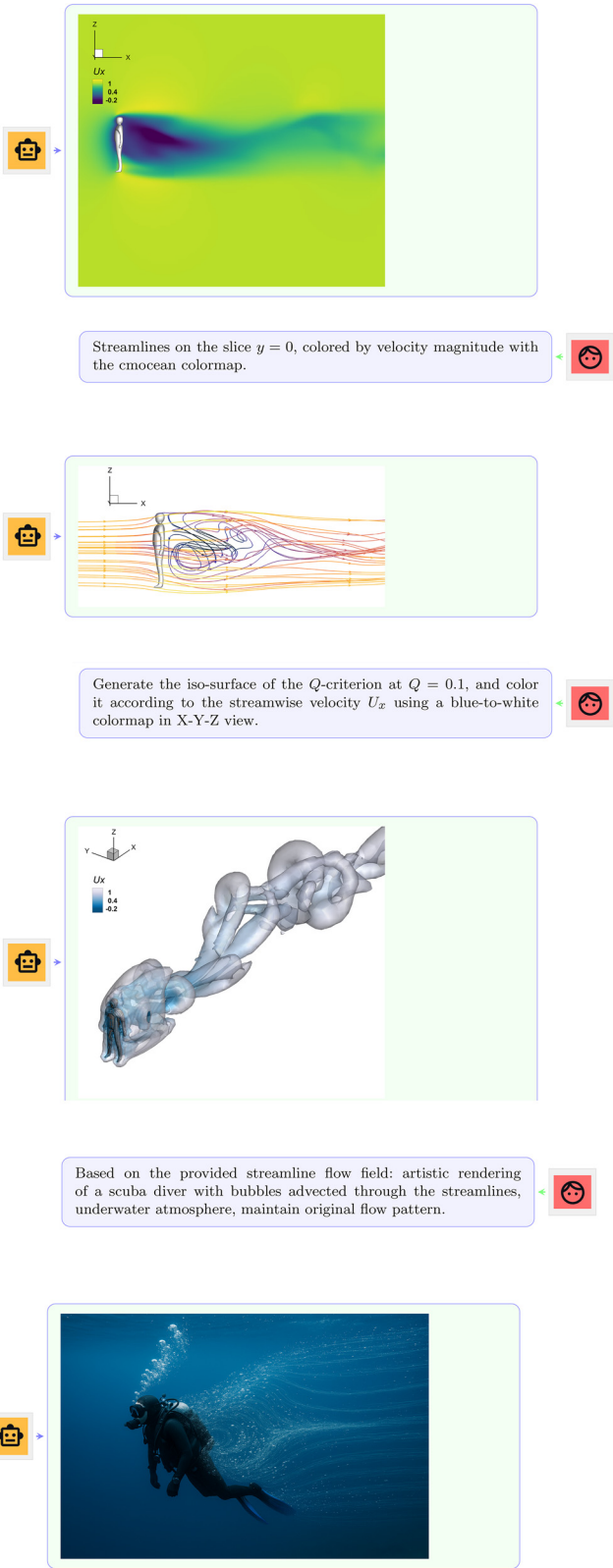




The time-averaged drag coefficient is approximately 1.3.

Visualize the  $u_x$  velocity component on the slice  $y = 0$  in the X-Z view, with default contour levels.





DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding authors upon reasonable request.

APPENDIX A: DETAILED PROMPT OF CFDAAGENT

This section provides a detailed exposition of the user-CFDAgent interaction. As an illustrative case, we consider the simulation of flow past a “human” body.

APPENDIX B: MODEL SELECTION AND COMPARISONS

As our experiments were conducted before the release of GPT-5 model families, all tests use GPT-4 series models; results are placed in the appendix for reference. This section focuses solely on LLM performance metrics, excluding Point-E success rates and CFD grid convergence analysis. We compared three models: GPT-4o (proprietary and high-fidelity reasoning), GPT-4o-mini (cost-efficient variant with reduced parameters), and GPT-oss-20B (an open-source 20B-parameter model, run on device with 16GB of RAM).

To assess model suitability for the multi-agent stages (preprocessing, solver, and postprocessing), we employed three complementary evaluation protocols: Monte Carlo test for robustness, ablation studies for agents, and adversarial inputs for contradiction detection. These tests used a fixed set of 100 diverse natural language prompts, measuring end-to-end task completion (successful orchestration from prompt to postprocessed output, excluding simulation execution).

In the Monte Carlo test, each model was queried 100 times with varied prompt phrasings (e.g., paraphrased descriptions of geometry and flow conditions). Success is defined as fully completing the pipeline: generating valid inputs for preprocessing, solver, and postprocessing without fatal errors. As shown in Table IX, GPT-4o achieved perfect success (100%), GPT-4o-mini demonstrated strong performance (91%), while GPT-oss-20B showed limited capability (52%). The GPT-oss-20B model exhibited typical failure patterns including excessive confirmation requests and a lack of default parameter values.

When consolidating the three agents into a single one, performance degraded significantly across all models. Only GPT-4o maintained partial functionality (32% success rate), while both GPT-4o-mini and GPT-oss-20B failed completely. GPT-4o-mini struggled to transition into the Agent Solver role after completing Agent Preprocessing tasks. GPT-4o, despite some success, exhibited hallucination behaviors during Agent Solver and Agent Postprocessing phases, including requesting unnecessary parameters and generating lengthy, irrelevant discussions about CFD simulation details. This underscores the value of specialized agents in mitigating role-switching challenges inherent to single-model approaches.

TABLE IX. Model performance comparison across different tests.

Test type	GPT-4o	GPT-4o-mini	GPT-oss-20B
Monte Carlo test	100%	91%	52%
Ablation test	32%	0%	0%
Contradiction detection	Pass	Pass	Fail

Both GPT-4o and GPT-4o-mini successfully identified logical contradictions when computational timesteps exceeded total simulation time. GPT-oss-20B partially detected such contradictions but failed to recover and complete the task loop after identification. Based on these findings, we select GPT-4o as the primary model for its exceptional robustness across all tests. For cost-sensitive deployments, GPT-4o-mini serves as a viable alternative, offering near-parity in simpler scenarios while reducing inference costs. GPT-oss-20B, despite local runnability, is unsuitable for production due to high failure rates in tasks.

With the language model selection process completed, we turn to a broader comparison between CFDAgent’s overall architecture and existing CFD automation methodologies. CFDAgent introduces an end-to-end, zero-shot and language-guided multi-agent framework that autonomously executes the full CFD pipeline. By integrating a zero-shot multi-agent architecture that orchestrates preprocessing, solving, and postprocessing entirely through natural language, CFDAgent enables simulations of complex, user-defined geometries without fine-tuning or domain-specific training. This approach fundamentally differs from recent works like OpenFOAMGPT,<sup>22</sup> which have made notable strides in automating CFD workflows through retrieval-augmented LLM agents that refine existing OpenFOAM cases, minimize syntax errors, and provide domain-specific guidance for high-fidelity simulations. Since the two are based on fundamentally different principles, with OpenFOAMGPT focusing on template optimization and guidance, while CFDAgent emphasizes direct geometry import and computation, a detailed comparison is not feasible. Based on the preceding tests and comparisons, several key challenges for CFDAgent were identified, as summarized in Table X.

For the Preprocess Agent, the system cannot specify complex conditions including non-uniform flows and static boundaries, limiting the simulation capability for realistic flow scenarios. Additionally, the geometry generation depends on Point-E’s generative capabilities, introducing instability and limiting deterministic control over features due to algorithmic randomness. For the Solver Agent, several limitations affect simulation quality. High Reynolds number computations are constrained by computational costs, restricting simulation scale. Furthermore, the complexity of parameter settings poses challenges where underperforming LLM agents may repeatedly confirm parameters or miss critical settings. For the Postprocess Agent, the visualization process heavily relies on predefined templates, making it difficult to handle specialized visualization instructions flexibly. Moreover, underperforming LLM agents may incorrectly process outputs, compromising the quality of final visualizations and analysis results.

TABLE X. Main challenges of three agents.

Agent	Key challenges
Preprocess	Complex inflow condition limitations Point-E geometry generation instability Static boundaries
Solver	High Reynolds number computational costs Parameter management complexity
Postprocess	Strong template dependency Output processing error risks

APPENDIX C: ALGORITHM OF CFDAgent

ALGORITHM 1. Workflow of CFDAgent.

```
1: procedure Main
2:   agent: 1→Preprocessing, 2→Solver,
           3→Postprocessing
3: end procedure
4: procedure Preprocessing
5:   on first entry: start GPT dialogue with SYSTEM PROMPT 1
6:   while agent = 1 do
7:     render UI → userInput → GPT reply
8:     if reply contains ###OUT1### then
9:       build point cloud from text → mesh
10:    else if reply contains ###OUT2### then
11:      build point cloud from uploaded image → mesh
12:    else if reply contains ###OUT3### then
13:      load user mesh
14:    end if
15:    visualize and select wind direction
16:    if any mesh exists then
17:      agent ← 2
18:    else
19:      save dialogue
20:    end if
21:  end while
22: end procedure
23: procedure Solver
24:   on first entry: start GPT dialogue
   with SYSTEM PROMPT 2
25:   while agent = 2 do
26:     render UI → userInput → GPT reply
27:     if reply contains ###OUTT### do
28:       parse CFD parameters & BCs; generate grid
29:       launch solver; agent ← 3
30:     else
31:       save dialogue
32:     end if
33:   end while
34: end procedure
35: procedure Postprocessing
36:   on first entry: start GPT dialogue with SYSTEM PROMPT 3
37:   while agent = 3 do
38:     render UI → userInput → GPT reply
39:     select by tag in reply
40:     Apply operation based on tag
     ∈ {POST1, POST2, POST3, POST4}:
41:     {time-series plot, slice contour,
      3D iso-surface, lift/drag statistics}
42:     save dialogue
43:   end while
44: end procedure
```

19 November 2025 06:05:42

## REFERENCES

- <sup>1</sup>A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," OpenAI, Technical Report (2018).
- <sup>2</sup>A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," OpenAI, Technical Report (2019).
- <sup>3</sup>T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, edited by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Curran Associates, Inc., Vancouver, Canada, 2020), Vol. 33, pp. 1877–1901.
- <sup>4</sup>J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat *et al.*, "GPT-4 technical report," [arXiv:2303.08774](https://arxiv.org/abs/2303.08774) (2023).
- <sup>5</sup>A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of the 30th International Conference on Neural Information Processing Systems (NIPS'17)* (Curran Associates Inc., Red Hook, NY, 2017), pp. 6000–6010.
- <sup>6</sup>J. Austin, A. Odena, M. Nye, M. Bosma, H. Michalewski, D. Dohan, E. Jiang, C. Cai, M. Terry, Q. Le *et al.*, "Program synthesis with large language models," [arXiv:2108.07732](https://arxiv.org/abs/2108.07732) (2021).
- <sup>7</sup>G. Li, H. Hammoud, H. Itani, D. Khizbullin, and B. Ghanem, "CAMEL: Communicative agents for 'mind' exploration of large language model society," in *Proceedings of the 37th International Conference on Neural Information Processing Systems (NeurIPS)*, 2023.
- <sup>8</sup>S. Farquhar, J. Kossen, L. Kuhn, and Y. Gal, "Detecting hallucinations in large language models using semantic entropy," *Nature* **630**, 625–630 (2024).
- <sup>9</sup>S. Wolfram, see <https://writings.stephenwolfram.com/2024/03/can-ai-solve-science/> for "Can AI solve science?" (Stephen Wolfram Writings, 2024).
- <sup>10</sup>H. Gao, L. Sun, and J.-X. Wang, "PhyGeoNet: Physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state PDEs on irregular domain," *J. Comput. Phys.* **428**, 110079 (2021).
- <sup>11</sup>P. Du, M. H. Parikh, X. Fan, X.-Y. Liu, and J.-X. Wang, "Conditional neural field latent diffusion model for generating spatiotemporal turbulence," *Nat. Commun.* **15**(1), 10416 (2024).
- <sup>12</sup>K. Duraisamy, G. Iaccarino, and H. Xiao, "Turbulence modeling in the age of data," *Annu. Rev. Fluid Mech.* **51**(1), 357–377 (2019).
- <sup>13</sup>S. L. Brunton, B. R. Noack, and P. Koumoutsakos, "Machine learning for fluid mechanics," *Annu. Rev. Fluid Mech.* **52**(1), 477–508 (2020).
- <sup>14</sup>A. Kashefi, "Kolmogorov–Arnold PointNet: Deep learning for prediction of fluid fields on irregular geometries," *Comput. Methods Appl. Mech. Eng.* **439**, 117888 (2025).
- <sup>15</sup>Y. Yuan and A. Lozano-Durán, "Dimensionless learning based on information," [arXiv:2504.03927](https://arxiv.org/abs/2504.03927) (2025).
- <sup>16</sup>L. Wang, L. Zhang, and G. He, "Evaluations of large language models in computational fluid dynamics: Leveraging, learning and creating knowledge," *Theor. Appl. Mech. Lett.* **15**, 100597 (2025).
- <sup>17</sup>Q. Jiang, Z. Gao, and G. E. Karniadakis, "DeepSeek vs. ChatGPT vs. Claude: A comparative study for scientific computing and scientific machine learning tasks," *Theor. Appl. Mech. Lett.* **15**(3), 100583 (2025).
- <sup>18</sup>M. Du, Y. Chen, Z. Wang, L. Nie, and D. Zhang, "Large language models for automatic equation discovery of nonlinear dynamics," *Phys. Fluids* **36**(9), 097121 (2024).
- <sup>19</sup>M. Lei and Y. Zhang, "Generating airfoils from text: FoilCLIP, a novel framework for language-conditioned aerodynamic design," *Theor. Appl. Mech. Lett.* **15**, 100602 (2025).
- <sup>20</sup>Z. Dong, Z. Lu, and Y. Yang, "Fine-tuning a large language model for automating computational fluid dynamics simulations," *Theor. Appl. Mech. Lett.* **15**, 100594 (2025).
- <sup>21</sup>X. Zhang, Z. Xu, G. Zhu, C. M. J. Tay, Y. Cui, B. C. Khoo, and L. Zhu, "Using large language models for parametric shape optimization," *Phys. Fluids* **37**(8), 083601 (2025).
- <sup>22</sup>S. Pandey, R. Xu, W. Wang, and X. Chu, "OpenFOAMGPT: A retrieval-augmented large language model (LLM) agent for OpenFOAM-based computational fluid dynamics," *Phys. Fluids* **37**(3), 035120 (2025).
- <sup>23</sup>W. Wang, R. Xu, J. Feng, Q. Zhang, S. Pandey, and X. Chu, "A status quo investigation of large-language models for cost-effective computational fluid dynamics automation with OpenFOAMGPT," *Theor. Appl. Mech. Lett.* (in press) (2025).
- <sup>24</sup>A. Q. Nichol and P. Dhariwal, "Improved denoising diffusion probabilistic models," in *International Conference on Machine Learning* (PMLR, 2021), pp. 8162–8171.
- <sup>25</sup>A. Jain, B. Mildenhall, J. T. Barron, P. Abbeel, and B. Poole, "Zero-shot text-guided object generation with dream fields," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, New Orleans, LA (IEEE, 2022), pp. 867–876.
- <sup>26</sup>C.-H. Lin, J. Gao, L. Tang, T. Takikawa, X. Zeng, X. Huang, K. Kreis, S. Fidler, M.-Y. Liu, and T.-Y. Lin, "Magic3D: High-resolution text-to-3D content creation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Nashville (IEEE, 2023), pp. 300–309.
- <sup>27</sup>A. Sanghi, H. Chu, J. G. Lambourne, Y. Wang, C.-Y. Cheng, M. Fumero, and K. R. Malekshan, "CLIP-Forge: Towards zero-shot text-to-shape generation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (IEEE, 2022), pp. 18603–18613.
- <sup>28</sup>P. Mittal, Y.-C. Cheng, M. Singh, and S. Tulsiani, "AutoSDF: Shape priors for 3D completion, reconstruction and generation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (IEEE, 2022), pp. 306–315.
- <sup>29</sup>R. Fu, X. Zhan, Y. Chen, D. Ritchie, and S. Sridhar, "ShapeCrafter: A recursive text-conditioned 3D shape generation model," in *Advances in Neural Information Processing Systems*, edited by S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (Curran Associates, Inc., New Orleans, 2022), Vol. 35, pp. 8882–8895.
- <sup>30</sup>X. Zeng, A. Vahdat, F. Williams, Z. Gojcic, O. Litany, S. Fidler, and K. Kreis, "LION: Latent point diffusion models for 3D shape generation," in *Proceedings of the 36th Conference on Neural Information Processing Systems (NeurIPS)*, 2022.
- <sup>31</sup>A. Sanghi, R. Fu, V. Liu, K. D. Willis, H. Shayani, A. H. Khasahmadi, S. Sridhar, and D. Ritchie, "Clip-sculptor: Zero-shot generation of high-fidelity and diverse shapes from natural language," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (IEEE, 2023), pp. 18339–18348.
- <sup>32</sup>A. Nichol, H. Jun, P. Dhariwal, P. Mishkin, and M. Chen, "Point-E: A system for generating 3D point clouds from complex prompts," [arXiv:2212.08751](https://arxiv.org/abs/2212.08751) (2022).
- <sup>33</sup>C. S. Peskin, "Flow patterns around heart valves: A numerical method," *J. Comput. Phys.* **10**(2), 252–271 (1972).
- <sup>34</sup>C. S. Peskin, "The immersed boundary method," *Acta Numer.* **11**, 479–517 (2002).
- <sup>35</sup>R. Mittal and G. Iaccarino, "Immersed boundary methods," *Annu. Rev. Fluid Mech.* **37**, 239–261 (2005).
- <sup>36</sup>J. Mohd-Yusof, "Combined immersed-boundary/B-spline methods for simulations of flow in complex geometries," in *Annual Research Briefs* (Center for Turbulence Research, Stanford, CA, 1997), pp. 317–327.
- <sup>37</sup>E. A. Fadlun, R. Verzicco, P. Orlandi, and J. Mohd-Yusof, "Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations," *J. Comput. Phys.* **161**(1), 35–60 (2000).
- <sup>38</sup>G. Iaccarino and R. Verzicco, "Immersed boundary technique for turbulent flow simulations," *Appl. Mech. Rev.* **56**(3), 331–347 (2003).
- <sup>39</sup>J. Kim, D. Kim, and H. Choi, "An immersed-boundary finite-volume method for simulations of flow in complex geometries," *J. Comput. Phys.* **171**(1), 132–150 (2001).
- <sup>40</sup>J. Yang and E. Balaras, "An embedded-boundary formulation for large-eddy simulation of turbulent flows interacting with moving boundaries," *J. Comput. Phys.* **215**(1), 12–40 (2006).
- <sup>41</sup>N. Zhang and Z. C. Zheng, "An improved direct-forcing immersed-boundary method for finite difference applications," *J. Comput. Phys.* **221**(1), 250–268 (2007).
- <sup>42</sup>M. Uhlmann, "An immersed boundary method with direct forcing for the simulation of particulate flows," *J. Comput. Phys.* **209**(2), 448–476 (2005).

- <sup>43</sup>K. Taira and T. Colonius, "The immersed boundary method: A projection approach," *J. Comput. Phys.* **225**(2), 2118–2137 (2007).
- <sup>44</sup>X. Yang, X. Zhang, Z. Li, and G.-W. He, "A smoothing technique for discrete delta functions with application to immersed boundary method in moving boundary simulations," *J. Comput. Phys.* **228**(20), 7821–7836 (2009).
- <sup>45</sup>S. Wang and X. Zhang, "An immersed boundary method based on discrete stream function formulation for two- and three-dimensional incompressible flows," *J. Comput. Phys.* **230**(9), 3479–3499 (2011).
- <sup>46</sup>Y. Chen, Y. Liu, and S. Wang, "Streamwise ram effect and tip vortex enhance the lift of a butterfly-inspired flapping wing," *J. Fluid Mech.* **1005**, A13 (2025).
- <sup>47</sup>C. Wang, Z. Xu, X. Zhang, and S. Wang, "Bayesian optimization for the spanwise oscillation of a gliding flat plate," *Optim. Eng.* **24**, 2763–2772 (2023).
- <sup>48</sup>Z. Zhou, Z. Xu, S. Wang *et al.*, "Wall-modeled large-eddy simulation of noise generated by turbulence around an appended axisymmetric body of revolution," *J. Hydrodyn.* **34**, 533–554 (2022).
- <sup>49</sup>C. Wang, Z. Xu, X. Zhang, and S. Wang, "Optimal reduced frequency for the power efficiency of a flat plate gliding with spanwise oscillations," *Phys. Fluids* **33**(11), 111908 (2021).
- <sup>50</sup>Streamlit, see <https://github.com/streamlit/streamlit> for "Streamlit: A faster way to build and share data apps" (accessed April 27, 2025).
- <sup>51</sup>J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *arXiv:2006.11239* (2020).
- <sup>52</sup>A. Nichol, P. Dhariwal, A. Ramesh, P. Shyam, P. Mishkin, B. McGrew, I. Sutskever, and M. Chen, "GLIDE: Towards photorealistic image generation and editing with text-guided diffusion models," *arXiv:2112.10741* (2021).
- <sup>53</sup>W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," *ACM SIGGRAPH Comput. Graph.* **21**(4), 163–169 (1987).
- <sup>54</sup>S. Wang, G. He, and X. Zhang, "Parallel computing strategy for a flow solver based on immersed boundary method and discrete stream-function formulation," *Comput. Fluids* **88**, 210–224 (2013).
- <sup>55</sup>T. A. Johnson and V. C. Patel, "Flow past a sphere up to a Reynolds number of 300," *J. Fluid Mech.* **378**, 19–70 (1999).
- <sup>56</sup>A. K. Saha, "Three-dimensional numerical simulations of the transition of flow past a cube," *Phys. Fluids* **16**(5), 1630–1646 (2004).
- <sup>57</sup>M. N. Linnick and H. F. Fasel, "A high-order immersed interface method for simulating unsteady incompressible flows on irregular domains," *J. Comput. Phys.* **204**(1), 157–192 (2005).