CFDagent: A Language-Guided, Zero-Shot Multi-Agent System for Complex Flow Simulation

Zhaoyue Xu[1,2], Long Wang[1,3], Chunyu Wang[1,3], Yixin Chen[1,3], Qingyong Luo[1,3], Hua-Dong Yao[1,2*], Shizhao Wang[1,3*], Guowei He[1,3]

[1]The State Key Laboratory of Nonlinear Mechanics, Institute of Mechanics, Chinese Academy of Sciences, Beijing, 100190, China.
[2]Department of Mechanics and Maritime Sciences, Chalmers University of Technology, Gothenburg, 41296, Sweden.
[3]School of Engineering Sciences, University of Chinese Academy of Sciences, Beijing, 100049, China.

*Corresponding author(s). E-mail(s): huadong.yao@chalmers.se; wangsz@lnm.imech.ac.cn;
Contributing authors: zhaoyue@chalmers.se; wanglong@imech.ac.cn; wangchunyu@imech.ac.cn; chenyixin@imech.ac.cn; luoqingyong@imech.ac.cn; hgw@lnm.imech.ac.cn;

Abstract

We introduce CFDagent, a zero-shot, multi-agent system that enables fully autonomous computational fluid dynamics (CFD) simulations from natural language prompts. CFDagent integrates three specialized LLM-driven agents: (i) the Preprocessing Agent that generates 3D geometries from textual or visual inputs using a hybrid text-to-3D diffusion model (Point-E) and automatically meshes the geometries; (ii) the Solver Agent that configures and executes an immersed boundary flow solver; and (iii) the Postprocessing Agent that analyzes and visualizes the results, including multimodal renderings. These agents are interactively guided by GPT-4o via conversational prompts, enabling intuitive and user-friendly interaction. We validate CFDagent by reproducing canonical sphere flows at Reynolds numbers of 100 and 300 using three distinct inputs: a simple text prompt (i.e., "sphere"), an image-based input, and a standard sphere model. The computed drag and lift coefficients from meshes produced by each input approach closely match available data. The proposed system enables synthesization of flow simulations and photorealistic visualizations for complex

geometries. Through extensive tests on canonical and realistic scenarios, we demonstrate the robustness, versatility, and practical applicability of CFDagent. By bridging generative AI with high-fidelity simulations, CFDagent significantly lowers barriers to expert-level CFD, unlocking broad opportunities in education, scientific research, and practical engineering applications.

Keywords: computational fluid dynamics, large language models, multi-agent system, immersed boundary method

1 Main

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

Despite their impressive capabilities, current standalone LLMs remain limited in tasks requiring extensive, precise, and systematic content generation—particularly in computational physics and simulation [7] due to persistent challenges such as hallucination [8]. As elucidated by Wolfram [9], fundamental constraints inherent to 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 [6], 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 [10–15], current AI technologies—including LLMs—nevertheless remain unable to function as direct solvers for the underlying governing equations [9].

To overcome these limitations, integrating LLM capabilities into simulation frameworks has emerged as an increasingly promising direction. Recent studies [16–19] have proposed hybrid approaches leveraging LLM-driven techniques for enhancing simulations and engineering design optimization. Notably, these approaches commonly involve interactive natural language prompts, automatic equation formulation, and assisted simulation frameworks. For example, MetaOpenFOAM [17] proposes an automated CFD framework based on multi-agent systems. Similarly, OpenFOAM-GPT [18] introduces a RAG-augmented LLM framework tailored to OpenFOAM-based CFD simulations. Nevertheless, existing OpenFOAM-based CFD agent systems continue to rely on manually curated case libraries for knowledge retrieval, potentially limiting their scalability and generalizability to previously unseen or highly complex flow scenarios. Consequently, such systems often require substantial supervision when

dealing with atypical or high-complexity tasks, thereby limiting their degree of end-to-end automation. This limitation highlights that current frameworks are not yet capable of robust zero-shot generalization and still depend on task-specific adaptation or expert intervention to operate effectively in novel problem domains.

In this work, we propose an LLM-driven multi-agent framework, CFDagent, to enable fully language guided, zero-shot and end-to-end complex flow simulations. 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, as illustrated in Fig. 1.

To reform the first stage, the Preprocessing Agent integrates a hybrid text-to-3D diffusion model, point-E, to autonomously generate geometry and mesh. Point-E generates semantically accurate 3D point clouds from natural-language prompts by first synthesizing images via GLIDE [20] and then reconstructing geometry, achieving 10-100x faster sampling than earlier text-to-3D approaches [21–28]. 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 [29–31] and leveraging direct-forcing methods [32–40], imposes boundary conditions on complex geometries without body-fitted meshes by solving small, localized linear systems [41]. Enhanced through parallel domain decomposition, our implementation delivers both high accuracy and computational efficiency, as validated by extensive benchmarks [41–45]. Lastly, the Postprocessing Agent equips the LLM with scripts for analyzing physical quantities, visualizing flow fields, and fusing simulation results to generate realistic imagery of physical objects.

2 Results and discussion

2.1 Autonomous flow simulation around a sphere as a canonical geometry

By leveraging the capabilities of the proposed LLM-driven CFD framework, we conducted three-dimensional numerical simulations of unsteady flow around a sphere at Reynolds numbers of 100 and 300. The sphere is a canonical geometry that serves as a benchmark for validating the accuracy and reliability of the simulation framework.

As shown in Table 1, when the user expresses interest in CFD, CFDagent 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 mesh. All three modes ultimately produce a Lagrangian surface mesh representing the object. Subsequently, the Solver Agent prompts the user for

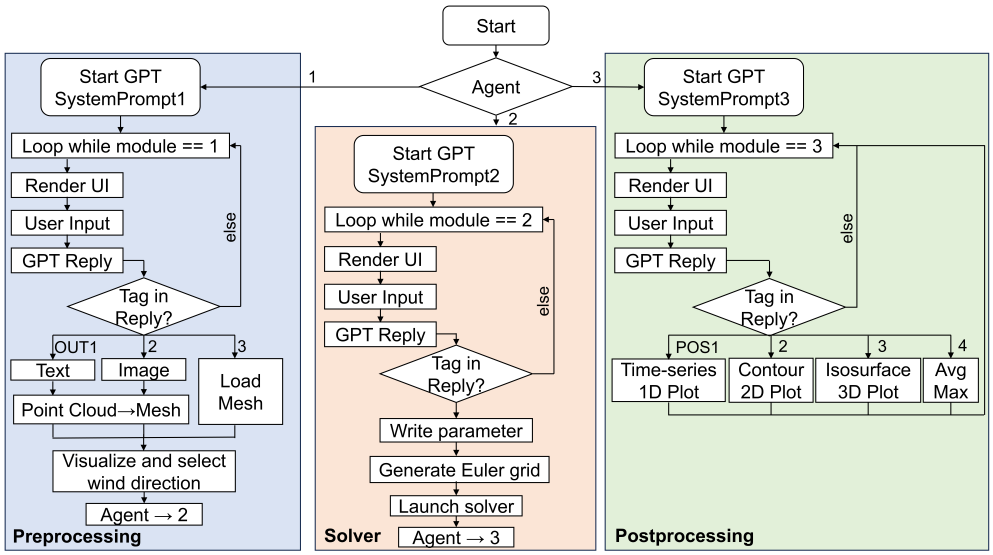

Fig. 1: **Workflow diagram of CFDagent.** The system consists of three core agents: the Preprocessing Agent, the Solver Agent, and the Postprocessing Agent. Each agent's role is defined through a tailored system prompt that conditions the LLM to perform stage-specific tasks within the simulation workflow. The agents interact with the user via natural language prompts, enabling dynamic input interpretation and autonomous execution of key operations. These include geometry and mesh generation, solver configuration and flow simulation, as well as postprocessing for quantitative analysis and result visualization.

detailed numerical simulation settings and initiates the solver. Details of the default Eulerian grid used in this study are provided in the Supplementary Information. Mesh convergence was rigorously validated through multiple tests, confirming that variations in mesh resolution and domain size have a negligible effect on the flow characteristics [41, 42, 45, 46].

The Postprocessing Agent generates figures to visualize the simulation results as specified by the user prompt. In the post-processing 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 by the dynamic pressure, $q = \frac{1}{2}\rho U^2$, and a reference area, S:

$$C_d = D/(qS), C_l = L/(qS). \tag{1}$$

Here, ρ is the fluid density and U the freestream velocity.

At Re= 100, the flow around a sphere is nearly 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 2 provides a comparison of the drag coeffient C_d from numerical simulations employing different geometry-inputting approaches, as well as previously reported literature

User Prompt	**CFDagent**
Compute the flow around a sphere at Re300	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]`	What is the end time of the simulation?
200 should be enough.	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. *Here's a summary of all values.* Do you confirm these values?
Yes!	Use the default Euler grid or customize it?
Default.	Parameters saved to Para.txt and Bc_Ic.txt. Euler grid is generated with default settings. Simulation has started!

Table 1: An example interaction between the user and CFDagent during a simulation of flow around a sphere. The complete dialogue example can be found in Supplementary Information.

values. 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. A representative instantaneous vortical structure under this condition is illustrated in Fig. 2, where vortices are visualized and identified based on the Q-criterion, a common method for vortex identification. The observed vortex shedding patterns are consistent with those previously documented by Wang and Zhang [41]. Table 3 compares the computed time-averaged drag and lift coefficients against literature benchmarks, demonstrating excellent quantitative agreement and further supporting the robustness and accuracy of the simulation methodology employed.

Table 2: 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 [47]	1.10
Fadlun et al. [33]	1.08

Table 3: 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 [47]	0.66	0.069
Kim et al. [35]	0.66	0.067

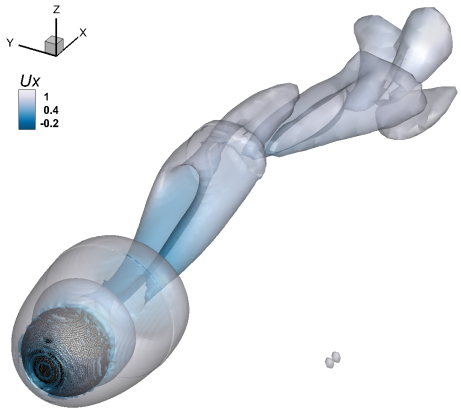

Fig. 2: **Instantaneous vortical structure for flow past a sphere at Re = 300.** The vortical structures are visualized using iso-surfaces of the Q-criterion, highlighting regions with a high ratio of vorticity to strain. The iso-surfaces are colored by the streamwise velocity, ranging from -0.2 to 1, with colors transitioning from blue to white, illustrating the variation of the flow speed around the vortex structures.

2.2 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 pre-existing CAD models or case-specific training. This capability allows for the simulation of arbitrary objects, significantly enhancing the versatility and applicability of the framework. Here, we select the geometry "human" as an illustrative scenario to demonstrate the framework's end-to-end automated simulation pipeline. Initially, the Preprocessing Agent receives the object description "human" and the specified Reynolds number, Re = 300. The agent automatically generates the

corresponding geometric model and computational mesh using the integrated text-to-3D model and the previously described mesh conversion methods. The simulation employs the same settings as those used in the sphere flow cases to ensure consistency across scenarios. Consequently, the prompt is similar to that of Table 1, differing only in the geometry name, and is therefore omitted.

After completing the simulation, the Postprocessing Agent executes several visualization and analysis commands provided through natural language prompts. Issuing the prompt "Plot the instantaneous drag coefficient over time." triggers the `POS1` analysis routine illustrated in Fig. 1. The instantaneous C_d of the human geometry is extracted from the data file and subsequently plotted, as shown in Fig. 3(a). The time-averaged drag coefficient obtained from this simulation is approximately $C_d \approx 1.3$. Subsequently, we explore additional flow features using natural language prompts. Responding to the prompt "Visualize the u_x velocity component on the slice $y = 0$", the agent generates the velocity distribution in the streamwise direction on the specified plane, as illustrated in Fig. 3(b). With the command "Streamlines on the $y = 0$ slice, colored by velocity magnitude with the cmocean colormap.", the flow streamlines on this slice are visualized in Fig. 3(c). These two figures clearly highlight the velocity field structure around the geometry. Further, the instruction "Generate the iso-surface of the Q-criterion at 0.1, and color it according to the streamwise velocity u_x using a blue-to-white colormap." results in the visualization of three-dimensional vortex structures, as depicted in Fig. 3(d). Finally, leveraging the image synthesis capabilities of GPT-4o, the framework responds to the prompt "Generate a visualization of a diver submerged in the ocean, with bubbles advected along the computed streamlines of the flow field.", yielding a realistic synthesized image reflecting the flow features obtained from the CFD simulation, as shown in Fig. 3(e).

To further validate the generalization capabilities of our proposed LLM-CFD framework, we examine simulations prompted by diverse textual inputs describing various complex geometries beyond the sphere example. Specifically, geometries "cat", "dog", "motorcycle", "pot", "side mirror", "tower", and "tree" are autonomously generated and simulated under identical conditions as the previous simulation of human geometry. Fig. 4 demonstrates representative results obtained for each case. For each geometry, the velocity fields, streamlines, and vortex structures are systematically visualized, alongside GPT-4o synthesized images reflecting specific flow-related scenarios described by natural language prompts, which are consistent with the "human" case. Notably, the velocity distributions on the symmetry plane consistently reveal flow around sharp edges and stagnation points at frontal surfaces. Streamline visualizations clearly illustrate distinct wake patterns, reflecting flow separation behavior specific to each geometry. For example, the complex wake behind the "pot" geometry exhibits dispersed vortical structures, contrasting sharply with the more streamlined wake structures observed behind the "motorcycle" geometries. The three-dimensional vortex structures, visualized via the Q-criterion iso-surfaces, emphasize differences in vortex shedding intensity and frequency across the geometries. The "side mirror" case, in particular, shows pronounced vortex shedding, indicative of complex, unsteady wake dynamics. Finally, the framework demonstrates its multi-modal synthesis capabilities

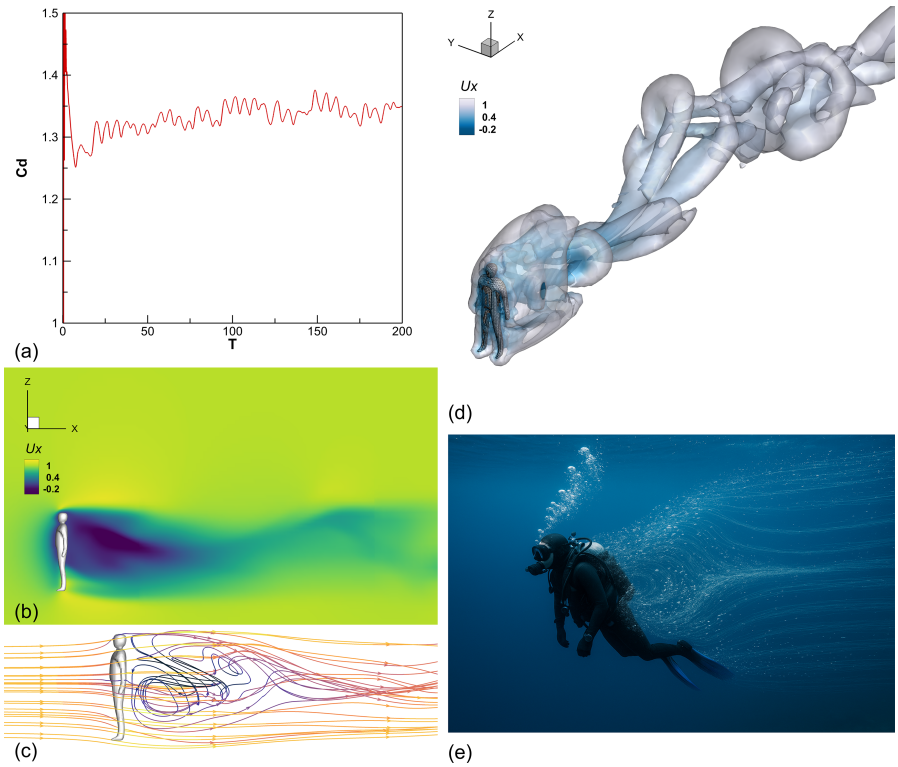

Fig. 3: **Postprocessing results from CFDagent for flow past a human geometry at Re**= 300**.** (a) instantaneous drag coefficient, (b) streamwise velocity component at slice $y = 0$, (c) streamlines colored by velocity magnitude at slice $y = 0$, (d) isosurface visualization of the Q-criterion ($Q = 0.1$) colored by streamwise velocity, and (e) GPT-4 synthesized visualization based on simulation results.

through GPT-4o-generated images, effectively visualizing realistic, scenario-specific phenomena inspired by the computed CFD results.

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.

3 Conclusion

In this study, we introduce CFDagent, a zero-shot, natural-language-driven multi-agent framework capable of autonomously performing complex CFD simulations. CFDagent integrates advanced LLM capabilities, a hybrid text-to-3D diffusion model, and an IB solver, enabling robust geometry generation and accurate flow simulation for arbitrary, user-defined geometries. This approach significantly simplifies the CFD

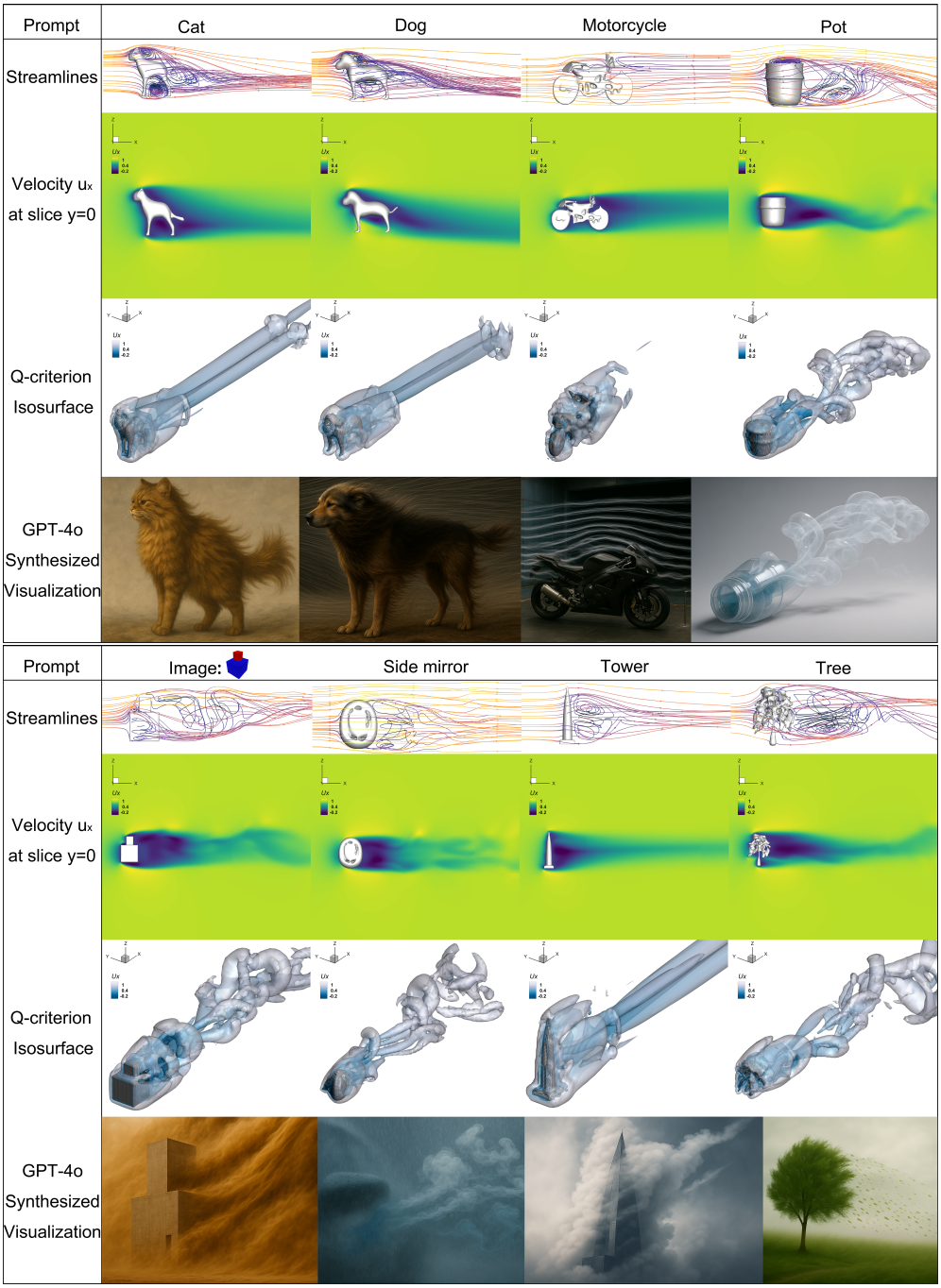

Fig. 4: CFDagent postprocessing results demonstrating generalization for different prompts: Streamlines, velocity distributions, vortex structures (Q-criterion iso-surface), and GPT-4o synthesized visualizations.

workflow, allowing users to initiate simulations and perform preliminary analyses without detailed expertise or extensive pre-processing. CFDagent supports multi-mode inputs for geometry generation, including natural language descriptions of object shapes, 2D images, and 3D geometry meshes. It also offers diverse output modalities, including quantitative plots, 2D/3D visualizations, and even synthesized realistic images generated from simulation results. CFDagent demonstrates strong generalization to simulate flows around diverse and complex geometries, underscoring its promise as an intelligent and accessible assistant to simulate and interpret real-world flow phenomena.

Despite its current strengths, CFDagent exhibits some limitations that must be addressed to further enhance its practicality and efficiency. The limited quality and controllability of the generated geometries hinder precision. Performance is constrained by the inherent complexity of CFD simulations, leading to significantly prolonged processing times. Future improvements in AI-driven geometry generation quality, increased geometric controllability, and optimized computational strategies could significantly alleviate these constraints.

The architecture of CFDagent offers strong potential for further development, especially through integration with rapidly evolving AI techniques. Notably, the IB solver's inherent capacity to handle dynamic meshes offers exciting potential for simulating flows around moving or deforming objects, expanding applicability of CFDagent to diverse, real-world scenarios such as biolocomotion or dynamic engineering components. Additionally, coupling CFDagent with emerging generative AI models could facilitate the automatic production of high-quality, informative visualizations, supporting more comprehensive fluid dynamics analysis. Ultimately, CFDagent could be integrated with more powerful general-purpose language models to provide more intuitive guidance for simulations and deeper analysis of results.

4 Methods

4.1 Zero-shot multi-agent system

The zero-shot multi-agent system integrates natural language interaction with three specialized agents—the Preprocessing Agent, the Solver Agent, and the Postprocessing Agent—to autonomously configure and execute CFD simulations from natural language instructions. Implemented as an interactive web application using Streamlit [48], the framework incorporates an OpenAI GPT-based assistant [4] to facilitate user-agent dialogue and task coordination. Through the three agents, the user's high-level natural language input is transformed into a fully configured simulation, with minimal manual effort required, as described in the algorithm shown in Fig. 1 and detailed in Supplementary Information.

The Preprocessing Agent initiates dialogue and determines whether the user intends to run a fluid-dynamics simulation and generate the corresponding object. In the first stage, the user engages with a GPT-driven conversational interface. If the user expresses interest in a fluid simulation, GPT will start to assist in 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 pre-existing 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 watertight surface mesh from this field. The resulting triangular mesh, also referred to as the Lagrangian mesh, approximates the target object's geometry. If the user instead uploads an existing 3D mesh in the Preprocessing Agent, the generative modeling step is skipped and the provided geometry 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. At the end, the geometry mesh is finalized and properly positioned.

Within the Solver Agent, the simulation parameters are first configured, followed by the launching of the simulation. First 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, 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 mesh can also be specified by setting the grid size and computational domain, as is illustrated in Supplementary Information. 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 grids 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 section 4.2.3.

Within the Postprocessing Agent, we perform postprocessing to visualize the simulation results and extract quantitative metrics for analysis. All postprocessing is performed collaboratively by the GPT-assisted agent and custom scripts: GPT first adjusts script parameters based on the user prompt, then executes the scripts accordingly. The framework explicitly enables automated extraction and visualization of probe-based time-series data, contour plots from cross-sectional slices, three-dimensional rendering of iso-surfaces, and lift/drag coefficient histories.

4.2 Relevant technologies

4.2.1 Large language models

LLMs such as GPT-4 series [4] have demonstrated strong zero-shot reasoning and multimodal understanding capabilities. In the context of CFD processing, these strengths

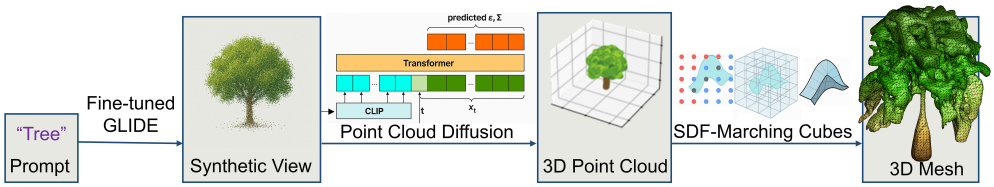

Fig. 5: **Text-to-3D Mesh Generation Pipeline.** A natural language prompt is processed to a fine-tuned GLIDE model to generate a synthetic image, which is then converted into a 3D point cloud using point cloud diffusion with CLIP and a transformer. The resulting point cloud is transformed into a 3D mesh using SDF estimation and Marching Cubes.

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.

Using strictly formatted command blocks within a three-tier slot-filling structure—system message, command block, and self-check—ensures deterministic parsing and natural dialogue, with any missing slot prompting exactly one concise follow-up question. After benchmarking several candidates on manually curated conversational test cases, we selected **GPT-4o** for production 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 production model only requires editing a single line in the Streamlit service, minimizing deployment friction.

4.2.2 3D geometry generation based on Point-E

In this study, we employ the Point-E framework [28] to efficiently generate 3D geometries from text prompts or 2D images. As illustrated in Fig. 5, the pipeline involves generating a synthetic image from a text prompt using a fine-tuned GLIDE model, converting it into a 3D point cloud, and then into a 3D mesh. The key idea of Point-E is to decouple the complex task of text-to-3D mapping into two successive diffusion-based generative steps: (1) text-to-image generation, and (2) image-to-point-cloud synthesis. If a 2D image is provided, the framework directly proceeds to the second stage.

The first stage of our pipeline involves generating a synthetic rendered view I from a given text prompt using a pretrained text-to-image diffusion model based on Guided Language-to-Image Diffusion for Generation and Editing (GLIDE) [20]. The GLIDE model builds on the Denoising Diffusion Probabilistic Model (DDPM) framework [49], which defines a forward process that progressively corrupts data $q(\mathbf{x}_0)$ by incrementally adding Gaussian noise across T timesteps, producing latent variables $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T$. At each step t, the conditional distribution is defined as

$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \mathcal{N}\left(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I}\right), \tag{2}$$

where $\beta_t \in (0,1)$ is a small positive scalar from a predefined variance schedule that determines the noise magnitude injected into $\mathbf{x}_{t-1}$ at step t. Intuitively, as $t \to T$, the data is progressively transformed into isotropic Gaussian noise. A model is then trained to approximate the reverse denoising process, enabling the generation of data. More details can be found in the Supplementary Information. The GLIDE model, which extends the standard Denoising Diffusion Probabilistic Model (DDPM) framework by introducing text conditioning, generates the rendered view I from a given text prompt

$$I \sim p_{\theta_1}(\cdot \mid \text{text}), \tag{3}$$

where θ_1 represents the pretrained parameters of the GLIDE model. Crucially, the GLIDE model is fine-tuned on a dataset of 3D renderings, so that the synthesized images are biased towards plausible single-view projections of 3D objects.

The second stage involves generating a 3D point cloud $\mathcal{P}$ conditioned on the synthesized image I via a point cloud diffusion model with the pretrained parameters θ_2 [28]

$$\mathcal{P} \sim p_{\theta_2}(\cdot \mid I), \quad \mathcal{P} = \{(\mathbf{p}_i, \mathbf{c}_i)\}_{i=1}^{N}, \quad \mathbf{p}_i \in \mathbb{R}^3, \quad \mathbf{c}_i \in [0,1]^3. \tag{4}$$

where $\mathbf{p}_i$ denotes the 3D coordinates of the i-th point, and $\mathbf{c}_i$ represents RGB color. The model directly outputs a tensor of shape $N \times 6$. To condition on the image, a frozen, pretrained CLIP model encodes the synthetic view I into a spatial feature grid, which is projected into transformer tokens serving as conditioning input. The noised point cloud $\mathbf{x}_t$ at timestep t, alongside a learned embedding of t, are tokenized and fed into the transformer. The transformer model then predicts both the mean and variance of the reverse distribution:

$$p_{\theta_2}(\mathbf{x}_{t-1} \mid \mathbf{x}_t, I) = \mathcal{N}(\mu_{\theta_2}(\mathbf{x}_t, t, I), \Sigma_{\theta_2}(\mathbf{x}_t, t, I)), \tag{5}$$

where output tokens corresponding to $\mathbf{x}_t$ are used to compute μ_θ and Σ_θ, typically by predicting the noise ϵ and using DDPM parameterizations [49]. In practice, the stage employs a hierarchical two-step procedure: the point cloud diffusion model produces a coarse point cloud with 1024 points, and an upsampler model refines it to 4096 points.

For simulations, we convert the generated point cloud $\mathcal{P}$ into a watertight triangular mesh by learning an implicit representation via a signed distance function (SDF). A transformer-based network $\phi_\psi : \mathbb{R}^3 \to \mathbb{R}$ takes arbitrary 3D spatial queries $\mathbf{x} \in \mathbb{R}^3$ as input to predict their signed distance relative to the object's surface, where $\phi_\psi(\mathbf{x}) < 0$ denotes points outside the surface and $\phi_\psi(\mathbf{x}) > 0$ denotes points inside. The object's surface is defined as the zero-level set $\mathcal{S} = \{\mathbf{x} \in \mathbb{R}^3 \mid \phi_\psi(\mathbf{x}) = 0\}$, which is subsequently converted into a watertight triangular mesh with Marching Cubes, followed by Laplacian smoothing to attenuate voxelization artifacts.

4.2.3 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:

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

$$\nabla\cdot\mathbf{u} = 0, \tag{7}$$

where $\mathbf{u}(\mathbf{x}, t)$ is the Eulerian velocity field, $p(\mathbf{x}, t)$ the pressure, and Re 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 S (with Lagrangian coordinate $\mathbf{X}$). This force is spread to the Eulerian grid via a convolution with a regularized delta function δ_h:

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

Here δ_h ensures a smooth transfer of force from the Lagrangian markers to the Eulerian grid. Conversely, the fluid velocity at the Lagrangian markers is obtained by interpolating the Eulerian field: $\mathbf{U}^*(\mathbf{X}, t) = \int_{\mathbf{x}\in\Omega} \mathbf{u}(\mathbf{x}, t)\delta_h(\mathbf{x}-\mathbf{X})d\mathbf{x}$. 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 strong scalability. More numerical details and validations can be found in Supplementary Information and our previous publications [41].

Data availability

All the simulations carried out in this study can be found under DOI:10.5281/zenodo.15383155. Source data are provided with this paper.

Code availability

An open-source version of the CFDagent framework has been released at DOI:10.5281/zenodo.1538315 as well. Access to the proprietary GPT-4 API can be obtained through OpenAI.

References

[1] Radford, A., Narasimhan, K.: Improving language understanding by generative pre-training. (2018)

[2] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I.: Language models are unsupervised multitask learners. OpenAI (2019)

[3] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., Amodei, D.: Language models are few-shot learners. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.F., Lin, H. (eds.) Advances in Neural Information Processing Systems, vol. 33, pp. 1877–1901. Curran Associates, Inc., Vancouver, Canada (2020)

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

[5] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Proceedings of the 30th International Conference on Neural Information Processing Systems. NIPS'17, pp. 6000–6010. Curran Associates Inc., Red Hook, NY, USA (2017)

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

[7] Li, G., Hammoud, H., Itani, H., Khizbullin, D., Ghanem, B.: Camel: Communicative agents for" mind" exploration of large language model society. Advances in Neural Information Processing Systems **36**, 51991–52008 (2023)

[8] Farquhar, S., Kossen, J., Kuhn, L., Gal, Y.: Detecting hallucinations in large language models using semantic entropy. Nature **630**(8017), 625–630 (2024)

[9] Wolfram, S.: Can AI solve science? https://writings.stephenwolfram.com/2024/03/can-ai-solve-science. Stephen Wolfram Writings (2024)

[10] Gao, H., Sun, L., Wang, J.-X.: Phygeonet: Physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state pdes on irregular domain. Journal of Computational Physics **428**, 110079 (2021)

[11] Du, P., Parikh, M.H., Fan, X., Liu, X.-Y., Wang, J.-X.: Conditional neural field latent diffusion model for generating spatiotemporal turbulence. Nature Communications **15**(1), 10416 (2024)

[12] Duraisamy, K., Iaccarino, G., Xiao, H.: Turbulence modeling in the age of data. Annual review of fluid mechanics **51**(1), 357–377 (2019)

[13] Brunton, S.L., Noack, B.R., Koumoutsakos, P.: Machine learning for fluid mechanics. Annual review of fluid mechanics **52**(1), 477–508 (2020)

[14] Kashefi, A.: Kolmogorov–arnold pointnet: Deep learning for prediction of fluid fields on irregular geometries. Computer Methods in Applied Mechanics and Engineering **439**, 117888 (2025)

[15] Yuan, Y., Lozano-Durán, A.: Dimensionless learning based on information. arXiv preprint arXiv:2504.03927 (2025)

[16] Zhang, X., Xu, Z., Zhu, G., Tay, C.M.J., Cui, Y., Khoo, B.C., Zhu, L.: Using large language models for parametric shape optimization. arXiv preprint arXiv:2412.08072 (2024)

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

[18] Pandey, S., Xu, R., Wang, W., Chu, X.: Openfoamgpt: A retrieval-augmented large language model (llm) agent for openfoam-based computational fluid dynamics. Physics of Fluids **37**(3), 035120 (2025)

[19] Du, M., Chen, Y., Wang, Z., Nie, L., Zhang, D.: Large language models for automatic equation discovery of nonlinear dynamics. Physics of Fluids **36**(9), 097121 (2024)

[20] Nichol, A.Q., Dhariwal, P.: Improved denoising diffusion probabilistic models. In: International Conference on Machine Learning, pp. 8162–8171 (2021). PMLR

[21] Jain, A., Mildenhall, B., Barron, J.T., Abbeel, P., Poole, B.: 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, Louisiana, pp. 867–876 (2022)

[22] Lin, C.-H., Gao, J., Tang, L., Takikawa, T., Zeng, X., Huang, X., Kreis, K., Fidler, S., Liu, M.-Y., Lin, T.-Y.: Magic3D: high-resolution text-to-3D content creation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, USA, pp. 300–309 (2023)

[23] Sanghi, A., Chu, H., Lambourne, J.G., Wang, Y., Cheng, C.-Y., Fumero, M., Malekshan, K.R.: Clip-forge: Towards zero-shot text-to-shape generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 18603–18613 (2022)

[24] Mittal, P., Cheng, Y.-C., Singh, M., Tulsiani, S.: Autosdf: Shape priors for 3D

completion, reconstruction and generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 306–315 (2022)

[25] Fu, R., Zhan, X., Chen, Y., Ritchie, D., Sridhar, S.: Shapecrafter: A recursive text-conditioned 3D shape generation model. In: Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., Oh, A. (eds.) Advances in Neural Information Processing Systems, vol. 35, pp. 8882–8895. Curran Associates, Inc., New Orleans, USA (2022)

[26] Vahdat, A., Williams, F., Gojcic, Z., Litany, O., Fidler, S., Kreis, K., *et al.*: Lion: Latent point diffusion models for 3D shape generation. Advances in Neural Information Processing Systems **35**, 10021–10039 (2022)

[27] Sanghi, A., Fu, R., Liu, V., Willis, K.D., Shayani, H., Khasahmadi, A.H., Sridhar, S., Ritchie, D.: 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, pp. 18339–18348 (2023)

[28] Nichol, A., Jun, H., Dhariwal, P., Mishkin, P., Chen, M.: Point-e: A system for generating 3D point clouds from complex prompts. arXiv preprint arXiv:2212.08751 (2022)

[29] Peskin, C.S.: Flow patterns around heart valves: A numerical method. Journal of Computational Physics **10**(2), 252–271 (1972)

[30] Peskin, C.S.: The immersed boundary method. Acta Numerica **11**, 479–517 (2002)

[31] Mittal, R., Iaccarino, G.: Immersed boundary methods. Annual Review of Fluid Mechanics **37**, 239–261 (2005)

[32] Mohd-Yusof, J.: For simulations of flow in complex geometries. Annual research briefs **317**, 35 (1997)

[33] Fadlun, E.A., Verzicco, R., Orlandi, P., Mohd-Yusof, J.: Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations. Journal of Computational Physics **161**(1), 35–60 (2000)

[34] Iaccarino, G., Verzicco, R.: Immersed boundary technique for turbulent flow simulations. Applied Mechanics Reviews **56**(3), 331–347 (2003)

[35] Kim, J., Kim, D., Choi, H.: An immersed-boundary finite-volume method for simulations of flow in complex geometries. Journal of Computational Physics **171**(1), 132–150 (2001)

[36] Yang, J., Balaras, E.: An embedded-boundary formulation for large-eddy simulation of turbulent flows interacting with moving boundaries. Journal of

Computational Physics **215**(1), 12–40 (2006)

[37] Zhang, N., Zheng, Z.C.: An improved direct-forcing immersed-boundary method for finite difference applications. Journal of Computational Physics **221**(1), 250–268 (2007)

[38] Uhlmann, M.: An immersed boundary method with direct forcing for the simulation of particulate flows. Journal of Computational Physics **209**(2), 448–476 (2005)

[39] Taira, K., Colonius, T.: The immersed boundary method: A projection approach. Journal of Computational Physics **225**(2), 2118–2137 (2007)

[40] Yang, X., Zhang, X., Li, Z., He, G.-W.: A smoothing technique for discrete delta functions with application to immersed boundary method in moving boundary simulations. Journal of Computational Physics **228**(20), 7821–7836 (2009)

[41] Wang, S., Zhang, X.: An immersed boundary method based on discrete stream function formulation for two- and three-dimensional incompressible flows. Journal of Computational Physics **230**(9), 3479–3499 (2011)

[42] Chen, Y., Liu, Y., Wang, S.: Streamwise ram effect and tip vortex enhance the lift of a butterfly-inspired flapping wing. Journal of Fluid Mechanics **1005**, 13 (2025)

[43] Wang, C., Xu, Z., Zhang, X., Wang, S.: Bayesian optimization for the spanwise oscillation of a gliding flat plate. Optimization and Engineering **24**, 2763–2772 (2023)

[44] Zhou, Z., Xu, Z., Wang, S., *et al.*: Wall-modeled large-eddy simulation of noise generated by turbulence around an appended axisymmetric body of revolution. Journal of Hydrodynamics **34**, 533–554 (2022)

[45] Wang, C., Xu, Z., Zhang, X., Wang, S.: Optimal reduced frequency for the power efficiency of a flat plate gliding with spanwise oscillations. Physics of Fluids **33**(11), 111908 (2021)

[46] Wang, S., He, G., Zhang, X.: Parallel computing strategy for a flow solver based on immersed boundary method and discrete stream-function formulation. Computers & Fluids **88**, 210–224 (2013)

[47] Johnson, T.A., Patel, V.C.: Flow past a sphere up to a reynolds number of 300. Journal of Fluid Mechanics **378**, 19–70 (1999)

[48] Streamlit: Streamlit: A faster way to build and share data apps. faster way to build and share data apps (2024). Accessed: 2025-04-27

[49] Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. arXiv preprint arXiv:2006.11239 (2020)

Acknowledgements

This work was supported by the NSFC Excellence Research Group Program for 'Multi-scale Problems in Nonlinear Mechanics' (No. 12588201), the National Natural Science Foundation of China (Nos. 12425207 and 92252203), 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 contributions

Z.X., G.H, S.W. and H.-D.Y. conceived the work. Z.X., S.W. and H.-D.Y. performed the system development, numerical simulations and results analysis. Z.X., L.W., Q.L., C.W., Y.C. and H.-D.Y. wrote the manuscript. All authors contributed to the analysis of the results.

Competing interests

The authors declare no competing interests.