RESEARCH-ARTICLE

A Highly-Efficient Hybrid Simulation System for Flight Controller Design and Evaluation of Unmanned Aerial Vehicles

JIWEI WANG, ShanghaiTech University, Shanghai, China

WENBIN SONG, ShanghaiTech University, Shanghai, China

YICHENG FAN, ShanghaiTech University, Shanghai, China

YANG WANG, ShanghaiTech University, Shanghai, China

XIAOPEI LIU, ShanghaiTech University, Shanghai, China

Published: 04 December 2025
Accepted: 09 August 2025
Received: 24 May 2025

A Highly-Efficient Hybrid Simulation System for Flight Controller Design and Evaluation of Unmanned Aerial Vehicles

JIWEI WANG, ShanghaiTech University, China
WENBIN SONG, ShanghaiTech University, China
YICHENG FAN, ShanghaiTech University, China
YANG WANG*, ShanghaiTech University, China
XIAOPEI LIU†, ShanghaiTech University, China

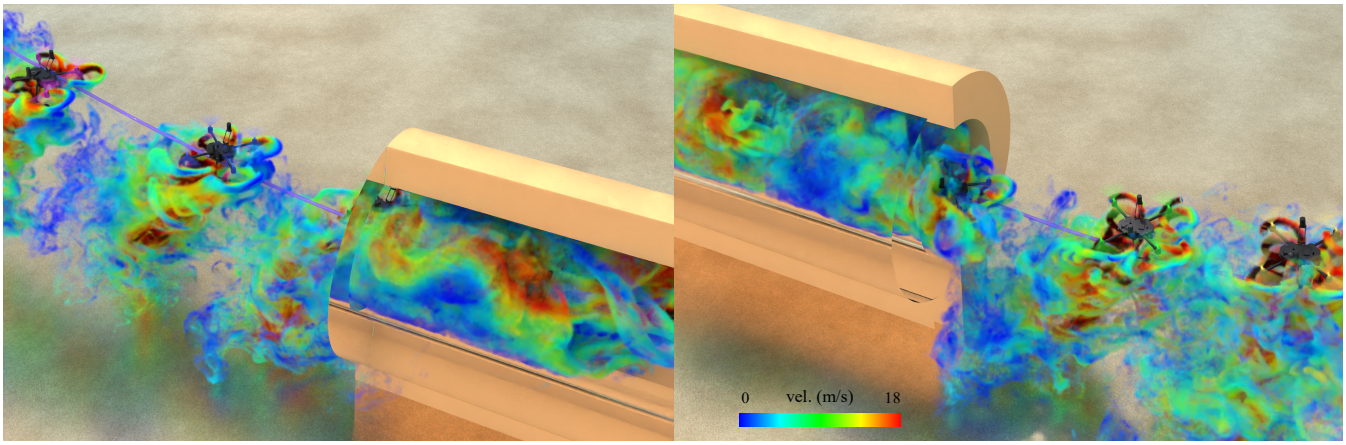

Fig. 1. **A quadcopter swarm executing sequential flight through a cylindrical tube.** Our highly-efficient hybrid simulation system enables controller design and evaluation for single or multi-UAV flight scenarios under strong surrounding fluid-structure interactions, in which the traditional empirical model becomes unreliable. This figure depicts a swarm of five quadcopters entering (left) and exiting (right) a narrow cylindrical tube, where the near-wall airflow effects may significantly impact the stability of flight control. With our simulator, flight controllers can be better optimized and evaluated in a more versatile fluid-interactive environment. To visualize airflow dynamics, smoke particles are injected near the rotors and color-mapped by the airflow velocity magnitude.

Unmanned aerial vehicles (UAVs) have demonstrated remarkable efficacy across diverse fields. Nevertheless, developing flight controllers tailored to a specific UAV design, particularly in environments with strong fluid-interactive dynamics, remains challenging. Conventional controller design experiences often fall short in such cases, rendering it infeasible to apply time-tested practices. Consequently, a simulation test bed becomes indispensable for controller design and evaluation prior to its actual implementation on the physical UAV. This platform should allow for meticulous adjustment of controllers and should be able to transfer to real-world systems without significant performance degradation. Existing simulators predominantly hinge on empirical models due to high efficiency, often overlooking the dynamic interplay between the UAV and the surrounding airflow. This makes it difficult to mimic more complex flight maneuvers, such as an abrupt mid-air halt inside narrow channels, in which the UAV may experience strong fluid-structure interactions. On the other hand, simulators considering the complex surrounding airflow are extremely slow and inadequate to support the design and evaluation of flight controllers. In this paper, we present a novel remedy for highly-efficient UAV flight simulations, which entails a hybrid modeling that deftly combines our novel far-field adaptive block-based fluid simulator with parametric empirical models situated near the boundary of the UAV, with the model parameters automatically calibrated. With this newly devised simulator, a broader spectrum of flight scenarios can be explored for controller design and assessment, encompassing those influenced by potent close-proximity effects, or situations where multiple UAVs operate in close quarters. The practical worth of our simulator has been authenticated through comparisons with actual UAV flight data. We further showcase its utility in designing flight controllers for fixed-wing, multi-rotor, and hybrid UAVs, and even exemplify its application when multiple UAVs are involved, underlining the unique value of our system for flight controllers.

CCS Concepts: • **Computing methodologies** → **Physical simulation**; **Animation**; • **Computer systems organization** → *Robotics*.

Additional Key Words and Phrases: Unmanned Aerial Vehicles, Computational Fluid Dynamics, Fluid-structure Interaction, Empirical Modeling, Model Parameter Optimization, Control Algorithms

ACM Reference Format:
Jiwei Wang, Wenbin Song, Yicheng Fan, Yang Wang, and Xiaopei Liu. 2025. A Highly-Efficient Hybrid Simulation System for Flight Controller Design and Evaluation of Unmanned Aerial Vehicles. *ACM Trans. Graph.* 44, 6, Article 198 (December 2025), 23 pages. https://doi.org/10.1145/3763283

† Corresponding author; * Co-corresponding author.
Authors' Contact Information: Jiwei Wang, ShanghaiTech University, Shanghai, China, wjw1190@126.com; Wenbin Song, ShanghaiTech University, Shanghai, China, songwb@shanghaitech.edu.cn; Yicheng Fan, ShanghaiTech University, Shanghai, China, fanych1@shanghaitech.edu.cn; Yang Wang, ShanghaiTech University, Shanghai, China, wangyang0864@163.com; Xiaopei Liu, ShanghaiTech University, Shanghai, China, aurorean.xp@gmail.com.

ACM 1557-7368/2025/12-ART198
https://doi.org/10.1145/3763283

1 Introduction

Unmanned aerial vehicles (UAVs) have carved out an expansive footprint across diverse fields. Prominent applications span aerial photography and videography [Klosterman et al. 2018], agriculture [Tsouros et al. 2019], delivery services [Chiang et al. 2019], surveying and mapping [Lucieer et al. 2014; Zhou et al. 2020], environmental monitoring [Asadzadeh et al. 2022], infrastructure inspection [Máthé and Buşoniu 2015], as well as military use [Gargalakos 2024], to name just a few. Over the years, the landscape of UAV configurations has evolved dramatically. Fixed-wing UAVs [Cai et al. 2010], multi-rotor UAVs [Sabour et al. 2023], and hybrids that combine fixed-wing and multi-rotor designs [Xiao et al. 2021] are now widely available. This proliferation of UAV shape and structure designs underscores a clear trajectory towards enhanced versatility. Despite these advancements, automatically obtaining a controller to ensure stable and desired flight remains a formidable challenge, especially for UAVs with atypical designs. The intricate interaction between the UAV and the often-turbulent ambient airflow exacerbates this complexity. Thus, testing a specific flight controller on a particular physical UAV in such scenarios is not only prohibitively expensive and time-consuming but can also pose significant risks.

To address the challenges described above, efficient and physically consistent simulation platform emerges as a crucial tool. It enables the efficient derivation and optimization of flight controllers tailored to specific UAV designs in various environments. The most commonly employed simulation model is the algebraic empirical model [Quan 2017], which calculates the aerodynamic forces acting on a UAV by leveraging the state of the vehicle. Its parameters are calibrated either through physical experiments or high-precision computational fluid dynamics (CFD) simulations. While the algebraic empirical model has achieved substantial success, its applicability is largely limited to UAVs in a stable cruising state, which is free from influence of surrounding turbulent airflow. In scenarios where obstacles are in close proximity—for instance, when ground or wall effects are pronounced, or when multiple UAVs are operating in confined spaces—these traditional models often prove inadequate.

In principle, a comprehensive direct fluid-structure interaction (FSI) simulation is needed, in which CFD and rigid-body solvers are coupled to calculate aerodynamic forces acting on a UAV in a wide range of scenarios [Akinci et al. 2012; Li et al. 2020; Robinson-Mosher et al. 2008]. Despite its theoretical appeal, this method is hampered by its low efficiency of many existing CFD solvers. To maintain accuracy for surface force calculation, the mesh near the boundary of the UAV must be highly refined. Even with the aid of parallel devices such as GPUs [Jespersen 2010], and algorithmic enhancements like the local and explicit lattice Boltzmann (LB) solvers with multi-resolution schemes [Liu et al. 2025; Liu and Liu 2023; Lyu et al. 2023], these simulation methods remain unsuitable for flight controller design due to their high computational demands.

In this paper, we introduce a highly-efficient hybrid solver tailored for simulating fluid-structure interactions (FSI), with the aim of facilitating flight controller design and evaluation. Our approach is grounded in a two-pronged strategy. In the simulation domain, for regions in the immediate vicinity of the UAV's body, we utilize localized parametric algebraic empirical models, which capture surface forces more accurately while maintaining computational efficiency than direct FSI solvers, obviating the need for extensive boundary refinement. Conversely, for regions distant from the UAV's boundary, we deploy a GPU-optimized LB solver due to its efficiency in handling unsteady, time-dependent air flows, as well as its low dissipation and dispersion property for treating turbulence. More significantly, we propose a novel adaptive block-based method, which enables the simulation domain to automatically track the movement of UAVs and their wake flows, effectively reducing computational overload—an advantage that is particularly pronounced in multi-UAV scenarios. Our adaptive block-based LB solver is systematically coupled with our parametric algebraic empirical models near the boundary of the UAV, forming an integrated hybrid solver that demonstrates remarkable simulation efficiency while preserving physically consistent results. The model parameters of our simulator can also be automatically calibrated by numerical optimization techniques using the measurement data from physical UAVs. In single-UAV flight simulations, it can achieve real-time performance for some scenarios, whereas in multi-UAV setups, it can operate at interactive speeds. As far as we are aware, this represents the first physically consistent FSI solver capable of delivering *real-time* or *interactive* performance, which paves a new way for the design and evaluation of flight controllers that can account for a wide variety of real-world applications in a unified manner.

Our simulator has been implemented to form a comprehensive system. By calibrating with the sensor data collected from the real-world flights, we can align the behavior of our simulator with the physical UAV. Our simulator has been validated against various real-flight data recorded under the same operating condition, showing a high degree of congruence with the physical system. To prove the effectiveness of our simulator in designing and evaluating flight controllers for diverse UAVs, we present flight results for fixed-wing, multi-rotor, and hybrid configurations, under various conditions of strong fluid-structure interactions, in which the flight controllers were iteratively optimized using our system. To further highlight the practical applicability of our system, we successfully transferred flight controllers developed within our simulator directly to a real-world UAV system without additional fine-tuning, underscoring the unique strength of our system in enabling a new paradigm for efficient seamless sim-to-real flight controller design. Fig. 1 illustrates a quadcopter swarm performing sequential flight through a narrow cylindrical tube, where the motion of the quadcopters is severely affected by the near-wall airflow, which necessitates meticulous controller design to ensure stable and safe flight.

2 Background and Related Work

Before presenting our hybrid fluid-structure interaction simulator, we will first provide a concise review of the modeling of the UAV and the calculation of surface forces exerted by the surrounding airflow through the use of empirical models as well as the direct fluid-structure interaction solvers. Following this overview, we will review some celebrated parameter optimization methods for simulators, as well as the literature for designing flight controllers. Finally, we will elaborate on our contributions to the field.

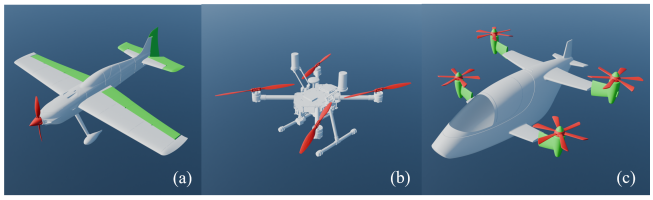

Fig. 2. **Different types of UAV configurations.** In this paper, three distinct types of UAV configurations are considered, with movable components colored differently from white: (a) a fixed-wing UAV; (b) a quadrotor UAV; and (c) a tilt-wing vertical take-off and landing UAV.

2.1 Modeling of UAV dynamics

Consider a UAV composed of multiple interconnected rigid bodies (see, e.g., Fig. 2 for different configurations). To accurately simulate the dynamics of UAV, a general model calculates the accelerations of the whole body at any given instant. Then, by applying Newton's laws of motion and Euler's equations for rotation, these accelerations are integrated to obtain the velocities and positions by:

$$\boldsymbol{a} = M^{-1}\boldsymbol{F}_{net}, \qquad \boldsymbol{\alpha} = \boldsymbol{I}^{-1}(\boldsymbol{\tau}_{net} - \boldsymbol{\omega} \times \boldsymbol{I}\boldsymbol{\omega}), \tag{1}$$

where $\boldsymbol{I}$, $\boldsymbol{\omega}$, $\boldsymbol{a}$, $\boldsymbol{\alpha}$, $\boldsymbol{F}_{net}$ and $\boldsymbol{\tau}_{net}$ are the moment of inertia, angular velocity, linear and angular accelerations, as well as the net force and torque, respectively, and M is the total mass of the UAV. The net force and torque can be further expressed as:

$$\boldsymbol{F}_{net} = \int_V \boldsymbol{F}_b(v)dv + \oint_S \boldsymbol{F}_c(s)ds, \tag{2}$$

$$\boldsymbol{\tau}_{net} = \int_V \boldsymbol{r}(v) \times \boldsymbol{F}_b(v)dv + \oint_S \boldsymbol{r}(s) \times \boldsymbol{F}_c(s)ds, \tag{3}$$

in which, volumetric force on body $\boldsymbol{F}_b$, such as the gravity force, are integrated over the body volume, and surface force by external contacts $\boldsymbol{F}_c$, such as friction and pressure, are integrated over the entire body surface. Torque is calculated in a similar manner but is combined with a displacement vector $\boldsymbol{r}$ to the center of mass. For aerial vehicles, the external surface forces predominantly originate from the aerodynamic force. It should be noted that determining the motion of a UAV using Eq. (1) is nontrivial even after obtaining the net force and torque, since the center of mass and moment of inertia of the UAV can vary over time due to moving components, such as the flaps in a fixed-wing aircraft or the rotating rotors in a hybrid one. To simplify the calculations, UAVs are typically modeled as a multiple rigid-body system by assuming that solid deformation is negligible. For each component, the center of mass and the moment of inertia remain constant in its local coordinate system and can be pre-computed. The components are connected via joints, with revolute and prismatic joints being the most basic types. The overall motion of the system can be determined by solving the rigid-body dynamics for each component independently, in which constraints are applied to enforce the joint properties [Wittenburg 2013]. Notably, Featherstone developed efficient algorithms for articulated rigid-body systems, which have been implemented in various open-source libraries [Felis 2017; Lee et al. 2018].

2.2 Empirical models for aerodynamic impact

To accurately reproduce UAV dynamics over time via Eq. (1), precise calculation of aerodynamic forces around the body is essential. However, this requires solving the fluid-structure interaction (FSI) problem, which is computationally challenging. In the literature, most UAV dynamic models simplify this by avoiding direct FSI solutions. Instead, aerodynamic forces are approximated by algebraic empirical models with angle-dependent coefficients derived from either experiments [Chowdhary and Jategaonkar 2010; Klein 1989] or high-fidelity FSI simulations [Da Ronch et al. 2011; Vallespin et al. 2012]. For common UAV configurations (e.g., fixed-wings and multi-copters), empirical models for cruise conditions have been well studied [Quan 2017; Stevens et al. 2015] and are widely used in design and analysis. Typically, empirical models decompose force estimation into components like body/wing aerodynamics and propulsion forces (e.g., rotor thrust). For instance, Xu et al. modeled forces on a hybrid UAV as gravity, propeller thrust, wing lift, and drag. Thrust is often approximated via a motor speed mapping, while wing forces use lift/drag coefficients with flat-plane models [Cory and Tedrake 2008]. For full-scale UAV applications, body drag is typically incorporated using drag coefficients derived from CFD simulations. There are also alternative methods in both computer graphics [Soliman et al. 2024] and UAV design [Xie et al. 2018] that leverage pre-processed geometric data, often based on surface samples or facets, to improve the estimation of body movement. These methods can account for turbulence effects at UAV boundaries and adapt to the changes of the UAV shape while retaining efficiency. Nevertheless, deriving a universal model applicable to all UAV types remains extremely difficult, if not outright impossible. Moreover, their reliability is constrained by the challenges posed by calibrating model parameters [Al-Shabi et al. 2013]. It should be noted that these empirical models are reliable only when all model parameters are correctly configured and when the UAV operates in an open space with no nearby obstacles. However, they prove ineffective when strong aerodynamic coupling with surrounding objects, such as ground, ceiling, and wall effects—referred to as "unconventional cases"—occurs [Gao et al. 2019; Matus-Vargas et al. 2021; Paz et al. 2020]. In such cases, additional models are needed to account for the impact of obstacle proximity. For instance, Rozhdestvensky reviewed the mathematical modeling of ground effects for fixed-wing UAVs. Danjun et al. experimentally modeled the ground effects of multicopters to optimize autonomous landing strategies, while Sanchez-Cuevas et al. analyzed multicopter ground effects and reported on partial ground effects—phenomena that arise when a flat obstacle is present only beneath some of the rotors.

2.3 Fluid-structure interaction for aerodynamic impact

Although empirical models can be devised to consider the influence of the airflow around UAVs, their general application across all UAV types remains challenging, necessitating direct fluid-structure interaction (FSI) simulations. In FSI problems, the incompressible or weakly compressible Navier-Stokes (NS) equations are numerically solved. In the field of Computational Fluid Dynamics (CFD), common methods include the finite difference method (FDM) [Ferziger and Perić 2002; Shukla et al. 2007; Strikwerda 2004], the finite volume

method (FVM) [Eymard et al. 2000; Jasak and Uroić 2020; Moukalled et al. 2016], and the finite element method (FEM) [Anderson and Wendt 1995; Szabó and Babuška 2021; Zienkiewicz et al. 2013]. Although FDM is relatively straightforward to implement, it is typically formulated on Cartesian grids and often requires dealing with cut-cells, which can compromise its accuracy [Brady and Livescu 2021]. In contrast, both FVM and FEM permit the use of body-fitted meshes, enabling more accurate calculations of surface forces at solid boundaries [Tu et al. 2023]. However, these methods are severely constrained by the computationally expensive process of mesh construction as well as frequent remeshing in a dynamic scenario [Chawner and Taylor 2019]. Coupled with the high-resolution requirement near the solid boundary to obtain accurate physical quantities, traditional CFD solvers are ill-suited for FSI simulations aimed at designing flight controllers, as these simulations typically demand fast or even real-time performance.

On the other hand, simulation methods developed in computer graphics prioritize higher efficiency, stability, and flexibility, making them more suitable for FSI simulations of UAVs. Originating from the early stable fluid method [Stam 1999], grid-based fluid solvers have advanced significantly with the incorporation of higher-order advection schemes [Kim et al. 2005; Qu et al. 2019; Selle et al. 2008; Zehnder et al. 2018]. In addition to grid-based solvers, particle-based [Adams et al. 2007; Becker and Teschner 2007; Desbrun and Gascuel 1996; Ihmsen et al. 2014; Müller et al. 2003] and hybrid [Fu et al. 2017; Jiang et al. 2015; Raveendran et al. 2011; Tao et al. 2022] methods have emerged and are better adapted to liquid flows. Vortex methods, which reformulate the NS equations in terms of vorticity [Park and Kim 2005; Selle et al. 2005], are particularly effective at preserving flow details. Nevertheless, their application to solve FSI problems remains largely unexplored. Although these computer graphics techniques can simulate fluid-structure interactions, they generally feature a coarse resolution near solid boundaries and cannot ensure physical consistency. Moreover, simulation methods in both CFD and computer graphics require solving global sparse linear systems or impose strict time-step constraints, thereby limiting their efficiency even when GPU-accelerated. The nonlinear advection term often necessitates very high-order methods to mitigate numerical diffusion, particularly in turbulent flows, which further diminishes the efficiency of simulation in practical scenarios.

The limitations of the traditional NS solvers can partially be overcome by the lattice Boltzmann (LB) solvers [Krüger et al. 2017; Rinaldi et al. 2012]. Thanks to their local dynamics nature, which feature low dissipation and dispersion characteristics, LB solvers are better suited for efficiently simulating real-world flow problems that often involve turbulence [Li et al. 2020; Liu and Liu 2023; Lyu et al. 2023]. In the literature, there has been a substantial body of research dedicated to using LB solvers for addressing FSI problems [Abaszadeh et al. 2022; Bouzidi et al. 2001; Ginzburg and d'Humieres 2003; Ladd 1994; Marson et al. 2021; Rinaldi et al. 2012; Tao et al. 2018; Wu and Shu 2009; Xu et al. 2013; Yu et al. 2003; Zhao and Yong 2017]. Specifically within the realm of computer graphics, LB solvers have been integrated with the immersed-boundary (IB) method [Chen et al. 2021; Li et al. 2018, 2020], various bounce-back schemes [Li et al. 2022; Liu and Liu 2023; Lyu et al. 2023, 2021], as well as overset-grid method [Xiao et al. 2025], to tackle FSI problems. These integrations also incorporate GPU accelerations to ensure high computational efficiency. As indicated in previous studies [Chen et al. 2021; Li et al. 2020], a GPU-accelerated LB solver has demonstrated the capability to achieve speeds tens of times faster than a GPU-based NS solver under similar accuracy. However, despite the significant enhancement in simulation efficiency, it remains challenging to satisfy the efficiency demands for flight controller design while simultaneously maintaining adequate physical accuracy.

2.4 Local-domain fluid-structure interaction solvers

Previous FSI simulators have functioned within a domain of finite size, which is unsuitable for simulating the UAV dynamics whose motion typically spans a significantly large area. Consequently, local-domain FSI solvers are of crucial importance in this context. These solvers utilize a finite domain that adapts to and tracks the movement of the UAV, making them more appropriate to simulate UAV dynamics for flight controller design and evaluation. Current local-domain solvers can be broadly categorized into three main types. Moving reference frame techniques maintain physical consistency by implementing inertial force corrections [Auguste et al. 2013; Camargo et al. 2012; Chrust et al. 2013; Li et al. 2002; Mbanjwa et al. 2022; Takeuchi et al. 2006]. Adaptive domain methods dynamically create or eliminate simulation grids based on the position of the central entity, thus keeping the overall computational size almost constant [Chen et al. 2019; Tan et al. 2011]. Overset or overlapping grid strategies incorporate refined local grids within a coarser background grid [English et al. 2013a,b; Huang et al. 2021]. However, their reliance on a global grid restricts their utility for completely unbounded-domain simulations. Despite these advancements, these solvers are generally founded on incompressible NS solvers, which may exhibit lower computational efficiency. Moreover, scaling them to accommodate multi-UAV systems remains a formidable challenge. Alternative sparse NS solvers [Aanjaneya et al. 2017; Bailey et al. 2015; Qiu et al. 2023; Raateland et al. 2022; Setaluri et al. 2014; Wu et al. 2018] allocate computational resources dynamically as required. However, they inherit the limitations of NS solvers and necessitate complex GPU-oriented data structures [Hoetzlein 2016; Raateland et al. 2022]. Additionally, they are burdened by the extra overhead stemming from frequent updates of sparse fields [Wu et al. 2018]. Instead of depending on NS solvers, the "FishGym" framework [Liu et al. 2022] and its variant [Song et al. 2025] employ a GPU-optimized LB solver to enhance both efficiency and accuracy. Nevertheless, they fail to efficiently adapt to more general scenarios including the support of multi-UAV flight simulations. It is crucial to note that all of the aforementioned local-domain solvers struggle to achieve physically consistent results comparable to experiments, as they do not incorporate appropriate modeling for the airflow near the boundary of the UAV due to turbulence in practice. As a result, flight controllers derived from these simulators may face significant sim-to-real gap when directly implemented on real-world UAVs.

2.5 Simulation optimization

Both empirical models and direct FSI solvers may involve model parameters that need to be determined, particularly for models employed in the vicinity of solid boundaries. These parameters are

typically ascertained through a manual trial-and-error approach, a process that is both laborious and time-consuming. To circumvent this arduous process, optimization methods are utilized in simulations to automatically identify the optimal model parameters. Such an approach aims to minimize the discrepancy between the simulation results and physical observations [Tahmasebi et al. 2012; Villaverde et al. 2022]. Traditional simulation optimization algorithms can be classified into local and global algorithms, as discussed in Venter [2010]. Most local algorithms leverage gradients to effectively converge to a local optimum. Examples of such local algorithms include Newton's method [Moré and Sorensen 1982] and conjugate gradient method [Fletcher and Reeves 1964]. On the other hand, global algorithms possess the ability to identify global optima, although they come with the drawback of higher computational expenses. Evolutionary algorithms [Yu and Gen 2010] represent a class of global optimization methods that do not require gradient information. This characteristic has rendered them a popular option for a wide range of optimization tasks. Among the most well-known evolutionary algorithms are the Genetic Algorithm (GA) [Holland 1975] and the Particle Swarm Optimization (PSO) [Kennedy and Eberhart 1995]. With gradient-based local methods, stochastic methods are developed to help escape local optima. The earliest stochastic method is stochastic gradient descent (SGD) [Amari 1993]. While it is straightforward, it necessitates the specification of hyper-parameters. Adaptive moment estimation (Adam) [Kingma and Ba 2015] is a highly popular stochastic optimizer that modifies hyper-parameters adaptively based on changes in the gradient, thereby minimizing the requirement for manual parameter tuning. In the case of differentiable systems, such as neural networks and differentiable simulators [Du et al. 2021; Holl et al. 2020; Macklin et al. 2020], the gradients of parameters can be obtained through back-propagation by applying the chain rule. These systems often possess thousands of parameters, yet all the gradients can be computed with reasonable efficiency. Conversely, for non-differentiable systems, gradients are typically determined using finite difference methods for each individual parameter [Berahas et al. 2022]. This approach leads to the execution of the system a number of times that is proportional to the quantity of parameters. As a result, the overall dimension of the parameters being optimized is restricted due to the inefficiency inherent in this process.

2.6 Flight controller

In order to enable the desired motion of a UAV, whether in simulation or during actual flight, a flight controller is essential. The flight control of UAVs has advanced considerably over the past few decades. Early developments centered on classical control methods, particularly the simple yet effective proportional–integral–derivative (PID) loops [Lopez-Sanchez and Moreno-Valenzuela 2023] (cascaded PID controllers), which remain the most widely used baseline for attitude and altitude stabilization due to their simplicity and reliability. Subsequently, various PID variants, such as nonlinear PID [Bouzid et al. 2017; Goodarzi et al. 2013; Tuan and Won 2013], adaptive PID [He et al. 2017; Yang et al. 2013], and fractional-order PID [Jiahe and Rui 2015], were introduced to improve robustness against disturbances and modeling uncertainties. Despite their practicality, PID controllers struggle with multivariable systems and lack capabilities to achieve optimal performance. To address these limitations, model-based optimal control methods were proposed; for instance, the Linear Quadratic Regulator (LQR) was applied to linearized UAV dynamic models, offering superior performance by explicitly minimizing a quadratic cost function [Anjali et al. 2016; Khatoon et al. 2014; Saraf et al. 2020]. Building on this, Model Predictive Control (MPC) has been explored for trajectory optimization, balancing performance with constraint handling [Ganga and Dharmana 2017; Hanover et al. 2021]. More recently, data-driven learning approaches have gained traction, aiming to enhance robustness in the face of nonlinearities and environmental uncertainties. Early work utilized feedforward neural networks to approximate UAV dynamics and generate control actions for off-nominal maneuvers [Bansal et al. 2016]. Current research include end-to-end neural controllers [Ferede et al. 2024] and reinforcement learning (RL)-based controllers [Kaufmann et al. 2023; Shen and Huang 2024; Song et al. 2021; Xu et al. 2019] trained in simulation, with increasing focus on addressing the sim-to-real transfer challenge to ensure real-world deployment. However, whether employing empirical PID, model-based MPC or RL approaches, developing high-performance controllers for UAVs remains challenging. Real-world controller tuning incurs substantial hardware and time costs with potential risks, while empirical simulators struggle to reproduce complex aerodynamic phenomena and bridge the sim-to-real gap. Thus, there is an urgent need for an efficient and physically consistent simulator to support the flight controller design and evaluation for future UAVs, especially for unconventional configurations.

2.7 Our contributions

In summary, empirical models and direct FSI solvers each possess their own limitations, rendering them insufficient for supporting the design and evaluation of UAV flight controllers across a broader range of scenarios. In this paper, we present a novel hybrid simulation system that tries to hit the highest computational efficiency while ensuring consistent physical behaviors with real-world UAV systems. Our paper puts forward the following contributions:

- We propose a novel hybrid FSI simulator combining parametric empirical models with a novel adaptive block-based direct FSI solver, which significantly boosts the simulation flexibility and efficiency especially for multi-UAV systems, enabling interactive or even real-time flight simulations.
- We further propose an automatic model calibration procedure, which is designed to make our simulator compatible with real-world UAV systems, and we validate our simulator using the flight data collected from real-world scenarios.
- We showcase a series of UAV flight control outcomes with flow-field rendering, and elucidate the process of obtaining optimal flight controllers, which can be successfully transferred to a real-world UAV without additional fine-tuning.

3 Our Hybrid Simulation System

To design and evaluate flight controller, the dynamics of airflow around a UAV are typically assumed subsonic and can be generally

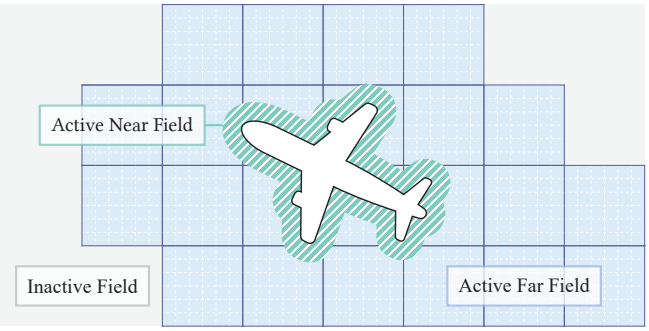

Fig. 3. **2D illustration of domain decomposition.** We decompose the entire simulation domain into a near-object field and a far field. Localized empirical models are employed for the near-object field, while an adaptive block-based LB solver is developed for the far field. Two types of solvers are coupled at the boundary of the near-object field in an immersed manner.

modeled by the following unsteady isothermal weakly compressible Navier-Stokes (NS) equations [Anderson 2010]:

$$\begin{gathered}\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}) = 0, \\ \frac{\partial (\rho \boldsymbol{u})}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}\boldsymbol{u}) = -\nabla p + \nabla \cdot \boldsymbol{\sigma} + \boldsymbol{F}, \\ p = \rho R T_0,\end{gathered} \tag{4}$$

where $\boldsymbol{u}$, ρ, and p represent the fluid velocity, density, and pressure fields, respectively; $\boldsymbol{F} = \rho \boldsymbol{a}$ is the external force field, with $\boldsymbol{a}$ denoting its corresponding acceleration. T_0 signifies the constant environmental temperature; R is the specific gas constant; t denotes time; and $\boldsymbol{\sigma}$ is the shear stress tensor, defined by the linear constitutive law as [Batchelor 2000]:

$$\boldsymbol{\sigma} = 2\rho\nu \boldsymbol{S} - \rho\nu' (\nabla \cdot \boldsymbol{u}) \boldsymbol{I}, \tag{5}$$

where ν denotes the kinematic viscosity, and ν' represents the bulk viscosity. For an ideal gas, such as the air considered in this paper, a specific relationship exists: $\nu' = 2\nu/3$. Additionally, $\boldsymbol{S} = (\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T)/2$ is defined as the strain rate tensor, which quantifies the rate of local fluid deformation, with $\boldsymbol{I}$ being the identity tensor. For FSI simulation of UAV dynamics, the fluid boundary condition at the solid surface must be specified. Ideally, the no-slip condition is satisfied [Krüger et al. 2017], and the surface force and torque are computed based on this condition. However, in practical flight scenarios, turbulent boundary layer flow near the solid surface demands additional models, such as the wall-modeled large-eddy simulation (WMLES) [Bose and Park 2018], to generate physically consistent results. A critical challenge arises from the requirement for high grid resolution near the UAV surface in such models, which significantly diminishes simulation speed and hinders its application to controller design and evaluation. This issue is particularly pronounced for rotor-based UAVs, where WMLES for fast-rotating blades imposes strict small time-step constraints, rendering the simulation impractically slow for controller development and assessment. Additionally, many fluid flow solvers are confined to fixed-size domains, thus limiting the range of flight that can be simulated.

3.1 Motivation

Aiming to develop an FSI simulator for controller design and evaluation, we necessitate that the simulator operates in an unbounded space while achieving the highest possible simulation efficiency—ideally, real-time performance. In addition, the FSI simulator is expected to possess sufficient generality to be applicable across different working scenarios. Although various local-domain solvers have been proposed that allow unbounded-domain simulations [Li et al. 2002; Liu et al. 2022], they usually suffer from insufficient accuracy due to coarse grid resolution near the UAV's boundary. They also meet difficulty in effectively handling multi-UAV flight simulations.

To address these challenges, we introduce a novel concept to form a highly-efficient hybrid simulator. Specifically, we partition the simulation domain into a near-object field and a far field, as illustrated in Fig. 3. To achieve high efficiency, we configure our fluid solver to operate with a relatively coarse resolution across the entire far field, utilizing an efficient GPU-optimized lattice Boltzmann (LB) solver [Krüger et al. 2017] instead of the traditional NS solver [Anderson and Wendt 1995; Eymard et al. 2000]. Inspired by the sparse solvers [Lesser et al. 2022; Zarifi 2020] and other works for adaptive grid refinement and coarsening [Bojsen-Hansen et al. 2021; Museth 2013; Setaluri et al. 2014; Wang et al. 2025], we develop a novel adaptive block-based LB solver. In this solver, blocks of simulation regions can be dynamically created or removed to enable more flexible moving local domains that naturally adapt to multi-UAV systems. This approach dynamically focuses computational resources on critical fluid regions, which is expected to outperform fixed moving local-domain solvers such as the LB solver presented in "FishGym" [Liu et al. 2022] and its variant [Song et al. 2025].

However, applying the same coarse grid resolution to the near-object field fails to adequately resolve boundary layer flow dynamics, resulting in inaccurate boundary force calculations that significantly affect the behavior of UAV flight dynamics. In this context, algebraic empirical models play a critical role. In this work, we employ two types of localized algebraic empirical models for the near-object field along the UAV boundary: a blade model for fast-rotating thin blades and a body model for other components of the UAV. These algebraic empirical models are systematically coupled with the far-field LB solver to ultimately form our novel hybrid FSI simulator. By eliminating the need to refine simulation grid along the UAV boundary and using an immersed coupling approach, our simulator achieves extremely high efficiency when implemented on GPU.

Note that our hybrid simulator features adjustable model parameters, which enables alignment of physical behavior with the corresponding real-world UAV system. For the design and evaluation of flight controllers—particularly when considering transferability to real-world systems—it is essential to systematically determine appropriate parameters. We propose an automatic calibration procedure to align the two systems starting from default parameters.

In the following sections, we will elaborate on the above three aspects in greater detail, providing proper justifications when needed for this novel hybrid flight simulation approach, with validations and applications presented in various scenarios.

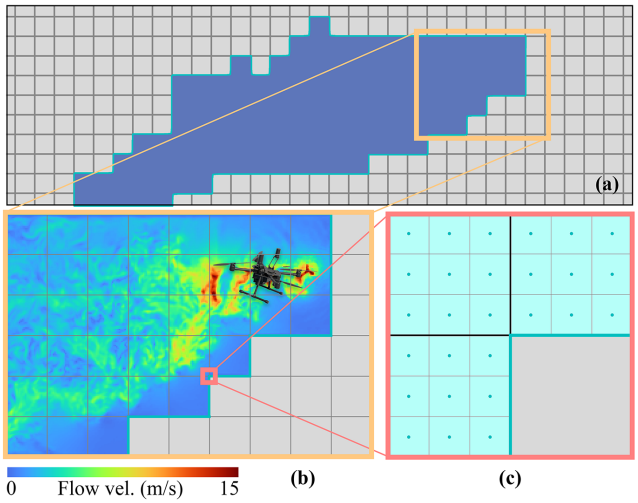

Fig. 4. **2D schematic of our adaptive block-based LB solver.** The entire computational domain is partitioned into blocks, with only those containing active flow dynamics being simulated, depicted by the blue regions in (a). A sample flow visualization and UAV configuration are shown in (b). Each block features a uniform Cartesian grid with seamless connectivity across block boundaries, as illustrated in the zoomed view (c). Nodes depicted as blue dots represent grid nodes for the LB fluid solver.

3.2 Far-field fluid simulator

Let us first consider the airflow farther from the UAV body in the far-field region. We choose to solve these dynamic airflow phenomena using the lattice Boltzmann method (LBM) [Krüger et al. 2017], leveraging its conservative local dynamics with low dissipation properties. This characteristic enables high efficiency when optimized on the GPU [Chen et al. 2021]. In LBM, a set of discrete distribution functions f_i corresponding to the discrete particle velocities $\boldsymbol{c}_i$ are employed on each lattice node $\boldsymbol{x}$ to characterize the airflow, which evolve over time according to the following dynamic equations:

$$f_i(\boldsymbol{x} + \boldsymbol{c}_i, t+1) = f_i(\boldsymbol{x}, t) + \Omega_i(\rho, \boldsymbol{u}), \tag{6}$$

where a D3Q27 lattice structure [Krüger et al. 2017] is employed, and Ω_i denotes the discretized collision operators determined by the lattice velocities, along with the macroscopic fluid density ρ and velocity $\boldsymbol{u}$. These macroscopic quantities are obtained by taking the zero-th and first order moments of the distribution functions:

$$\rho = \sum_i f_i \,, \qquad \rho \boldsymbol{u} = \sum_i \boldsymbol{c}_i f_i \,. \tag{7}$$

Eq. 6 can be solved by decomposing it into two steps: a *streaming* step, $f_i^*(\boldsymbol{x}, t) = f_i(\boldsymbol{x} - \boldsymbol{c}_i, t)$, and a *collision* step, $f_i(\boldsymbol{x}, t+1) = f_i^* + \Omega_i$. For practical scenarios involving high Reynolds numbers, more advanced collision models are required. In this paper, we employ a cumulant model [Geier et al. 2015] with adaptive high-order relaxation rates [Geier et al. 2017] to significantly reduce numerical dissipation. Turbulent eddy viscosity is also incorporated to account for sub-grid structures in turbulent flow, which is added to the kinetic viscosity. Here, we use the wall-adapting local eddy-viscosity (WALE) model [Nicoud and Ducros 1999] for this purpose.

In this paper, since we focus on UAV flight simulations, local-domain LB solvers are adopted to enable unbounded domain flight simulations. The methods proposed in "FishGym" and its variant represent potential approaches, but they struggle to adapt to multi-UAV flight scenarios where different UAVs may dynamically cluster and disperse. In such scenarios, determining a moving local domain that minimizes computational resources is challenging. Moreover, the inertial force employed in these solvers may compromise stability under large force conditions, such as when UAVs undergo significant accelerations. Our idea draws inspiration from the sparse solvers, which dynamically allocates or deletes simulation domain blocks to track flow near moving objects. This method prioritizes computational resources on critical fluid regions while eliminating the inertial force used in previous local-domain LB solvers. However, direct adaptation of the sparse solvers designed for NS equations becomes difficult for LBM as it assumes incompressibility, necessitating novel developments to enable its proper application.

3.2.1 Adaptive block-based LB solver. To maintain sufficient simulation efficiency, our local-domain LB solver first assumes an infinite domain, which is divided into non-overlapping uniform cubic blocks of size L, with a 2D illustration shown in Fig. 4 (a). Note that none of these blocks corresponds to a specific section of memory for storing fluid data initially. Within each block, an $N \times N \times N$ uniform grid is laid out, with a cubic cell size of $\Delta x = L/N$, also known as the actual spatial resolution of the solver, as depicted in Fig. 4 (c). Notably, the grid node is situated at the cell center, and the outer surfaces of the boundary cells form the boundary surface of the block. To advance the simulation, we focus solely on fluid dynamics; it should be noted that we do not consider object boundaries at this stage, as only the far-field fluid is being simulated. Since we rely on the LBM to efficiently solve far-field fluid flows, we perform the normal collision and streaming processes, respectively. While collision is executed locally at each grid node, streaming requires special treatment for boundary cells within each block. These cells are regarded as either normal cells that stream distribution functions from other blocks or domain boundary cells that require treatment for an open domain. Blocks for which memory is actually allocated for the simulation are called *active blocks*. Criteria must be specified to dynamically determine active blocks according to the flow conditions, allowing them to be adaptively created or removed, see Fig. 4 (b). For efficient execution, all blocks are pre-allocated as memory pool on the GPU prior to simulation, and an algorithm is needed to flag the active blocks used in the simulation. We will elaborate on the details of the above adaptive block-based LB solver below.

3.2.2 Boundary treatment of each block. The LB solver can be applied independently to each block, except for the cells near the block boundary, which require special treatment. There are two cases for the boundary faces of the block—faces connected to active or inactive blocks, see Fig. 5 (a). For faces adjacent to active blocks, normal streaming is performed, and the distribution functions for the grid node at the center of the boundary cells are streamed directly from the node of the boundary cells of the adjacent blocks, see Fig. 5 (b). Note that the boundary cells of two neighboring blocks are well aligned, causing the connected cells to form a larger uniform grid.

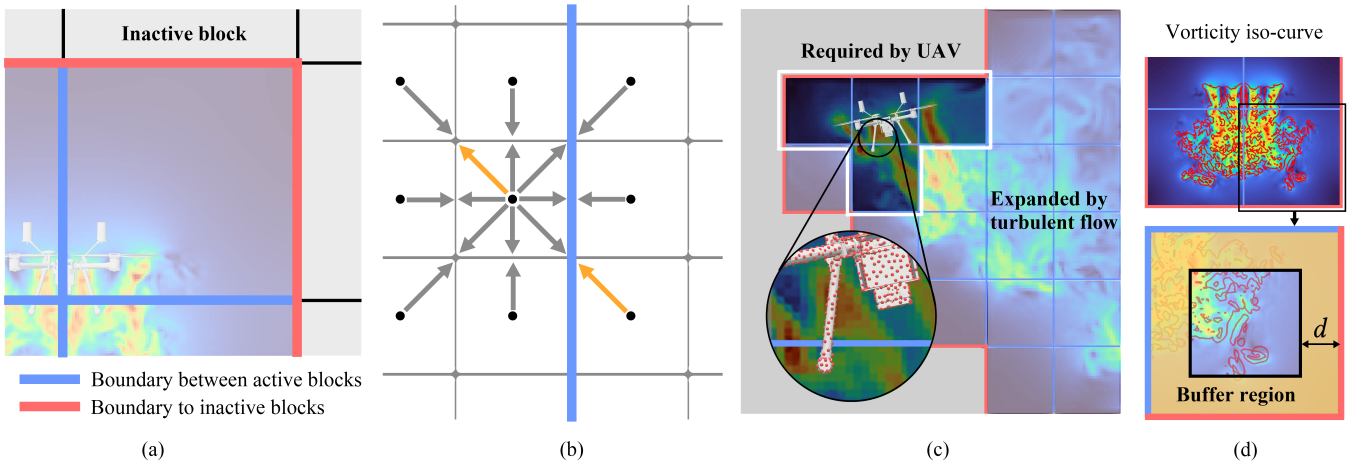

Fig. 5. **Components of our adaptive block-based LB solver.** (a) Two possible boundary types of a single block. (b) Illustration of LB streaming across block boundary. Highlighted direction refers to one of the distribution functions streamed across the block boundary and other directions are handled similarly. (c) Partition of blocks inherently required by the UAV and expanded according to the flow behavior based on local vorticity. The zoom-in view shows the surface samples on the UAV for determining required blocks. (d) Schematic of buffer region detecting turbulent flow behaviors that are of most concern.

Thus, no interpolation is needed during streaming to preserve conservativeness. However, for faces adjacent to inactive blocks, they are actually domain boundaries. Thus, a suitable domain boundary treatment should be applied. We assume the airflow can freely flow in and out of the domain boundary, and an open boundary condition is applied. Such a condition should allow for both laminar and turbulent flows. We adopt the enhanced convective boundary condition presented by Song et al. [2025], which outperforms traditional outflow conditions such as the Neumann or original convective boundary condition in handling both inflow and outflow:

$$\frac{\partial X^B}{\partial t} + \begin{cases} (\boldsymbol{u}^B \cdot \boldsymbol{n}^B)(\nabla^B X^B \cdot \boldsymbol{n}^B), & \text{if } (\boldsymbol{u}^B \cdot \boldsymbol{n}^B) \geq 0 \\ \beta(X^B - X^0), & \text{if } (\boldsymbol{u}^B \cdot \boldsymbol{n}^B) < 0 \end{cases} = 0, \tag{8}$$

where X represents macroscopic quantities (velocity $\mathbf{u}$ and density ρ); X^0 is the corresponding rest-state value; the superscript B denotes the value at boundary nodes, and $\mathbf{n}^B$ is the boundary normal. The dissipation parameter β is the decay rate of incoming quantities toward X^0. As suggested by the original authors, the dissipation parameter is set to 0.1 in our experiments. This boundary treatment method exhibits significantly better stability in cases of maneuver involving both incoming and outgoing flows.

3.2.3 Criteria for determining active blocks. As previously mentioned, only active blocks are allocated memory on the GPU. Consequently, an algorithm capable of automatically determining active blocks is required, which can be subdivided into two distinct types. For airflow around a UAV, active blocks are inherently required (see Fig. 5 (c)) to ensure that the UAV remains fully immersed in the air domain. Additionally, the airflow near the boundary of the UAV is critical, as it initiates turbulent wake flow and influences surface forces exerted by the air on the UAV body. To achieve this, we first sample uniformly over the UAV body using Poisson disk sampling [Yuksel 2015], see the zoom in view in Fig. 5 (c), in which the area represented by each sample is defined as the entire surface area divided by the total number of samples. Blocks that contain surface samples are always designated as active. If the samples are located close to the boundary of the block, which is identified by a distance threshold, the neighboring blocks should all be marked active to ensure that sufficient fluid regions are prescribed near the UAV boundary. For airflow farther from the boundary, active blocks must be deployed with greater adaptability. The criteria for selecting active blocks should ensure that turbulent wake flows are largely preserved, maintaining fidelity in resolving dynamic flow structures while optimizing computational resources; at the same time, the blocks should be selected such that the airflow near the domain boundary are relatively smooth to avoid introducing serious impact on the UAV dynamics. To achieve this, we delineate a buffer region of width d adjacent to each block boundary, see Fig. 5 (d). Within each block, we parallelize the computation of a turbulence metric at every grid node, categorizing nodes into "strong", "normal", or "weak" classes based on the turbulence activity using the pre-defined thresholds. In this work, vorticity serves as the primary metric; however, more sophisticated measures, such as the Q-criterion [Hunt et al. 1988] or turbulent kinetic energy [Pope 2000], could be potentially employed. First, we check the grid nodes inside the buffer region. If one of the nodes has a "strong" property, we label the current block as active and, at the same time, label the neighboring blocks as active too. If one of the nodes has a "normal" property, we keep the fluid properties of the current block as well as its neighboring blocks as they are, without proposing new active blocks. Then, we check the grid nodes outside the buffer region but inside the block. If any node exhibits a "strong" property, we label the current block as active. Conversely, if a node has a "normal" property, the block's activation state remains unchanged. Note that if the newly labeled active blocks are not yet created, the block should be allocated memory on the GPU and initialized. Finally, if all grid nodes inside the block have "weak" properties, this block

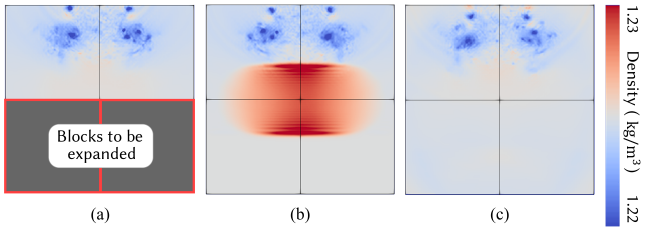

Fig. 6. **Comparison of different block initialization schemes.** (a) Block-based flow field prior to expansion. (b) Initialization of new blocks with constant values introduces spurious waves due to boundary discontinuities. (c) Solving Laplace equations for new block initialization mitigates spurious waves, yielding a physically plausible fluid field. Density colormaps are used for visualization to highlight spurious wave patterns.

should be removed from the memory. The buffer region width d influences the number of active blocks created. In this paper, we empirically choose d to be 20% of the block size.

3.2.4 Block creation and removal. Once a block is labeled as active and it has not been allocated memory on the GPU, it should be created first. The creation not only simply allocates memory for all grid nodes inside the block but also initializes the macroscopic quantities and distribution function values for each grid node. It is critical that macroscopic quantities initialized for the newly created active blocks should maintain continuity across the block boundaries; otherwise, erroneous compression waves will be formed that can damage the flow simulated inside the active blocks, see Fig. 6 (b) for a direct initialization by setting constant density and velocity. To ensure smooth field transitions across blocks, we first solve Laplace equations on the new block, and then reconstruct distribution functions based on the solutions as initialization. The Laplace equations we solve target both density and velocity fields as:

$$\nabla^2 \rho = 0, \qquad \nabla^2 \boldsymbol{u} = \boldsymbol{0}, \tag{9}$$

with boundary conditions classified into two distinct types. When the boundary face of a new block adheres to an existing active block, a Dirichlet boundary condition is applied, wherein the values of the boundary nodes of the new block are set exactly to those of the boundary nodes of the adjacent block. For other boundary faces corresponding to the domain boundary, the following Neumann boundary conditions are applied instead:

$$\boldsymbol{n} \cdot \nabla \rho = 0, \qquad \boldsymbol{n} \cdot \nabla \boldsymbol{u} = \boldsymbol{0}, \tag{10}$$

where $\boldsymbol{n}$ denotes the boundary normal. We discretize Eqs. 9 using finite differences on a uniform grid and employ a GPU-based Jacobian solver [Giles 2008] to numerically compute the macroscopic quantities. The distribution functions are then reconstructed directly from these quantities and their gradients by computing equilibrium and non-equilibrium parts, respectively [Malaspinas 2015]. Compared to block creation, the removal of a block is straightforward: one can simply delete the block from memory by labeling it inactive. During the streaming step of the subsequent time step, domain wall boundary conditions will automatically be enforced at the new boundary faces formed by the removal of the block.

3.2.5 Block data structure. As explained previously, our adaptive block-based LB solver necessitates dynamic block creation and removal. Allocating GPU memory on the fly for these operations leads to significant performance degradation. To mitigate this, we pre-allocate a large contiguous GPU memory array for a fixed number of blocks, with each block node storing essential information—fluid velocity, density, distribution functions, and associated properties. Note that a structure-of-array (SoA) layout [Obrecht et al. 2011] is employed for storing all nodes within a block. All pre-allocated blocks are linearized on the GPU and initialized as inactive. For efficient neighboring block access, an index table is maintained within each block, enabling constant-time lookups during fluid solver operations such as streaming. Initially, active blocks surrounding the UAV boundary are labeled sequentially starting from the first block. During simulation, blocks are dynamically marked active or inactive according to the selection criteria. For block creation, if a block lacks allocated memory, we search for the first inactive block in the pre-allocated array and assign it to the newly activated block. While a priority queue could optimize this search [Jaiswal 1968], we employ linear search due to the manageable number of blocks. For block removal, we simply mark the block as inactive in the pre-allocated array. Finally, since boundary samples are used to locate blocks around the UAV, a hash table [Alcantara et al. 2009] is constructed to enable efficient block indexing given a sample point.

3.3 Near-field boundary models and far-field coupling

For the near field around the UAV, physical consistency demands significant refinement of the boundary resolution, yet this proves impractical for flight controller design and evaluation due to the requirement for high efficiency. Thus, we rely on algebraic empirical modeling around the UAV, which couples with the far-field LB solver by providing the boundary conditions. For different types of UAVs, we classify the boundaries into a rotor-type boundary and other types of boundaries, and develop two kinds of parametric empirical models for them. In the following, we will explain in detail these two models as well as their coupling with the far-field solver.

3.3.1 Actuator line model for rotor blades. Rotor blades, serving as the propulsion system for various UAVs such as helicopters, multicopters, and some fixed-wing aircraft [Raymer 2012], rotate at high speeds during flight and feature thin structures. Direct aerodynamic simulation demands extremely high spatial and temporal resolution to accurately capture forces, rendering it infeasible for UAV control applications. To reproduce the physically consistent impact of surrounding aerodynamic flows on rotor blades, we adapt the Actuator Line Model (ALM) [Asmuth et al. 2019; Mittal et al. 2015; Troldborg 2009] into our simulation framework for efficient computation. To significantly reduce the computational load, we treat the blade as a finite series of segments along the spanwise direction (as shown in Fig. 7 (a)) and compute the force for each element using pre-computed data corresponding to the respective segment. We summarize the force for each segment according to the local fluid state as a function $\boldsymbol{F}_i(\hat{\boldsymbol{u}}(\boldsymbol{s}_i), \rho(\boldsymbol{s}_i))$, where $\boldsymbol{s}_i$ denotes the center of each segment; $\hat{\boldsymbol{u}}$ represents the flow velocity relative to the blade segment in the local coordinate system of the rotating blade, and ρ is the fluid density near the boundary of the blade. Once the force

is computed for each element, the total torque and force can be approximated by summing the contributions from all elements. Now, the question is how the force $\boldsymbol{F}_i$ on each element can be efficiently computed. From the observation that the cross-section of each blade element can be treated as a 2D airfoil, we decompose $\boldsymbol{F}_i$ into lift $\boldsymbol{F}_i^l$ and drag $\mathbf{F}_i^d$ forces, respectively ($\boldsymbol{F}_i^l = \boldsymbol{F}_i^l + \boldsymbol{F}_i^d$), which can be approximated by the following empirical model using lift and drag coefficients (refer to blade element theory [Glauert 1983]):

$$\boldsymbol{F}_i^l = \frac{k_{ll}}{2} C_l(\alpha_i)\rho(\boldsymbol{s}_i)u_i^2 c_i \Delta r_i \boldsymbol{n}_l, \;\; \boldsymbol{F}_i^d = \frac{k_{ld}}{2} C_d(\alpha_i)\rho(\boldsymbol{s}_i)u_i^2 c_i \Delta r_i \boldsymbol{n}_d, \tag{11}$$

where k_{ll} and k_{ld} are adjustable parameters, which are both set to 1 by default; c_i, Δr_i, α_i denote the chord length, span-wise size, and angle of attack (AoA) of the i-th element (note that c_i and α_i can vary according to the location of the element center); u_i is the relative incoming flow speed; C_l and C_d are the corresponding lift and drag coefficients (functions of AoA); and $\boldsymbol{n}_l$ and $\boldsymbol{n}_d$ are unit vectors along the lift and drag force directions. The element centers along the blade form an *actuator line* where the force is defined. The model presented in Eqs. 11 necessitates a reference flow speed u_i which should be specified. In many previous implementations, u_i is calculated based on the linear velocity of the blade rotation; if there are environment flows, an additional constant airflow speed is added [Asmuth et al. 2019; Troldborg 2009]. However, in our solver, since we have simulated far-field airflow, we can directly utilize the airflow from the far-field solver to form the reference speed u_i for each element. To obtain reliable u_i, a probe is defined in our solver for each element, which is located a distance of a cell size ahead of the element center along the rotating direction, see the blue cross pattern in Fig. 7 (a). The reference speed u_i is obtained by first interpolating the airflow velocity calculated by the far-field solver, and then taking the magnitude of that velocity. The angle of attack α_i for each element is defined as the angle between the interpolated velocity projected onto the cross-section plane of the element and the chord direction. It is also crucial to obtain reliable coefficient functions $C_l(\alpha_i)$ and $C_d(\alpha_i)$. We employ the well-known software XFoil [Drela 1989] to obtain C_l and C_d at discrete AoAs and interpolate function values for an arbitrary AoA. For airfoil shapes not supported by XFoil, we can directly use CFD software such as OpenFOAM [The OpenFOAM Foundation 2024] for computation. Typically, a steady-state solution suffices, offering efficient results.

3.3.2 Actuator surface model. Other than the rotating blade, other components of the UAV require an additional force calculation model that does not necessitate grid refinement near the boundary. Inspired by Yang and Sotiropoulos [2018] and Soliman et al. [2024], an Actuator Surface Model (ASM) is adopted, which is further modified to account for the influence of far-field flows. Notably, the UAV body has been uniformly sampled, and each sample $\boldsymbol{x}_i$ with a surface normal $\boldsymbol{n}_i$ represents a local element which is small enough to be approximated by a flat plane. Similarly, the force of each sample $\boldsymbol{F}_i$ is estimated by decomposing it into lift $\boldsymbol{F}_i^l$ and drag $\boldsymbol{F}_i^d$ components: $\boldsymbol{F}_i = \boldsymbol{F}_i^l + \boldsymbol{F}_i^d$. An empirical model analogous to that in ALM is employed to compute the lift and drag forces:

$$\boldsymbol{F}_i^l = \frac{k_{sl}}{2} C_l(\alpha_i)\rho_i u_i^2 S_i \boldsymbol{n}_l, \quad \boldsymbol{F}_i^d = \frac{k_{sd}}{2} C_d(\alpha_i)\rho_i u_i^2 S_i \boldsymbol{n}_d, \tag{12}$$

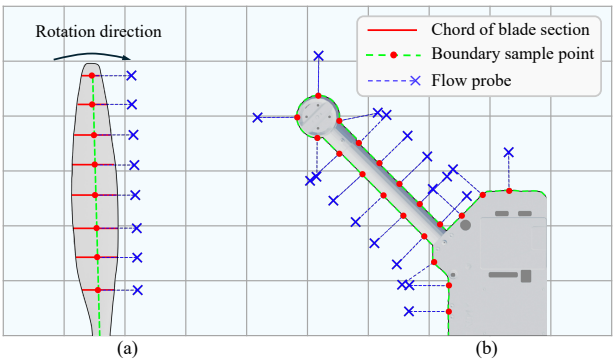

Fig. 7. **Illustration of near-field boundary models.** (a) Actuator line model applied to a propeller blade. (b) Actuator surface model applied to the UAV body. It should be noted that both models necessitate the use of probes to interpolate far-field flow velocities and densities to ensure reliability.

where k_{sl} and k_{sd} are adjustable parameters analogous to those in ALM, both set to 1 by default; u_i, α_i, and S_i denote the reference flow speed, angle of attack, and surface area for the sample $\boldsymbol{x}_i$, respectively; $\boldsymbol{n}_l$ and $\boldsymbol{n}_d$ are unit vectors along the lift and drag force directions. Since the UAV body surface is uniformly sampled, S_i can be approximated by dividing the entire surface area by the total number of samples. α_i and u_i can be calculated based on the interpolated far-field velocity in the vicinity of the sample $\mathbf{x}_i$. Assuming the local area around the sample $\mathbf{x}_i$ is nearly flat, these samples share identical lift C_l and drag C_d coefficient functions, which can be fitted into the following forms [Caplan and Gardner 2007]:

$$C_l(\alpha_i) = \sin(2\alpha_i), \qquad C_d(\alpha_i) = 2\sin^2(\alpha_i). \tag{13}$$

Similarly to ALM, a probe is utilized to determine u_i, as illustrated by the blue cross pattern in Fig. 7 (b). The probe is positioned at a distance of one cell size along the normal direction of the sample $\boldsymbol{x}_i$ to avoid interference from adjacent samples. At this probe location, the far-field velocity relative to the sample velocity is interpolated, and its magnitude is adopted as the reference flow speed u_i. The angle of attack α_i is computed based on the angle between the interpolated velocity from far-field block and the local plane of the sample, which is always positive. Note that ρ_i denotes the fluid density at the sample $\boldsymbol{x}_i$, also interpolated from the far-field block based on the location of the probe, which is more reliable. With the forces calculated at the samples, the total force and torque can be derived, serving as the input for the rigid-body solver.

3.3.3 Coupling with far-field LB solver. Until now, we have estimated the force acting on the solid surface samples using our near-field boundary models. However, these forces at the sampled locations should also influence the far-field fluid solver as a feedback for physical consistency. We adopt the concept from the immersed boundary method [Peskin 1972], in which force spreading is employed to effectively transfer the boundary force calculated from our algebraic empirical models to the surrounding fluid:

$$\boldsymbol{F}(\boldsymbol{x}_k) = \sum_{i \in N(\boldsymbol{x}_k)} \boldsymbol{F}_i \Delta(r_i; \boldsymbol{x}_k), \tag{14}$$

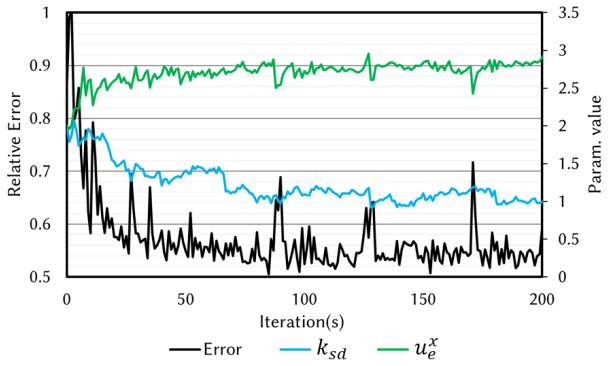

Fig. 8. **Automatic calibration for model parameters.** This figure depicts the variation of error metrics and parameter values as a function of iteration numbers during the optimization process used for automatic calibration. Convergence is attained after approximately 50 to 100 iterations. Note that since the environment wind velocity is along the x-direction only, we omit the y and z components of $\boldsymbol{u}_e$ in this plot.

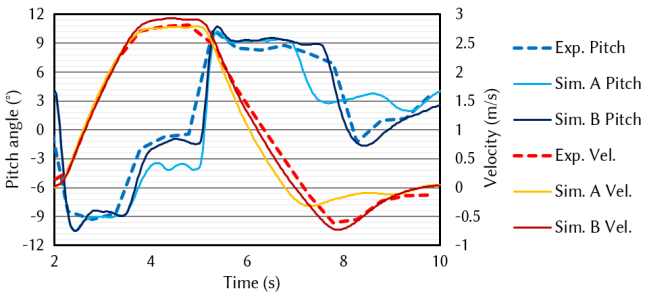

Fig. 9. **Sim. and exp. comparisons before and after calibration.** Comparisons between experimental results (dotted lines) and simulation outcomes (solid lines) are shown here. The trajectory corresponds to a quadcopter accelerating to 3 m/s and then decelerating until it stops. Simulation A employs initial setting, whereas Simulation B utilizes calibrated parameter values, showing improved closeness to the experiment.

where $\boldsymbol{x}_k$ is the far-field fluid node; i indexes the surface samples around the fluid node which has a total number of $N(\boldsymbol{x}_k)$; $\boldsymbol{F}$ denotes the force, and $\Delta(r_i; \boldsymbol{x}_k)$ is the spreading kernel function, in which $r_i = \|\boldsymbol{x}_i - \boldsymbol{x}_k\|$. Owing to the coarse grid resolution utilized in the far-field solver, the spreading kernel must remain sharp to avoid over-smoothing near the boundaries. Thus, we employ the following kernel function with the smallest size for spreading:

$$\Delta(r_i; \boldsymbol{x}_k) = \begin{cases} 1 - r_i, & \text{if } 0 \le r_i \le 1, \\ 0, & \text{otherwise.} \end{cases} \tag{15}$$

Note that ALM suggests that the far-field grid resolution should be comparable to the blade chord length. If the grid is too fine, applying body force at the center of each element may lead to instability during force spreading. Conversely, a grid that is too coarse cannot capture the overall trend of turbulent flows induced by the rotating blades. Experimental results show that a moderate grid resolution of 0.5 to 2 times the average blade chord length leads to consistent body forces and stable simulations within this range.

3.4 Calibration for simulation parameters

The FSI solver for UAV dynamics described above constitutes a hybrid simulation system that integrates a direct adaptive LB solver for the far field with localized algebraic empirical near-field models. These components are coupled and executed at a relatively coarse grid resolution to achieve extremely high performance on the GPU, enabling real-time or interactive operation. Although plausible animations can be easily produced, ensuring that the simulated UAV dynamics aligns with the corresponding real-world system remains challenging. To address this issue, a calibration procedure aimed at finding optimal model parameters, mostly for adjustable parameters in ALM and ASM, using a series of recorded data from real UAV flights (also known as system identification [Ljung 1998]) should be performed. We employ gradient-based optimization [Moré and Sorensen 1982] to derive optimal model parameters for this purpose.

We assemble the internal parameters k_{ll}, k_{ld}, k_{sl}, and k_{sd} of our simulator into a vector $\boldsymbol{\theta}_i$. Given that the UAV may fly in an environment with non-zero velocity, we augment the simulation parameters with a spatially constant environmental flow velocity $\boldsymbol{u}_e$ as the external parameter $\boldsymbol{\theta}_e = \boldsymbol{u}_e$. It should be noted that $\boldsymbol{u}_e$ could be temporally varying, and can be incorporated into the simulator by initializing the flow velocity with $\boldsymbol{u}_e$ and continuously specifying $\boldsymbol{u}_e$ to the domain boundary of the far-field LB solver. Now, the overall parameters for our simulator can be defined as $\boldsymbol{\theta} = [\boldsymbol{\theta}_i; \boldsymbol{\theta}_e]$, which are subject to optimization. By specifying the state variables $\boldsymbol{s}(t)$ that include the position and orientation of the UAV, the objective function to be minimized can be formulated as:

$$E = \sum_j \|\tilde{\boldsymbol{s}}(t_j; \boldsymbol{\theta}) - \boldsymbol{s}^*(t_j)\|^2 + \lambda \|\boldsymbol{\theta}_i - \boldsymbol{\theta}_i^0\|^2, \tag{16}$$

where $\tilde{\boldsymbol{s}}(t_j; \boldsymbol{\theta})$ denotes the states at discrete time samples from the simulator which can be influenced by all the parameters, and $\boldsymbol{s}^*(t_j)$ denotes the corresponding measured states from real-world flight. Note that the second term is a regularization term, which prevents the optimized internal parameters from deviating excessively from the default parameters $\boldsymbol{\theta}_i^0$, a vector of ones. λ serves as a regularization parameter for each specific calibration. The minimization is achieved via gradient descent starting from the default parameters. For $\boldsymbol{u}_e$, we initialize it to zero velocity as the initial guess. Owing to the highly-efficient simulation, the parameter gradients are calculated simply by performing numerical differentiation.

When calibrating the model parameters for a specific UAV, the above optimization process can be performed using various flight tasks. We propose a general guideline for selecting such tasks in two steps. First, we calibrate in a wind-tunnel setup across various possible flight states, without free movement. Key force contributors, including rotor forces and torques, body drag and lift, and control surface forces, can be effectively calibrated in this step. Such setups are easier to implement in real experiments or high-fidelity CFD simulations and yield a set of parameters reliable for normal flight scenarios. The second step involves selecting complete flight tasks to further calibrate overall performance, minimizing the sim-to-real

gap. Trajectory selection depends on the UAV type, but we recommend it covers movements along all three axes and a broad range for each control degree of freedom. An example calibration process for a complete task is shown in Fig. 8, which illustrates the convergence of errors and parameters. Meanwhile, Fig. 9 compares tracked states (e.g., pitch) over time—with and without the aforementioned calibration—for the flight example shown in Fig. 12, demonstrating improved consistency with the real-world system after calibration.

4 Design and Evaluation of Flight Controller

Flight controllers are crucial for stable flight, particularly for UAVs with atypical designs such as a hybrid UAV system, or in environments with strong disturbances from surrounding turbulent airflows—such as when a UAV approaches the ground or navigates through narrow channels [Matus-Vargas et al. 2021; Petrlík et al. 2020]. Traditional controller design methodologies typically depend on relatively simple dynamic models to establish the baseline structure of the controller, followed by manual parameter tuning or numerical optimization to finalize the control gains. However, such approaches face performance limitations in extreme operating conditions where intense fluid-structure interactions occur, as the empirical aerodynamic models underlying these controllers become substantially inaccurate under strong turbulent airflow conditions. While advanced FSI simulators could theoretically improve modeling fidelity, they mostly remain impractical for controller development due to prohibitive computational latency, creating a critical gap between more accurate theoretical modeling and real-world control requirements in complex aerodynamic environments.

In contrast, our hybrid simulator stands out with its superior efficiency, enabling novel exploration of flight controller design for a more general type of UAVs and in diverse environments with significant airflow interactions. Taking the PID controller [Kada and Ghazzawi 2011; Salih et al. 2010] as an example, which is often used for UAV flight control, manual tuning of control parameters based on the state feedback for both stable cruising and navigation through narrow channels can yield a controller applicable to broader scenarios. The real-time performance of our simulator facilitates efficient trial-and-error manual tuning. The calibration of our simulator with a real-world UAV even enables the tuned flight controller to be directly applicable to the real system—a process termed *sim-to-real transfer*. Notably, this process has been challenging for many prior simulators in more general airflow environments. On the other hand, control parameters can also be automatically optimized efficiently by following the similar procedure presented in Sec. 3.4. In addition to the PID controller, more advanced controllers such as the state-of-the-art adaptive controllers [Mo and Farid 2019; Yang and Xian 2019], or even controllers developed through reinforcement learning [Azar et al. 2021; Liu et al. 2018], can also be automatically derived using our simulator. These controllers are likewise anticipated to be applicable to real-world systems.

Flight controller evaluation for UAV systems—particularly those employing open-source platforms like PX4 [Meier et al. 2015]—conventionally begins with simulation-based verification to mitigate deployment risks and costs. Traditional approaches utilize simulators built upon empirical models to assess basic performance metrics through standardized tests such as setpoint tracking, path following, and attitude stabilization, with controller performance quantified by tracking errors, maximum overshoot, response time, and related dynamic indicators. However, this methodology exhibits critical limitations of inability to simulate obstacle-induced aerodynamic disturbances, and inapplicability to UAVs with unconventional configurations where complex fluid-structure interactions dominate, as well as restricted validation of controller robustness under realistic environmental challenges. Our simulation framework attempts to overcome these constraints by enabling more rigorous evaluation of flight controllers under general airflow conditions formed by environmental obstacles or unconventional UAV configurations, specifically analyzing tracking precision in attitude and positional control. This capability significantly reduces operational risks during real-world deployment while maintaining compatibility with standard performance quantification methodologies.

5 Results and Discussions

Due to its high efficiency that imposes no significant computing resource demands, our hybrid simulation system can run on a normal computer. In all experiments presented below, simulations were executed on a computer featuring an AMD Ryzen 7 5700G CPU, 64 GB of RAM, and an NVIDIA GeForce RTX 2080 Ti GPU with 11 GB memory. The system is also compatible with laptop computers featuring even lower hardware configurations. Depending on the resolution and the scale of simulation, our system achieves real-time performance or at least interactive responsiveness. Note that the overall efficiency can be significantly higher when using more advanced GPUs, such as the NVIDIA GeForce RTX 4090, making real-time simulations more achievable. By default, flow visualization is achieved via direct volume rendering of the velocity field using colormapping of velocity magnitudes [Brebin et al. 1998; Kruger and Westermann 2003]. To enhance visual quality, particles are injected into the flow field and advected along with the velocity field. Each particle is colored by colormapping the velocity magnitude at its location. Particles are rendered in Blender [Community 2018] by exporting them as VDB [Museth 2013] files. Note that while simulation and direct volume rendering occur in real-time, realistic particle rendering (as shown in subsequent results) is processed offline.

In the following sections, we first validate our simulator by comparing it to real-world experiments across different setups, both with and without flight control. We then demonstrate a series of applications of our simulator for deriving flight controllers for various UAV types (fixed-wing, multi-rotor, and hybrid configurations) under diverse flight conditions: normal cruising, aerodynamic stall, ground/obstacle proximity effects, and multi-UAV navigation in complex environments. We highlight the advantages of our simulator over traditional empirical models by referencing real-world flight data. Finally, we showcase how our system facilitates flight control optimization, enabling unique sim-to-real transfer without additional model parameter fine-tuning, followed by further discussions and presentations of the limitations of the present simulator. Table 2 lists the details of the configurations and timings for all the simulation and flight control examples shown in this paper.

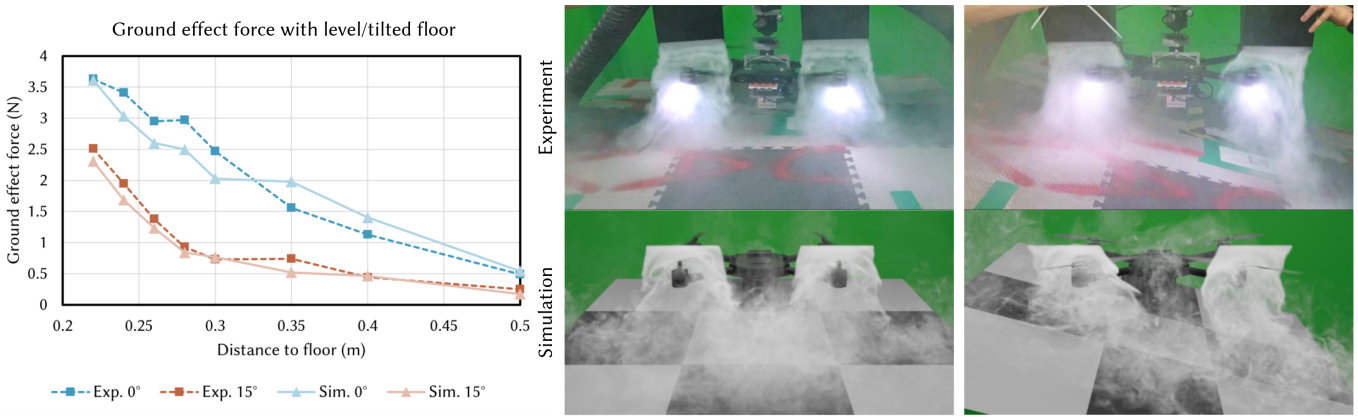

Fig. 10. **Quadrotor ground effect validation with level and tilted floors.** Left: thrust forces vs. ground proximity (0.2–0.5 m) for level (0°, blue dashed/light blue solid lines) and 15° tilted (red dashed/orange solid lines) floors. Right: Flow visualization comparison: Experimental water fog trajectories (white patterns, top) vs. hybrid simulations with particle injections (bottom), resolving airflow interactions with level (left column) and tilted (right column) floors.

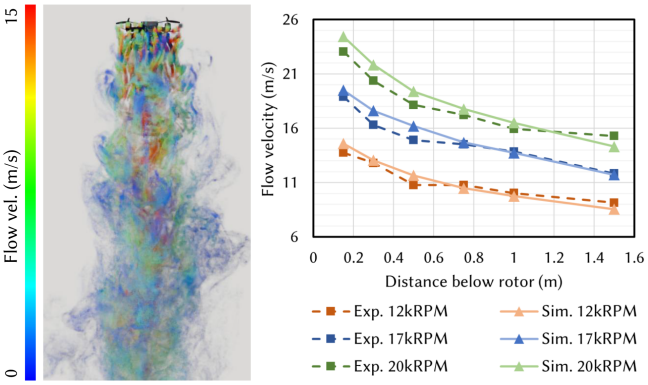

Fig. 11. **Downwash velocity validation under varying rotor speeds.** Left: Our hybrid simulation of turbulent flow velocity distribution below rotor using particle tracing and volume visualization, colored by velocity magnitude. Right: Experimental vs. simulated velocity profiles at multiple vertical distances, demonstrating parameter-calibrated consistency (dashed lines: experiments; solid lines: simulations) across all tested rotor speeds.

5.1 Validations

It is crucial for a flight control simulator to yield physically consistent results. Here, we verify our simulator under various conditions and comparing them with real-world experimental data, thereby assessing the potential capabilities of our simulator applied for real-world flight controller design and evaluation.

5.1.1 Quadcopter thrust under ground effects. The thrust generated by a quadcopter can be calibrated in our simulator by adjusting the parameters of ALM to match experimental values. Notably, when a flat plane is positioned below and in close proximity to the quadcopter, the thrust increases at the same rotor speed—a phenomenon often referred to as the ground effect force [Sanchez-Cuevas et al. 2017]. The ground effect force encountered by a quadrotor is tested in both simulations and experiments across different distances to the flat floor and tilt angles of the floor, as illustrated in Fig. 10. The results demonstrate strong consistency with the physical experiments, validating the ability of our simulator to accurately resolve additional aerodynamic forces induced by nearby objects on a UAV.

5.1.2 Downwash flow speed. In addition to the forces acting on the UAV, we also compared the impact of a UAV on the flow field. The downwash flow from rotors can exhibit relatively high speeds and cover a large area, significantly affecting nearby aerial vehicles. We evaluated a standard case with a stationary quadcopter (Fig. 11 (left)) and measured the flow velocity beneath it at varying distances (Fig. 11 (right)). This comparison was performed under the precondition that the total thrust matched experimental data at identical rotor speeds through model calibration. While noticeable errors still exist at locations close to the rotor, the overall accuracy is sufficient to replicate the downwash effects on other vehicles.

5.1.3 Quadcopter acceleration and braking. In addition to comparisons with static physical experiments, we also performed evaluations against real-world flight tasks. For a quadcopter, an autonomous flight controller is essential to stabilize it; otherwise, it cannot even hover briefly due to small external disturbances. In our comparison, the PX4 flight controller [Meier et al. 2015] was deployed on both the real and simulated vehicles, which shared the same set of control parameters. The task was designed as accelerating along the heading direction to a certain speed, maintaining flight for a period, and then braking to stop at the current position. We also matched the simulated environmental wind speed to the averaged measurement value from outdoor experiments, as shown in the comparison in Fig. 12. The simulated trajectory exhibits high consistency with the experimental data, demonstrating that the simulator is sufficiently accurate for basic practical tasks.

5.1.4 Quadcopter following infinity-shaped trajectory. A more complex infinity-shaped trajectory following task is also compared between real flights and simulations, which were conducted in two different setups as shown in Fig. 13. In Setup A (top row), a smaller quadcopter (wheelbase length = 250mm) with a simple position

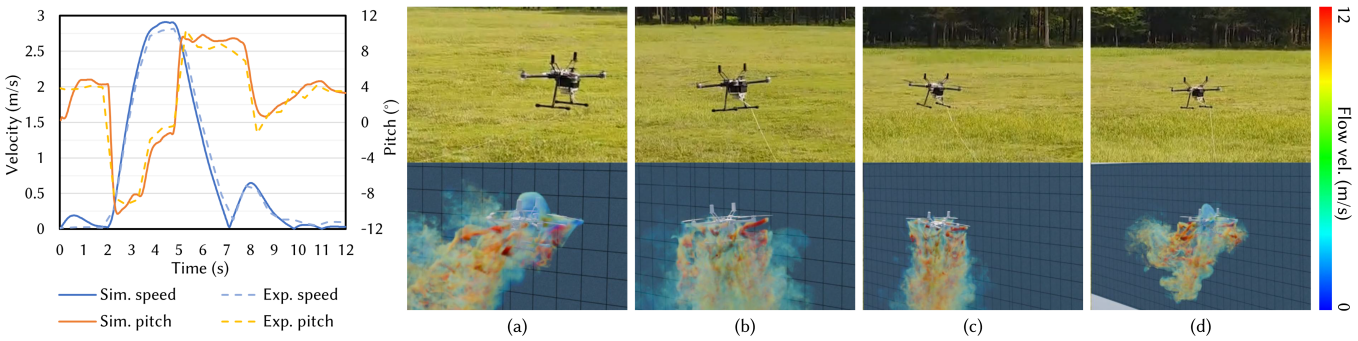

Fig. 12. **Validation of acceleration-braking maneuver for a P600 quadcopter.** Left: Temporal evolution of horizontal speed (blue) and pitch angle (orange) in both simulation (solid lines) and real flight experiment (dashed lines), demonstrating model-calibrated consistency. Right: State sequence under 3 m/s forward airflow: (a) hovering, (b) accelerating, (c) braking, and (d) stopped, with particle-visualized flow fields.

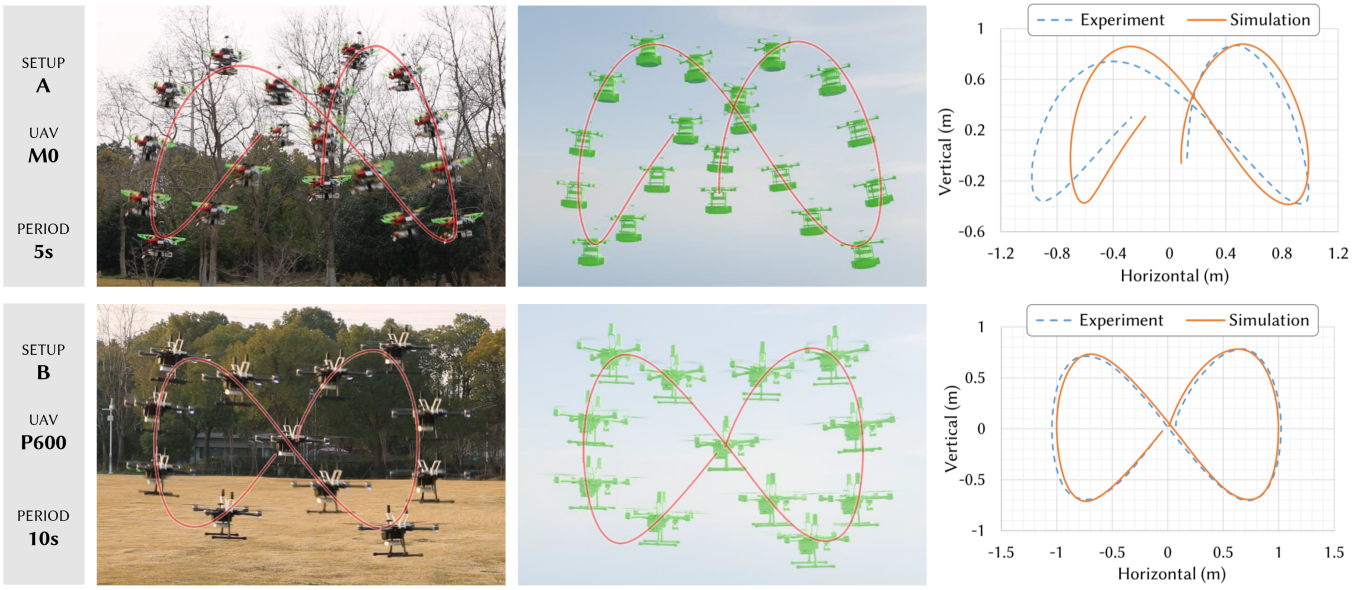

Fig. 13. **Validation of infinity-shaped trajectory following.** Left & center: Snapshot comparisons of UAV in real-world experiments and our simulations for two configurations: Setup A (UAV M0, wheelbase 250mm, 5s/loop) and Setup B (UAV P600, wheelbase 600mm, 10s/loop), demonstrating velocity-dependent trajectory following under higher-speed (A) vs. lower-speed (B) conditions. Right: Horizontal/vertical trajectory deviations (blue dashed: experiment; orange solid: simulation) validate model-calibrated physical consistency in Setup A and Setup B environments.

controller is used. At the target speed of 5 seconds per loop, its trajectory deviates significantly from the target shape due to the limited agility of the quadcopter. The simulated trajectory exhibits a noticeable shift relative to the real flight test, primarily attributed to unstable outdoor wind conditions and localization errors in the real UAV (sensor noise is unapplied in this simulation case). In addition, a pure inertia-based positioning scheme is employed in this setup, further exacerbating the accumulated error for the real flight. Nevertheless, even under this condition with strong uncertainty, the overall trend of the trajectory following can be achieved. Conversely, Setup B (bottom row) employs a larger quadcopter (wheelbase length = 600mm) with enhanced wind resistance, operating at a slower speed of 10 seconds per loop. Moreover, a centimeter-level accurate RTK positioning scheme with almost no accumulative error is adopted. The trajectory closely aligns with the infinity-shaped target, and the discrepancy between real flight and simulated results becomes minimal. The two different setups indicate that environmental disruptions and sensoring errors can greatly contribute to a poor matching between simulation and experiment. To produce more robust controller through the simulator, modeling such uncertainty with randomization is encouraged.

Previous validations, through real-world experiments paired with calibrated simulations, have verified the acceptable physical consistency of our simulator. These findings demonstrate the applicability of our simulator to diverse flight control tasks.

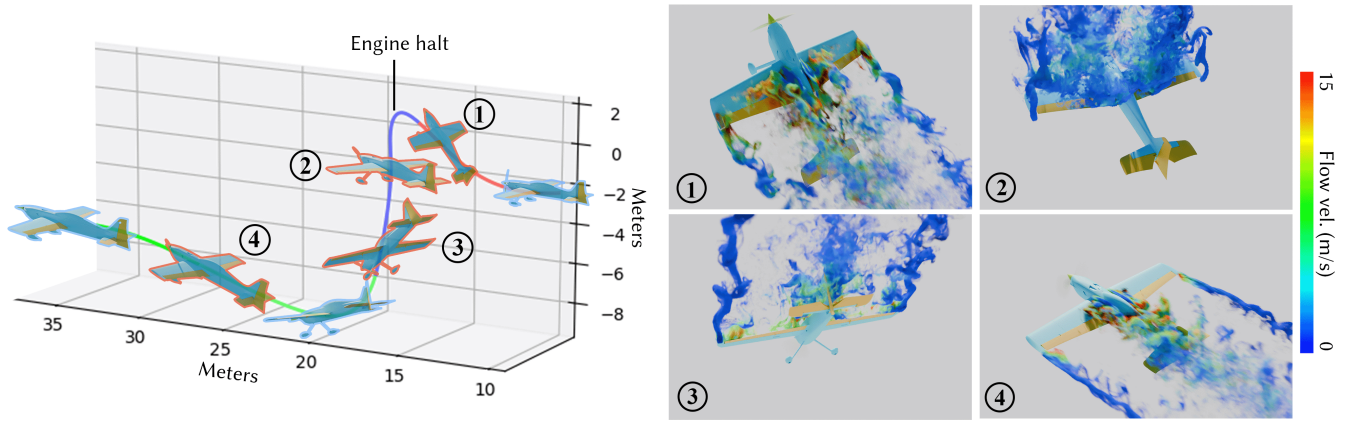

Fig. 14. **Aerodynamic stall and recovery of a fixed-wing UAV.** The left panel displays the 3D flight trajectory from sudden climb (red) to engine halt with stall-induced descent (blue), and to controlled recovery (green), highlighting four critical flight stages. The right panels show sequential attitude snapshots illustrating (1) pitch-up dynamics near stall, (2) free-fall, (3) stabilization with reactivated engine, and return to (4) level flight.

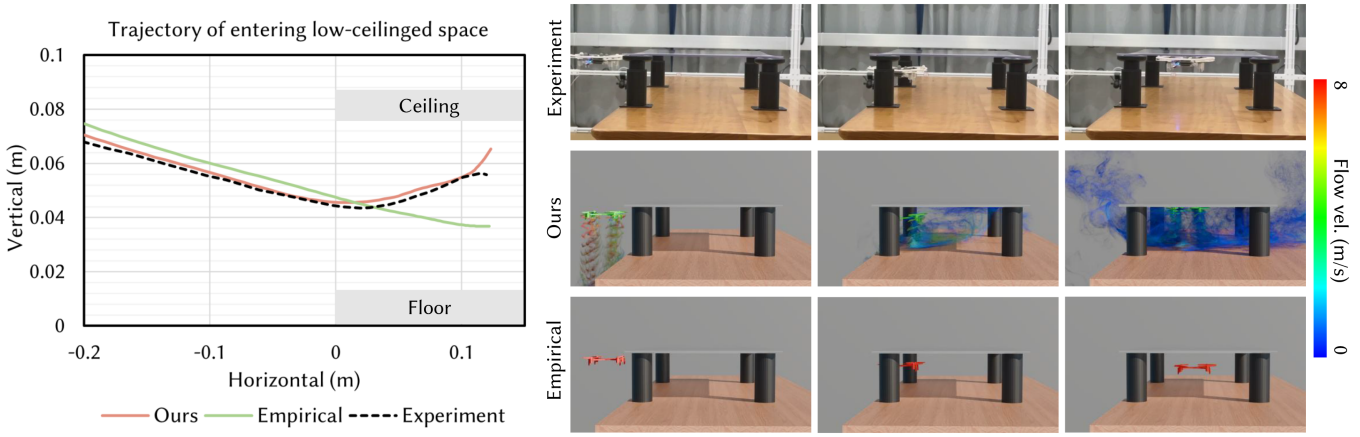

Fig. 15. **Quadrotor traversing in low-ceilinged confined space**. Left: Horizontal vs. vertical trajectory comparisons among experiment (dashed black), our simulator (solid red) and empirical models (solid green). Our simulator replicates ground and ceiling confinement effects via floor-ceiling aerodynamic interactions, while the empirical model fails to capture such effects, yielding open-air behavior. Right: Snapshots comparisons of experiment, our simulator and empirical model validate strong airflow-UAV interactions during key flight phases: (I) entry, (II) close-proximity effect accumulation, and (III) ground effects and ceiling adhesion, indicating the importance of our simulator for flight controller design in these extreme scenarios.

5.2 Applications to flight control

In the following sections, we demonstrate the diverse potential applications of our simulation system to UAV control problems across different UAV configurations. These scenarios involve complex airflow interactions with the UAV, where our simulator offers essential advantages over traditional empirical models.

5.2.1 Fixed-wing UAV flight with stall and recovery. In this application example, the PX4 autopilot system—incorporating a cascaded-PID controller for a fixed-wing UAV—was implemented within our simulation system to regulate aircraft speed and attitude. Our simulator with default parameters was used to replicate the complete aerodynamic stall and recovery process of a fixed-wing UAV, as depicted in Fig. 14. The experimental sequence begins with engine (rotor) deactivation after a pull-up maneuver, inducing a full aerodynamic stall condition. Following a brief free-fall phase marked by unsteady dynamics, engine reactivation enables recovery to nominal flight attitude with the aid of flight control. This methodology assesses the operational robustness of the flight controller under extreme scenarios with strong turbulent flow interactions, as the highly nonlinear stall dynamics evade conventional empirical modeling and render traditional prediction methods insufficient for attitude variation analysis. The results highlight the necessity of FSI solutions to resolve complex fluid-structure interactions during these critical flight regimes in order to obtain more versatile flight control.

5.2.2 Quadcopter traversing through a low-ceilinged space. Flying in confined spaces presents unique challenges for any types of UAV. Fig. 15 systematically compares experimental recordings, our simulation results, and empirical model predictions for a Crazyflie small

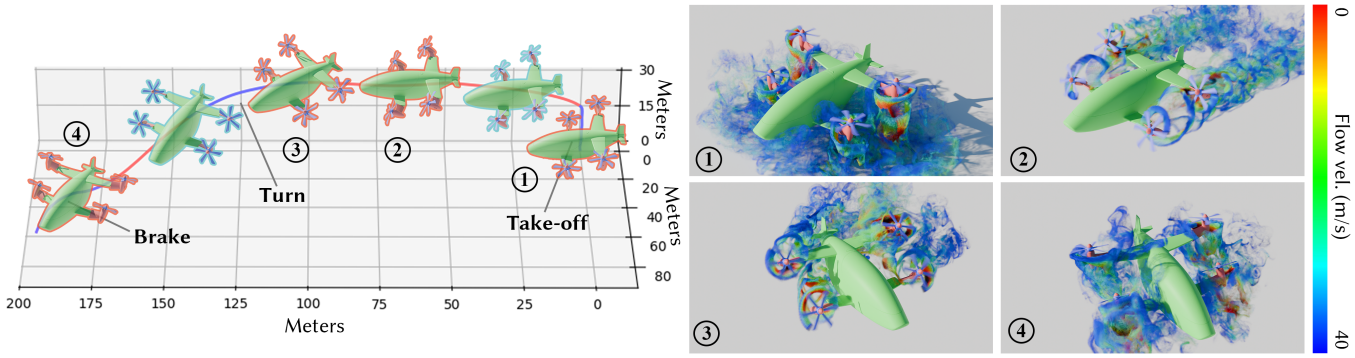

Fig. 16. **Tilt-rotor hybrid UAV with multi-regime flight.** Left: Overall view illustrating takeoff, turning, and braking trajectories under a simplified cascaded-PID controller, with rotor tilt angles manually configured per state. Right: Detailed aerodynamic snapshots: (1) vertical ascent, (2) high-speed cruise, (3) turning-phase asymmetric tilt-induced airflow, and (4) braking-phase pitch instability. UAV body and rotor-blade flow fields are simulated via our simulator, capturing nonlinear rotor-rotor/rotor-body aerodynamic couplings that challenge traditional PID controller design relying on empirical models.

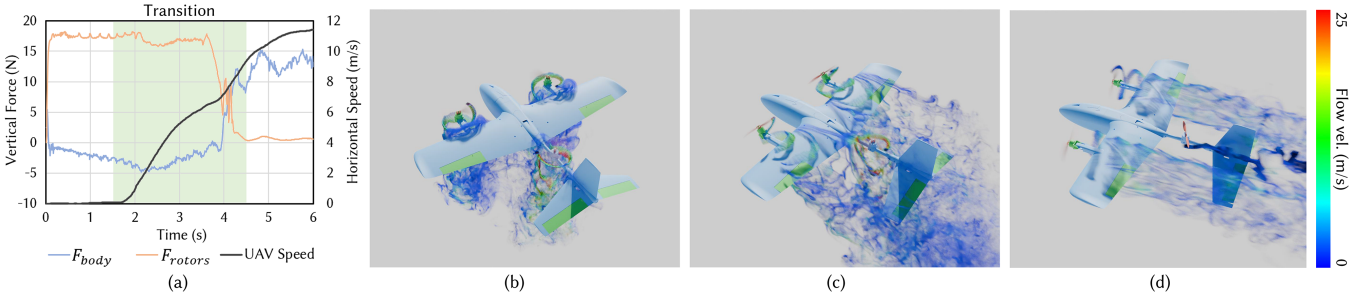

Fig. 17. **Tilt-rotor hybrid UAV with large wingspan performing transition from hovering to cruising.** The hybrid UAV, equipped with two front tilt rotors, one fixed rear rotor, and built on a fixed-wing platform, exhibits a smooth transition from hovering to cruising, as illustrated in (b), (c), and (d). Nonlinear coupling effects between the rotors and main wing are captured, as depicted in particle visualizations of the velocity field. During the transition, the lift force of the main wing gradually takes effect, reducing the vertical force required from the rotors (see (a)), and improving flight endurance.

quadrotor [Giernacki et al. 2017]—equipped with a cascaded-PID controller obtained from PX4—as it enters a narrow space with a low ceiling. This comparative analysis highlights the critical advantage of our simulation framework in capturing confined-space aerodynamic interactions that fundamentally alter quadcopter dynamics. While conventional empirical models fail to account for ground/ceiling-induced flow recirculation effects, our approach successfully replicates the experimentally observed ceiling adhesion phenomenon with ground effects, evidenced by the captured images and measured altitude data shown in Fig. 15. The divergence between empirical predictions and experimental/simulation outcomes underscores the limitations of the aerodynamic assumptions employed in conventional empirical models in constrained environments. Our simulation system thus emerges as an essential tool for resolving complex fluid-structure interactions that elude traditional empirical models, offering critical insights for developing robust navigation strategies in confined spaces.

5.2.3 Tilt-rotor hybrid UAV flight. For tilt-rotor hybrid UAVs, the rotors can dynamically adjust their angles to shift the flight state from vertical hovering to forward cruising. Designing flight controllers capable of adapting well to this scenario poses challenges. Herein, we demonstrate tilt-rotor hybrid UAVs operating across multiple flight regimes—encompassing takeoff, turning, and braking maneuvers—with controllable rotor tilt and rotating speed (see Fig. 16). Our simulation system implements a cascaded-PID controller derived from PX4, initially designed for quadroters, to stabilize the attitude of the vehicle, with rotor tilt angles manually set for each operational state. This represents a far more complex UAV control challenge due to the intricate aerodynamic coupling between rotor downwash and fixed-wing surfaces. Although an ideal flight controller would dynamically incorporate tilt-angle modulation into attitude control loops for enhanced stability, the current implementation uses a simplified control architecture to showcase the capability of our simulator in handling such multifaceted scenarios. While the hybrid UAV shown in Fig. 16 has a smaller wingspan, Fig. 17 presents another hybrid UAV with a sufficiently large wingspan, where aerodynamic lift exerts a significant effect during cruising, as evidenced by the measured vertical forces in Fig. 17 (a). Note that stabilizing the hybrid UAV remains an open challenge due to the inherent difficulties in empirical dynamic modeling—arising from both the increased degrees of freedom of the actuator system and the

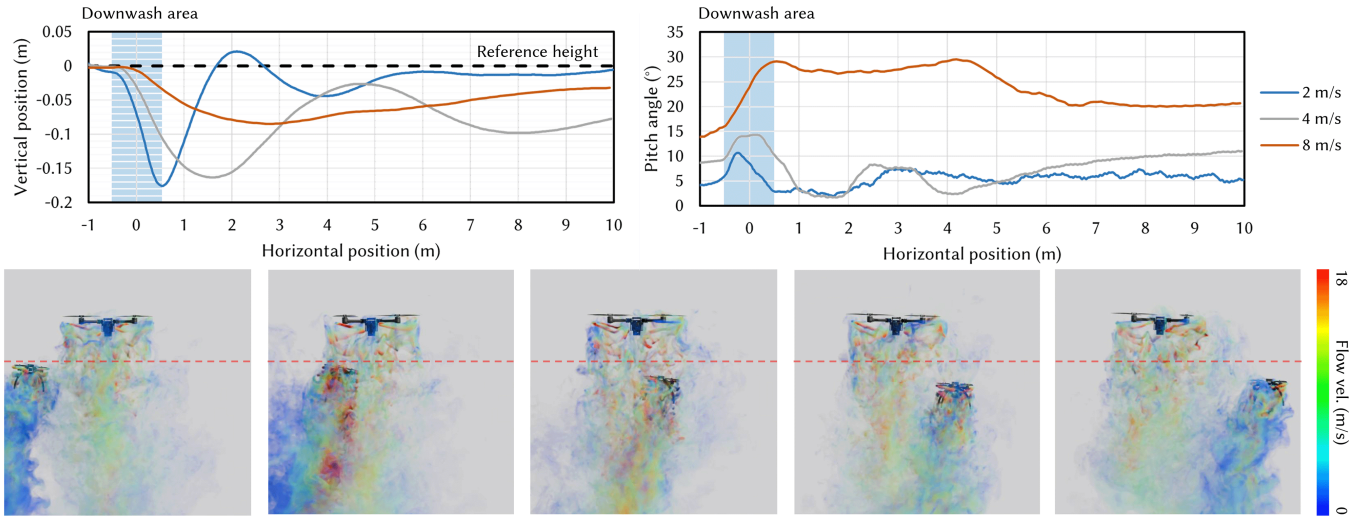

Fig. 18. **Downwash interference of two quadcopters.** Top: Vertical position (left) and pitch angle (right) versus horizontal displacement under three speeds (2/4/8 m/s, blue/gray/orange lines), capturing aerodynamic interference during downwash penetration. Bottom: Particle-visualized velocity fields for the 2 m/s flight scenario using our simulator. This example demonstrates the advantage of our simulator for multi-UAV flight simulation, highlighting the challenge of designing flight controllers to effectively compensate for downwash effects.

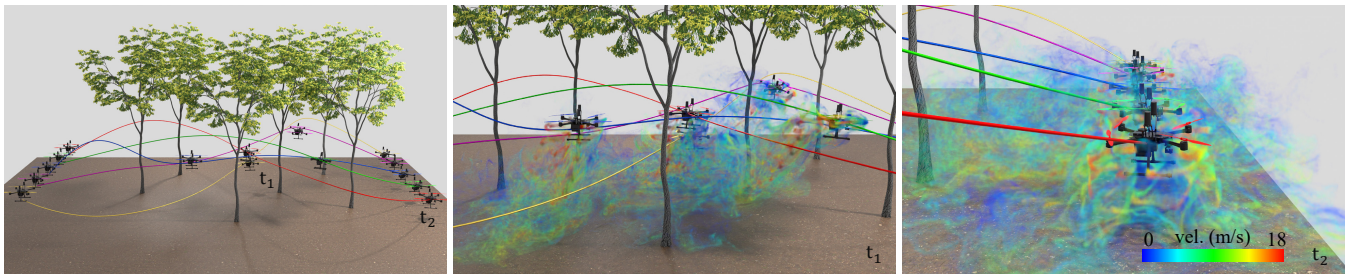

Fig. 19. **Multiple quadcopters navigating through forest.** This scenario illustrates left-to-right coordinated flight of multiple UAVs along distinct color-coded predefined trajectories. Turbulent flow fields at times t_1 and t_2 are visualized, demonstrating the efficiency of the simulator in modeling multi-UAV aerodynamic interactions within complex obstructed environments. The visualization captures dynamic rotor wake interactions with terrain and obstacles, highlighting the ability of our simulator to resolve fluid-structure coupling in real-world scenarios.

strong rotor-rotor/rotor-body couplings observed during transient maneuvers like turning and braking. These nonlinear interactions effectively render many model-based control design methods that are effective in conventional UAV configurations inapplicable. As a result, our simulator offers new opportunities for addressing such complex flight control problems through more advanced approaches, including end-to-end learning and data-driven control strategies.

5.2.4 Downwash interference of two quadcopters. In scenarios involving multiple UAVs, their formations may transition dynamically between clustered configurations and scattered patterns. It is challenging to simulate such scenarios using local-domain solvers like "FishGym" [Liu et al. 2022] and its variant [Song et al. 2025] with traditional box-like setup. Additionally, multi-UAV systems introduce novel control challenges due to the need for coordinated flight in close proximity. Our simulator presents unique advantages due to adaptive block-based design. Here, we demonstrate the capability of simulating two flying UAVs using our system in a downwash interference case. A smaller quadrotor UAV (wheelbase length = 250mm) flies horizontally at a constant speed and altitude using the controller derived from PX4, and passes through a downwash area generated by another larger hovering quadrotor UAV (wheelbase length = 600mm). The altitude and pitch angle of the small UAV fluctuates after entering the downwash area, as shown in Fig. 18. Under different speeds, the fluctuation behavior varies. For example, under 2 m/s and 4 m/s, the altitude change behaves as oscillating and underdamped; while under 8 m/s, an overdamped behavior is observed. This variation is caused by the controller tracking a target height, with different speeds and attitudes. The downwash effects are hard to model empirically, since the flow structure below the UAV is non-uniform and could be disturbed by interaction with other UAVs; the force produced by the propeller within a downwash area is also hard to predict efficiently without our hybrid simulator.

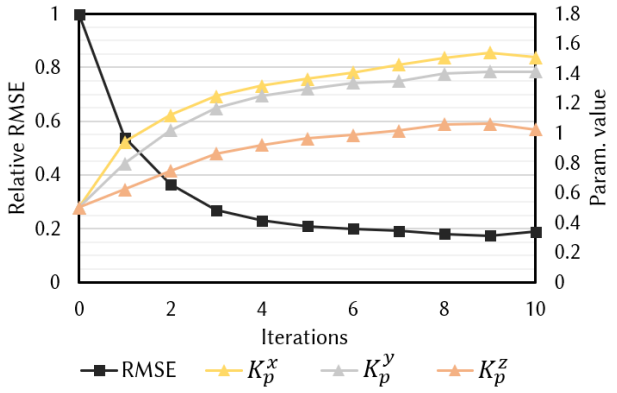

Fig. 20. **Controller optimization.** We show the convergence of flight controller optimization for the root-mean-square error (RMSE, black) and the proportional gain (yellow/gray/orange curves) across gradient-based iterations, validating monotonic error reduction with parameter regularization.

5.2.5 Multiple quadcopters navigating through forest. To further demonstrate the capability of our hybrid simulation system for multiple UAVs in complex environments, we showcase a five-quadcopter swarm navigating through forested terrain (Fig. 19). Here, predefined trajectories—generated via cubic spline interpolation of waypoints with velocity profiles—are tracked using the cascaded-PID controller derived from PX4. This scenario deliberately includes both expansive and proximity flight phases, necessitating seamless transitions between adaptive grid blocks. Such capability is unattainable in many other FSI solvers due to domain decomposition limitations, while simultaneously testing the robustness of the flight controller against aerodynamic coupling effects during close formation flight. The experiment not only verifies trajectory-tracking performance under more realistic aerodynamic interactions but can also offer critical insights for developing coordinated multi-UAV control strategies in obstacle-rich environments.

5.3 Controller optimization and sim-to-real transfer

Notably, our simulation system enables optimization of not only model parameters but also control parameters. This dual optimization ensures reliable sim-to-real transfer, as demonstrated in subsequent analyses. Here, we demonstrate optimization for two tasks using a simple PD controller, along with zero-shot transfer to real-world deployment. Specifically, we focus on quadrotor UAV (wheelbase: 250 mm) flight control tasks: complex trajectory tracking (Fig. 21) and checkpoint racing (Fig. 22). The UAV's flight control uses a PD controller with 30-Hz updating frequency for position regulation, with attitude stabilization handled by the PX4 flight system.

Trajectory tracking task. The target trajectory resides on a hyperbolic paraboloid surface, forming a saddle shape (see the bottom-left image in Fig. 21). Initial control parameters are manually assigned as a baseline. Using these nominal gains, our simulation executes the full trajectory and records tracking performance. To enhance accuracy, we employ a gradient-based optimization within the simulation environment. The cost function is defined as the root-mean-square error (RMSE) between the simulated UAV trajectory and the reference path. At each iteration, the system evaluates performance under current control gains, computes the RMSE, and updates the gains accordingly. The evolution of RMSE and parameter values during optimization is depicted in Fig. 20. After convergence, the optimized parameters are transferred directly to the real-world UAV system without further fine-tuning. The same reference trajectory is used in the real-world experiment. During flight, the UAV estimates its motion using onboard optical flow sensors, which provide real-time position feedback for quantitative evaluation. Fig. 21 compares the simulated and real-world tracking results with both the initial and optimized control parameters. In the baseline case, both simulation and physical experiments exhibit notable deviations from the reference trajectory. After optimization, the UAV demonstrates significantly improved tracking performance in both domains.

Checkpoint racing task. Unlike trajectory tracking which minimizes tracking error with respect to the desired trajectory, the racing scenario minimizes the time required to reach consecutive checkpoints instead. In our demonstration, we use a sequence of checkpoints arranged in a vertical five-pointed star; this layout challenges the quadrotor with sharp turns and rapid height changes. The flight controller uses the next checkpoint as its position setpoint until the quadrotor reaches a specified proximity (0.1 m in this case). The optimization process is similar to the trajectory tracking task, except the objective is minimizing task completion time rather than the trajectory tracking error. Compared to the initial control, the optimized control significantly reduces task completion time, allowing the quadrotor to smoothly and rapidly pass through all checkpoints (see their trajectories in Fig. 22). Detailed flight times and optimized control parameters are listed in Table 1.

The above results validate the ability of our simulation system to support closed-loop controller design and optimization, and demonstrate strong sim-to-real transfer consistency. The framework enables efficient iteration in simulation and reliable deployment on hardware, providing a practical attempt as a pipeline for UAV flight controller design, optimization and transfer.

5.4 Discussions

The design and evaluation of flight controllers demand highly efficient simulations, explaining why empirical models are predominantly used over full FSI solvers in the past. However, empirical models struggle to handle more general scenarios, especially those involving strong surrounding airflow interactions. Our hybrid simulator aims to bridge this gap, offering the best of both worlds. By calibrating with the target physical system, our simulator effectively yields flight controllers that can be directly transferred to real-world UAVs. This capability paves the way for exploring the design of more sophisticated controllers entirely within a simulation environment. The efficient simulation of multi-UAV systems further expands the application scope to the exploration of more universal swarm flight control algorithms. On the other hand, the capability of our simulator also extends the evaluation of flight controller beyond basic trajectory tracking to assess resilience against unpredictable aerodynamic disturbances. Insights from these evaluations can guide the development of more adaptive control strategies—such as airflow

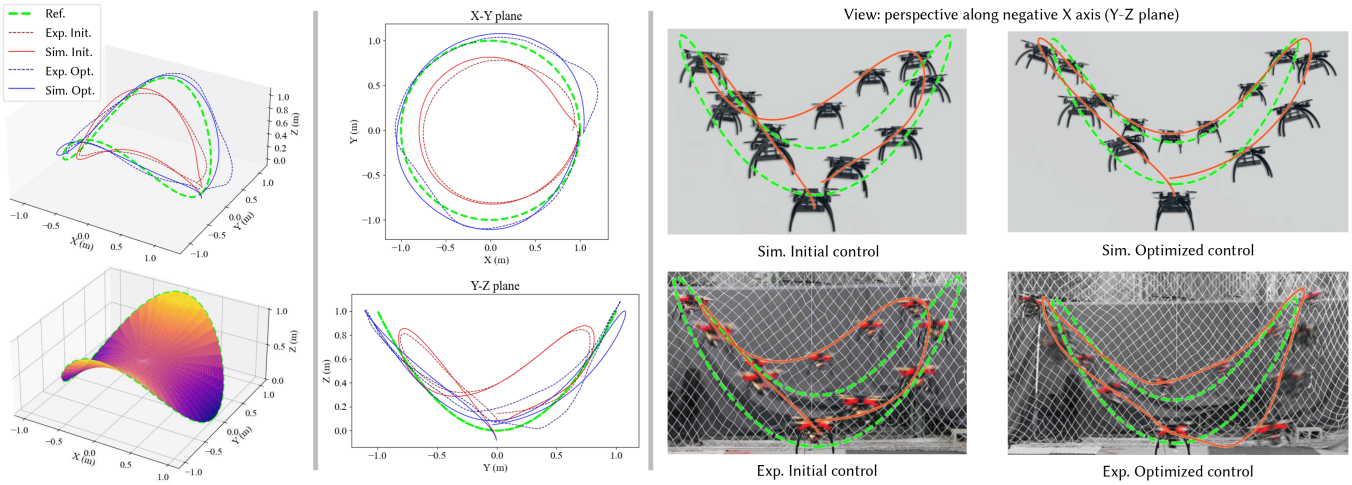

Fig. 21. **Saddle-shaped trajectory tracking with controller optimization and sim-to-real transfer.** Left: 3D view of a saddle-shaped trajectory tracking results, comparing simulation (solid blue: optimized; solid red: initial values) with experimental results (dashed blue: optimized; dashed red: initial values). (Middle) 2D planar projections of the saddle-shaped trajectory tracking results shown in left figure. Right: Stacked photography overlay: (top) recorded simulation trajectories (orange) vs. target (green dotted), and (bottom) recorded experimental trajectories (orange) vs. target (green dotted), demonstrating the effectiveness of controller optimization through gradient-based parameter calibration as well as the sim-to-real transfer without additional parameter tuning.

Table 1. **Detailed data of checkpoint racing task with controller optimization and sim-to-real transfer.** The specific time cost of flying through consecutive checkpoints is compared between simulation and sim-to-real transfer, both before and after flight control parameter optimization, with the corresponding control parameters also listed. Note that all checkpoints are on $y-z$ plane; thus, control parameters on x-dimension are not optimized.

Trial	$P_0 \rightarrow P_1$	$P_1 \rightarrow P_2$	$P_2 \rightarrow P_3$	$P_3 \rightarrow P_4$	$P_4 \rightarrow P_5$	$P_5 \rightarrow P_1$	t_{total}	K_p^y	K_p^z	K_d^y	K_d^z
Sim. Init.	1.40s	2.32s	2.04s	2.08s	2.20s	2.16s	**12.20s**	0.5	0.5	0.5	0.5
Exp. Init.	2.00s	1.88s	2.12s	2.68s	1.76s	2.60s	**13.04s**				
Sim. Opt.	0.76s	1.16s	1.12s	1.32s	1.16s	1.40s	**6.92s**	0.67	0.68	0.31	0.20
Exp. Opt.	1.00s	1.04s	0.96s	1.40s	1.04s	1.64s	**7.14s**				

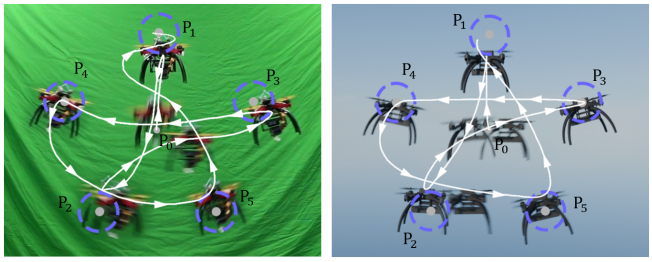

Fig. 22. **Checkpoint racing task with sim-to-real transfer.** Using optimized flight control parameters to minimize flying time, the trajectory from the experiment (left) and our simulation (right) is compared. The UAV starts at P_0, passes through checkpoints P_1, P_2, ..., P_5, and finally returns to P_1. The violet circles around each checkpoint indicate the distance threshold for determining whether the UAV has reached a checkpoint.

state estimation or bandwidth adjustments for managing unsteady aerodynamic loads—while supporting research into cooperative control algorithms where inter-agent aerodynamic coupling cannot be ignored. By integrating optimization and visualization/rendering techniques, the entire system has the potential to offer an efficient and intuitive platform for the design of motion control systems across various UAV types. Our simulation system is also scalable beyond UAVs and robotics. One promising application domain is the generation of highly realistic animations, such as avian flight, wherein the control policies governing bird wing flapping can be systematically explored across diverse complex scenarios.

5.5 Limitations

As an early attempt at direct FSI for UAV simulation, our system still has limitations in several aspects. First, our hybrid simulator exhibits model restrictions and uncertainties. Due to relatively coarse grid resolution, particularly near the UAV boundary, strong turbulent flow separation may not be accurately captured. Thus, a single default set of model parameters cannot be universally applied to different scenarios, and calibration with reference data is always required to ensure physical consistency. Second, our simulator lacks support for multi-resolution domains, leaving computing resource allocation not fully optimized. Additionally, the selection criteria for active blocks could be enhanced. Finally, the overall performance of the simulation system is not fully optimized, and future work could explore more advanced GPU-based data structures.

Table 2. **Simulation setup and performance on various cases.** We present the configurations of various flight simulations shown in this paper. The task duration denotes the time of actual physical flight, whereas the time cost represents the duration taken for simulation. The ratio, calculated as the quotient of task duration and time cost, serves to quantify the level of real-time performance, where values approaching one signify higher real-time capability.

Simulations	UAV Type	UAV Size	Task Duration	Resolution	Timestep	Memory	Time Cost	Ratio
Fig. 11	Quadcopter	Wheelbase 250mm	5s	4cm	6.67×10^{-4}s	100MB	7s	71%
Fig. 10	Quadcopter	Wheelbase 600mm	10s	8cm	2.67×10^{-3}s	200MB	9s	111%
Fig. 12	Quadcopter	Wheelbase 600mm	12s	6cm	1.5×10^{-3}s	150MB	8s	150%
Fig. 13 A	Quadcopter	Wheelbase 250mm	12s	4cm	6.67×10^{-4}s	100MB	20s	60%
Fig. 13 B	Quadcopter	Wheelbase 600mm	12s	6cm	1.5×10^{-3}s	150MB	8s	150%
Fig. 14	Fixed-wing	Wingspan 1.13m	6s	3cm	3×10^{-4}s	150MB	30s	20%
Fig. 15	Quadcopter	Wheelbase 130mm	5s	2cm	5×10^{-4}s	150MB	25s	20%
Fig. 16	Tilt-rotor	Body length 9.36m	25s	0.2m	1×10^{-3}s	800MB	150s	17%
Fig. 17	Tilt-rotor	Wingspan 1.1m	6s	3cm	3×10^{-4}s	200MB	40s	15%
Fig. 18	Quadcopters	Wheelbase 600mm/250mm	5s	3cm	3×10^{-4}s	300MB	25s	20%
Fig. 19	Quadcopters	Wheelbase 600mm	12s	6cm	1.5×10^{-3}s	800MB	120s	10%

6 Conclusion

Considering the limitations of existing simulation methodologies for flight controller design and evaluation, this paper introduces a novel paradigm for simulating UAV dynamics and streamlining flight controller design and assessment. Aiming at both computational efficiency and physical consistency, a hybrid simulator is proposed, which decomposes the simulation domain into near-object field where parametric algebraic empirical models are employed and adapted, as well as far field where a novel adaptive block-based GPU-optimized LB solver is developed. These near-object model and the far-field solver are coupled together using a force spreading scheme near the object boundary, enabling highly-efficient fluid-structure interaction simulations. Through model calibration with physical system and controller optimization, our platform facilitates direct transfer of flight controllers to real-world UAVs, demonstrating close trajectory tracking performance. Our simulation system has been validated through physical experiments under diverse conditions, both with and without flight control, to demonstrate the applicability of our simulator in more realistic scenarios. We have showcased different types of flight control applications across various UAV configurations, comparing results against traditional empirical models and real-world flight data to highlight the advantages and practical value of our new simulator in not only aerial robotics but also realistic animations for various flying creatures.

Acknowledgments

The authors sincerely thank the reviewers for their insightful comments, which have significantly improved the quality of this paper. We also acknowledge the contributors of the models used in this study: the P600 UAV model was provided by AMOVLAB [AMOVLAB 2024]; the M0 UAV in Fig. 13 (a) was sourced from CHAOWEIKONGJIAN [CHAOWEIKONGJIAN 2024]; the fixed-wing UAV in Fig. 14 was developed by Eclipson-Airplanes [Eclipson 2025]; the Crazyflie drone in Fig. 15 was from bitcraze [bitcraze 2024]; the "Flying Fish" tilt-rotor UAV in Fig. 16 was created by Ridwan Sept; and the "Hornet" hybrid UAV in Fig. 17 was provided by TITAN DYNAMICS [TITAN DYNAMICS 2025]. This work was supported by the National Natural Science Foundation of China (No. 62072310) and ShanghaiTech University.

References

bitcraze. 2024. Crazyflie 2.1. https://www.bitcraze.io/products/old-products/crazyflie-2-1/

Eclipson. 2025. Eclipson - 3D printed airplanes. https://www.eclipson-airplanes.com/allplanes

Mridul Aanjaneya, Ming Gao, Haixiang Liu, Christopher Batty, and Eftychios Sifakis. 2017. Power diagrams and sparse paged grids for high resolution adaptive liquids. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–12.

Mahdi Abaszadeh, Ali Safavinejad, Hossein Amiri, and Amin Amiri Delouei. 2022. A direct-forcing IB-LBM implementation for thermal radiation in irregular geometries. *Journal of Thermal Analysis and Calorimetry* 147, 20 (2022), 11169–11181.

Bart Adams, Mark Pauly, Richard Keiser, and Leonidas J Guibas. 2007. Adaptively sampled particle fluids. In *ACM SIGGRAPH 2007 papers*. 48–es.

Nadir Akinci, Markus Ihmsen, Gizem Akinci, Barbara Solenthaler, and Matthias Teschner. 2012. Versatile rigid-fluid coupling for incompressible SPH. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 1–8.

Mohammad A Al-Shabi, Khaled S Hatamleh, and Asad A Asad. 2013. UAV dynamics model parameters estimation techniques: A comparison study. In *2013 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT)*. IEEE, 1–6.

Dan A Alcantara, Andrei Sharf, Fatemeh Abbasinejad, Shubhabrata Sengupta, Michael Mitzenmacher, John D Owens, and Nina Amenta. 2009. Real-time parallel hashing on the GPU. In *ACM SIGGRAPH Asia 2009 papers*. 1–9.

Shun-ichi Amari. 1993. Backpropagation and stochastic gradient descent method. *Neurocomputing* 5, 4-5 (1993), 185–196.

AMOVLAB. 2024. Prometheus 600 Scientific Research Drone Development Platform. https://www.amovlab.com/product/detail?pid=76

J. D. Anderson. 2010. *Fundamentals of Aerodynamics*. McGraw-Hill Education.

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

BS Anjali, A Vivek, and JL Nandagopal. 2016. Simulation and analysis of integral LQR controller for inner control loop design of a fixed wing micro aerial vehicle (MAV). *Procedia Technology* 25 (2016), 76–83.

Saeid Asadzadeh, Wilson José de Oliveira, and Carlos Roberto de Souza Filho. 2022. UAV-based remote sensing for the petroleum industry and environmental monitoring: State-of-the-art and perspectives. *Journal of Petroleum Science and Engineering* 208 (2022), 109633.

Henrik Asmuth, Hugo Olivares-Espinosa, Karl Nilsson, and Stefan Ivanell. 2019. The actuator line model in lattice Boltzmann frameworks: numerical sensitivity and computational performance. In *Journal of Physics: Conference Series*, Vol. 1256. IOP Publishing, 012022.

Franck Auguste, Jacques Magnaudet, and David Fabre. 2013. Falling styles of disks. *Journal of Fluid Mechanics* 719 (2013), 388–405.

Ahmad Taher Azar, Anis Koubaa, Nada Ali Mohamed, Habiba A Ibrahim, Zahra Fathy Ibrahim, Muhammad Kazim, Adel Ammar, Bilel Benjdira, Alaa M Khamis, Ibrahim A Hameed, et al. 2021. Drone deep reinforcement learning: A review. *Electronics* 10, 9 (2021), 999.

Dan Bailey, Harry Biddle, Nick Avramoussis, and Matthew Warner. 2015. Distributing liquids using OpenVDB. In *ACM SIGGRAPH 2015 Talks* (Los Angeles, California). Association for Computing Machinery, New York, NY, USA, Article 44, 1 pages.

Somil Bansal, Anayo K Akametalu, Frank J Jiang, Forrest Laine, and Claire J Tomlin. 2016. Learning quadrotor dynamics using neural network for flight control. In *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE, 4653–4660.

G. K. Batchelor. 2000. *An Introduction to Fluid Dynamics*. Cambridge University Press.

Markus Becker and Matthias Teschner. 2007. Weakly compressible SPH for free surface flows. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 209–217.

Albert S Berahas, Liyuan Cao, Krzysztof Choromanski, and Katya Scheinberg. 2022. A theoretical and empirical comparison of gradient approximations in derivative-free optimization. *Foundations of Computational Mathematics* 22, 2 (2022), 507–560.

Morten Bojsen-Hansen, Michael Bang Nielsen, Konstantinos Stamatelos, and Robert Bridson. 2021. Spatially adaptive volume tools in bifrost. In *ACM SIGGRAPH 2021 Talks*. 1–2.

Sanjeeb T. Bose and George Ilhwan Park. 2018. Wall-Modeled Large-Eddy Simulation for Complex Turbulent Flows. *Annual Review of Fluid Mechanics* 50, 1 (2018), 535–561.

Yasser Bouzid, Houria Siguerdidjane, and Yasmina Bestaoui. 2017. Comparative Results on 3D Navigation of Quadrotor using two Nonlinear Model based Controllers. In *Journal of Physics: Conference Series*, Vol. 783. IOP Publishing, 012046.

M'hamed Bouzidi, Mouaouia Firdaouss, and Pierre Lallemand. 2001. Momentum transfer of a Boltzmann-lattice fluid with boundaries. *Physics of Fluids* 13, 11 (2001), 3452–3459.

Peter T Brady and Daniel Livescu. 2021. Foundations for high-order, conservative cut-cell methods: stable discretizations on degenerate meshes. *J. Comput. Phys.* 426 (2021), 109794.

Robert A Brebin, Loren Carpenter, and Pat Hanrahan. 1998. Volume rendering. In *Seminal graphics: pioneering efforts that shaped the field*. 363–372.

Guowei Cai, Kai-Yew Lum, Ben M Chen, and Tong H Lee. 2010. A brief overview on miniature fixed-wing unmanned aerial vehicles. *IEEE ICCA 2010* (2010), 285–290.

Jonathan Camargo, Omar López, and Nicolás Ochoa-Lleras. 2012. A computational tool for unsteady aerodynamic flow simulations coupled with rigid body dynamics and control. In *30th AIAA Applied Aerodynamics Conference*. 3034.

Nicholas Caplan and Trevor N Gardner. 2007. A fluid dynamic investigation of the Big Blade and Macon oar blade designs in rowing propulsion. *Journal of Sports Sciences* 25, 6 (2007), 643–650.

CHAOWEIKONGJIAN. 2024. [Open-source drone] Assemble an open-source drone M0-F250 from scratch. https://www.bilibili.com/video/BV1owSQYTExr

John R Chawner and Nigel J Taylor. 2019. Progress in geometry modeling and mesh generation toward the CFD vision 2030. In *AIAA Aviation 2019 Forum*. 2945.

Sung-Hua Chen, Yen Ku, and Chao-An Lin. 2019. Simulations of settling object using moving domain and immersed-boundary method. *Computers & Fluids* 179 (2019), 735–743.

Yixin Chen, Wei Li, Rui Fan, and Xiaopei Liu. 2021. GPU Optimization for High-Quality Kinetic Fluid Simulation. *IEEE Transactions on Visualization and Computer Graphics* 28, 9 (2021), 3235–3251.

Wen-Chyuan Chiang, Yuyu Li, Jennifer Shang, and Timothy L Urban. 2019. Impact of drone delivery on sustainability and cost: Realizing the UAV potential through vehicle routing optimization. *Applied Energy* 242 (2019), 1164–1175.

Girish Chowdhary and Ravindra Jategaonkar. 2010. Aerodynamic parameter estimation from flight data applying extended and unscented Kalman filter. *Aerospace Science and Technology* 14, 2 (2010), 106–117.

Marcin Chrust, Gilles Bouchet, and Jan Dušek. 2013. Numerical simulation of the dynamics of freely falling discs. *Physics of Fluids* 25, 4 (2013).

Blender Online Community. 2018. *Blender-a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam. http://www.blender.org

Rick Cory and Russ Tedrake. 2008. Experiments in fixed-wing UAV perching. In *AIAA Guidance, Navigation and Control Conference and Exhibit*. 7256.

A Da Ronch, M Ghoreyshi, and KJ Badcock. 2011. On the generation of flight dynamics aerodynamic tables by computational fluid dynamics. *Progress in Aerospace Sciences* 47, 8 (2011), 597–620.

Li Danjun, Zhou Yan, Shi Zongying, and Lu Geng. 2015. Autonomous landing of quadrotor based on ground effect modelling. In *2015 34th Chinese control conference (CCC)*. IEEE, 5647–5652.

Mathieu Desbrun and Marie-Paule Gascuel. 1996. Smoothed particles: A new paradigm for animating highly deformable bodies. In *Computer Animation and Simulation'96: Proceedings of the Eurographics Workshop in Poitiers, France, August 31–September 1, 1996*. Springer, 61–76.

Mark Drela. 1989. XFOIL: An Analysis and Design System for Low Reynolds Number Airfoils. In *Low Reynolds Number Aerodynamics: Proceedings of the Conference Notre Dame, Indiana, USA, 5–7 June 1989*. Springer, 1–12.

Tao Du, Kui Wu, Pingchuan Ma, Sebastien Wah, Andrew Spielberg, Daniela Rus, and Wojciech Matusik. 2021. Diffpd: Differentiable projective dynamics. *ACM Transactions on Graphics (TOG)* 41, 2 (2021), 1–21.

R Elliot English, Linhai Qiu, Yue Yu, and Ronald Fedkiw. 2013a. An adaptive discretization of incompressible flow using a multitude of moving Cartesian grids. *J. Comput. Phys.* 254 (2013), 107–154.

R Elliot English, Linhai Qiu, Yue Yu, and Ronald Fedkiw. 2013b. Chimera grids for water simulation. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 85–94.

Robert Eymard, Thierry Gallouët, and Raphaèle Herbin. 2000. Finite volume methods. *Handbook of Numerical Analysis* 7 (2000), 713–1018.

Roy Featherstone. 2014. *Rigid body dynamics algorithms*. Springer.

Martin L Felis. 2017. RBDL: an efficient rigid-body dynamics library using recursive algorithms. *Autonomous Robots* 41, 2 (2017), 495–511.

Robin Ferede, Guido de Croon, Christophe De Wagter, and Dario Izzo. 2024. End-to-end neural network based optimal quadcopter control. *Robotics and Autonomous Systems* 172 (2024), 104588.

Joel H Ferziger and Milovan Perić. 2002. *Computational methods for fluid dynamics*. Springer.

Reeves Fletcher and Colin M Reeves. 1964. Function minimization by conjugate gradients. *Comput. J.* 7, 2 (1964), 149–154.

Chuyuan Fu, Qi Guo, Theodore Gast, Chenfanfu Jiang, and Joseph Teran. 2017. A polynomial particle-in-cell method. *ACM Transactions on Graphics (TOG)* 36, 6 (2017), 1–12.

G Ganga and Meher Madhu Dharmana. 2017. MPC controller for trajectory tracking control of quadcopter. In *2017 International Conference on Circuit, Power and Computing Technologies (ICCPCT)*. IEEE, 1–6.

Shijie Gao, Carmelo Di Franco, Darius Carter, Daniel Quinn, and Nicola Bezzo. 2019. Exploiting ground and ceiling effects on autonomous UAV motion planning. In *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 768–777.

Michael Gargalakos. 2024. The role of unmanned aerial vehicles in military communications: application scenarios, current trends, and beyond. *The Journal of Defense Modeling and Simulation* 21, 3 (2024), 313–321.

Martin Geier, Andrea Pasquali, and Martin Schönherr. 2017. Parametrization of the Cumulant Lattice Boltzmann Method for Fourth-Order Accurate Diffusion Part I: Derivation and Validation. *J. Comput. Phys.* 348 (2017), 862–888.

Martin Geier, Martin Schönherr, Andrea Pasquali, and Manfred Krafczyk. 2015. The Cumulant Lattice Boltzmann Equation in Three Dimensions: Theory and Validation. *Computers & Mathematics with Applications* 70, 4 (2015), 507–547.

Wojciech Giernacki, Mateusz Skwierczyński, Wojciech Witwicki, Paweł Wroński, and Piotr Kozierski. 2017. Crazyflie 2.0 quadrotor as a platform for research and education in robotics and control engineering. In *2017 22nd International Conference on Methods and Models in Automation and Robotics (MMAR)*. IEEE, 37–42.

Mike Giles. 2008. Jacobi iteration for a Laplace discretisation on a 3D structured grid.

Irina Ginzburg and Dominique d'Humieres. 2003. Multireflection boundary conditions for lattice Boltzmann models. *Physical Review E* 68, 6 (2003), 066614.

Hermann Glauert. 1983. *The elements of aerofoil and airscrew theory*. Cambridge University Press.

Farhad Goodarzi, Daewon Lee, and Taeyoung Lee. 2013. Geometric nonlinear PID control of a quadrotor UAV on SE (3). In *2013 European Control Conference (ECC)*. IEEE, 3845–3850.

Drew Hanover, Philipp Foehn, Sihao Sun, Elia Kaufmann, and Davide Scaramuzza. 2021. Performance, precision, and payloads: Adaptive nonlinear mpc for quadrotors. *IEEE Robotics and Automation Letters* 7, 2 (2021), 690–697.

Xiang He, Marc Calaf, and Kam K Leang. 2017. Modeling and adaptive nonlinear disturbance observer for closed-loop control of in-ground-effects on multi-rotor UAVs. In *Dynamic Systems and Control Conference*, Vol. 58295. American Society of Mechanical Engineers, V003T39A004.

Rama Karl Hoetzlein. 2016. GVDB: Raytracing sparse voxel database structures on the GPU. In *Proceedings of High Performance Graphics*. 109–117.

Philipp Holl, Nils Thuerey, and Vladlen Koltun. 2020. Learning to Control PDEs with Differentiable Physics. In *International Conference on Learning Representations (ICLR)*.

John H Holland. 1975. Adaptation in natural and artificial systems. *University of Michigan Press* 32 (1975).

Libo Huang, Ziyin Qu, Xun Tan, Xinxin Zhang, Dominik L Michels, and Chenfanfu Jiang. 2021. Ships, splashes, and waves on a vast ocean. *ACM Transactions on Graphics (TOG)* 40, 6 (2021), 1–15.

Julian C. R. Hunt, Alan A. Wray, and Parviz Moin. 1988. Eddies, Streams, and Convergence Zones in Turbulent Flows. *Proceedings of the 1988 Summer Program* (1988).

Markus Ihmsen, Jens Orthmann, Barbara Solenthaler, Andreas Kolb, and Matthias Teschner. 2014. SPH Fluids in Computer Graphics. In *Eurographics 2014 - State of the Art Reports*, Sylvain Lefebvre and Michela Spagnuolo (Eds.). The Eurographics Association. doi:10.2312/egst.20141034

Narendra Kumar Jaiswal. 1968. *Priority queues*. Vol. 50. Academic press New York.

Hrvoje Jasak and Tessa Uroić. 2020. Practical computational fluid dynamics with the finite volume method. *Modeling in Engineering Using Innovative Numerical Methods for Solids and Fluids* (2020), 103–161.

Dennis C Jespersen. 2010. Acceleration of a CFD code with a GPU. *Scientific Programming* 18, 3-4 (2010), 193–201.

Fu Jiahe and Li Rui. 2015. Fractional PID and backstepping control for a small quadrotor helicopter. In *2015 34th Chinese Control Conference (CCC)*. IEEE, 5701–5706.

Chenfanfu Jiang, Craig Schroeder, Andrew Selle, Joseph Teran, and Alexey Stomakhin. 2015. The affine particle-in-cell method. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 1–10.

Belkacem Kada and Y Ghazzawi. 2011. Robust PID controller design for an UAV flight control system. In *Proceedings of the World Congress on Engineering and Computer Science*, Vol. 2. 1–6.

Elia Kaufmann, Leonard Bauersfeld, Antonio Loquercio, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. 2023. Champion-level drone racing using deep reinforcement learning. *Nature* 620, 7976 (2023), 982–987.

James Kennedy and Russell Eberhart. 1995. Particle swarm optimization. In *Proceedings of ICNN'95-International Conference on Neural Networks*, Vol. 4. ieee, 1942–1948.

Shahida Khatoon, Dhiraj Gupta, and LK Das. 2014. PID & LQR control for a quadrotor: Modeling and simulation. In *2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE, 796–802.

ByungMoon Kim, Yingjie Liu, Ignacio Llamas, and Jarek Rossignac. 2005. FlowFixer: Using BFECC for Fluid Simulation.. In *NPH*. 51–56.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. *ICLR (Poster)* (2015).

Vladislav Klein. 1989. Estimation of aircraft aerodynamic parameters from flight data. *Progress in Aerospace Sciences* 26, 1 (1989), 1–77.

Stephen Klosterman, Eli Melaas, Jonathan A Wang, Arturo Martinez, Sidni Frederick, John O'Keefe, David A Orwig, Zhuosen Wang, Qingsong Sun, Crystal Schaaf, et al. 2018. Fine-scale perspectives on landscape phenology from unmanned aerial vehicle (UAV) photography. *Agricultural and Forest Meteorology* 248 (2018), 397–407.

Jens Kruger and Rüdiger Westermann. 2003. Acceleration techniques for GPU-based volume rendering. In *IEEE Visualization, 2003. VIS 2003.* IEEE, 287–292.

Timm Krüger, Halim Kusumaatmaja, Alexandr Kuzmin, Orest Shardt, Goncalo Silva, and Erlend Magnus Viggen. 2017. The lattice Boltzmann method. *Springer International Publishing* 10, 978-3 (2017), 4–15.

Anthony JC Ladd. 1994. Numerical simulations of particulate suspensions via a discretized Boltzmann equation. Part 1. Theoretical foundation. *Journal of Fluid Mechanics* 271 (1994), 285–309.

Jeongseok Lee, Michael X. Grey, Sehoon Ha, Tobias Kunz, Sumit Jain, Yuting Ye, Siddhartha S. Srinivasa, Mike Stilman, and C. Karen Liu. 2018. DART: Dynamic Animation and Robotics Toolkit. *The Journal of Open Source Software* 3, 22 (Feb 2018), 500. doi:10.21105/joss.00500

Steve Lesser, Alexey Stomakhin, Gilles Daviet, Joel Wretborn, John Edholm, Noh-Hoon Lee, Eston Schweickart, Xiao Zhai, Sean Flynn, and Andrew Moffat. 2022. Loki: a unified multiphysics simulation framework for production. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 1–20.

Linzhong Li, SJ Sherwin, and Peter W Bearman. 2002. A moving frame of reference algorithm for fluid/structure interaction of rotating and translating bodies. *International Journal for Numerical Methods in Fluids* 38, 2 (2002), 187–206.

Wei Li, Kai Bai, and Xiaopei Liu. 2018. Continuous-scale kinetic fluid simulation. *IEEE Transactions on Visualization and Computer Graphics* 25, 9 (2018), 2694–2709.

Wei Li, Yixin Chen, Mathieu Desbrun, Changxi Zheng, and Xiaopei Liu. 2020. Fast and scalable turbulent flow simulation with two-way coupling. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 47.

Wei Li, Yihui Ma, Xiaopei Liu, and Mathieu Desbrun. 2022. Efficient kinetic simulation of two-phase flows. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 114.

Chi Harold Liu, Zheyu Chen, Jian Tang, Jie Xu, and Chengzhe Piao. 2018. Energy-efficient UAV control for effective and fair communication coverage: A deep reinforcement learning approach. *IEEE Journal on Selected Areas in Communications* 36, 9 (2018), 2059–2070.

Mengyun Liu, Kai Bai, and Xiaopei Liu. 2025. A Hybrid Near-wall Model for Kinetic Simulation of Turbulent Boundary Layer Flows. *ACM Transactions on Graphics (TOG)* 44, 4, Article 148 (July 2025), 24 pages. doi:10.1145/3730829

Mengyun Liu and Xiaopei Liu. 2023. A Parametric Kinetic Solver for Simulating Boundary-Dominated Turbulent Flow Phenomena. *ACM Transactions on Graphics (TOG)* 42, 6 (2023), 1–20.

Wenji Liu, Kai Bai, Xuming He, Shuran Song, Changxi Zheng, and Xiaopei Liu. 2022. Fishgym: A high-performance physics-based simulation framework for underwater robot learning. In *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 6268–6275.

Lennart Ljung. 1998. System identification. In *Signal analysis and prediction*. Springer, 163–173.

Ivan Lopez-Sanchez and Javier Moreno-Valenzuela. 2023. PID control of quadrotor UAVs: A survey. *Annual Reviews in Control* 56 (2023), 100900. doi:10.1016/j.arcontrol.2023.100900

Arko Lucieer, Steven M de Jong, and Darren Turner. 2014. Mapping landslide displacements using Structure from Motion (SfM) and image correlation of multi-temporal UAV photography. *Progress in Physical Geography* 38, 1 (2014), 97–116.

Chaoyang Lyu, Kai Bai, Yiheng Wu, Mathieu Desbrun, Changxi Zheng, and Xiaopei Liu. 2023. Building a Virtual Weakly-Compressible Wind Tunnel Testing Facility. *ACM Transactions on Graphics (TOG)* 42, 4 (2023), 1–20.

Chaoyang Lyu, Wei Li, Mathieu Desbrun, and Xiaopei Liu. 2021. Fast and versatile fluid-solid coupling for turbulent flow simulation. *ACM Transactions on Graphics (TOG)* 40, 6 (2021), 201.

Miles Macklin, Kenny Erleben, Matthias Müller, Nuttapong Chentanez, Stefan Jeschke, and Tae-Yong Kim. 2020. Primal/dual descent methods for dynamics. In *Computer Graphics Forum*, Vol. 39. Wiley Online Library, 89–100.

Orestis Malaspinas. 2015. Increasing stability and accuracy of the lattice Boltzmann scheme: recursivity and regularization. *arXiv preprint arXiv:1505.06900* (2015).

Francesco Marson, Yann Thorimbert, Bastien Chopard, Irina Ginzburg, and Jonas Latt. 2021. Enhanced single-node lattice Boltzmann boundary condition for fluid flows. *Physical Review E* 103, 5 (2021), 053308.

Koppány Máthé and Lucian Buşoniu. 2015. Vision and control for UAVs: A survey of general methods and of inexpensive platforms for infrastructure inspection. *Sensors* 15, 7 (2015), 14887–14916.

Antonio Matus-Vargas, Gustavo Rodriguez-Gomez, and Jose Martinez-Carranza. 2021. Ground effect on rotorcraft unmanned aerial vehicles: A review. *Intelligent Service Robotics* 14, 1 (2021), 99–118.

Mesuli B Mbanjwa, Kevin Harding, and Irvy MA Gledhill. 2022. Numerical Modelling of Mixing in a Microfluidic Droplet Using a Two-Phase Moving Frame of Reference Approach. *Micromachines* 13, 5 (2022), 708.

Lorenz Meier, Dominik Honegger, and Marc Pollefeys. 2015. PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 6235–6240.

Anshul Mittal, Kidambi Sreenivas, Lafayette K Taylor, and Levi Hereth. 2015. Improvements to the actuator line modeling for wind turbines. In *33rd Wind Energy Symposium*. 0216.

Hongwei Mo and Ghulam Farid. 2019. Nonlinear and adaptive intelligent control techniques for quadrotor uav–a survey. *Asian Journal of Control* 21, 2 (2019), 989–1008.

Jorge J Moré and Danny C Sorensen. 1982. *Newton's method*. Technical Report. Argonne National Lab.(ANL), Argonne, IL (United States).

Fadl Moukalled, Luca Mangani, Marwan Darwish, F Moukalled, L Mangani, and M Darwish. 2016. *The finite volume method*. Springer.

Matthias Müller, David Charypar, and Markus Gross. 2003. Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Citeseer, 154–159.

Ken Museth. 2013. VDB: High-resolution sparse volumes with dynamic topology. *ACM Transactions on Graphics (TOG)* 32, 3 (2013), 1–22.

Franck Nicoud and Frédéric Ducros. 1999. Subgrid-Scale Stress Modelling Based on the Square of the Velocity Gradient Tensor. *Flow, Turbulence and Combustion* 62, 3 (1999), 183–200.

Christian Obrecht, Frédéric Kuznik, Bernard Tourancheau, and Jean-Jacques Roux. 2011. A new approach to the lattice Boltzmann method for graphics processing units. *Computers & Mathematics with Applications* 61, 12 (2011), 3628–3638.

Sang Il Park and Myoung Jun Kim. 2005. Vortex fluid for gaseous phenomena. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 261–270.

C Paz, E Suárez, C Gil, and C Baker. 2020. CFD analysis of the aerodynamic effects on the stability of the flight of a quadcopter UAV in the proximity of walls and ground. *Journal of Wind Engineering and Industrial Aerodynamics* 206 (2020), 104378.

Charles S Peskin. 1972. Flow patterns around heart valves: a numerical method. *J. Comput. Phys.* 10, 2 (1972), 252–271.

Matěj Petrlík, Tomáš Báča, Daniel Heřt, Matouš Vrba, Tomáš Krajník, and Martin Saska. 2020. A robust UAV system for operations in a constrained environment. *IEEE Robotics and Automation Letters* 5, 2 (2020), 2169–2176.

Stephen B. Pope. 2000. *Turbulent Flows*. Cambridge University Press.

Yuxing Qiu, Samuel Temple Reeve, Minchen Li, Yin Yang, Stuart Ryan Slattery, and Chenfanfu Jiang. 2023. A sparse distributed gigascale resolution material point method. *ACM Transactions on Graphics (TOG)* 42, 2 (2023), 1–21.

Ziyin Qu, Xinxin Zhang, Ming Gao, Chenfanfu Jiang, and Baoquan Chen. 2019. Efficient and conservative fluids using bidirectional mapping. *ACM Trans. Graph.* 38, 4, Article 128 (jul 2019), 12 pages. doi:10.1145/3306346.3322945

Quan Quan. 2017. *Introduction to multicopter design and control*. Springer.

Wouter Raateland, Torsten Hädrich, Jorge Alejandro Amador Herrera, Daniel T. Banuti, Wojciech Pałubicki, Sören Pirk, Klaus Hildebrandt, and Dominik L. Michels. 2022. DCGrid: An Adaptive Grid Structure for Memory-Constrained Fluid Simulation on the GPU. *Proc. ACM Comput. Graph. Interact. Tech.* 5, 1, Article 3 (May 2022), 14 pages. doi:10.1145/3522608

Karthik Raveendran, Chris Wojtan, and Greg Turk. 2011. Hybrid smoothed particle hydrodynamics. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 33–42.

Daniel Raymer. 2012. *Aircraft design: a conceptual approach*. American Institute of Aeronautics and Astronautics, Inc.

Pablo R Rinaldi, EA Dari, Marcelo J Vénere, and Alejandro Clausse. 2012. A Lattice-Boltzmann solver for 3D fluid simulation on GPU. *Simulation Modelling Practice and Theory* 25 (2012), 163–171.

Avi Robinson-Mosher, Tamar Shinar, Jon Gretarsson, Jonathan Su, and Ronald Fedkiw. 2008. Two-way coupling of fluids to rigid and deformable solids and shells. *ACM Transactions on Graphics (TOG)* 27, 3 (2008), 1–9.

Kirill V Rozhdestvensky. 2006. Wing-in-ground effect vehicles. *Progress in Aerospace Sciences* 42, 3 (2006), 211–283.

MH Sabour, P Jafary, and S Nematiyan. 2023. Applications and classifications of unmanned aerial vehicles: A literature review with focus on multi-rotors. *The Aeronautical Journal* 127, 1309 (2023), 466–490.

Atheer Salih, Mahmoud Moghavvemi, and Haider AF Mohamed. 2010. Flight PID controller design for a UAV quadrotor. *Scientific Research and Essays (SRE)* 5, 23 (2010), 3660–3667.

Pedro Sanchez-Cuevas, Guillermo Heredia, and Anibal Ollero. 2017. Characterization of the aerodynamic ground effect and its influence in multirotor control. *International Journal of Aerospace Engineering* 2017, 1 (2017), 1823056.

Prathamesh Saraf, Manan Gupta, and Alivelu Manga Parimi. 2020. A comparative study between a classical and optimal controller for a quadrotor. In *2020 IEEE 17th India Council International Conference (INDICON)*. IEEE, 1–6.

Andrew Selle, Ronald Fedkiw, Byungmoon Kim, Yingjie Liu, and Jarek Rossignac. 2008. An unconditionally stable MacCormack method. *Journal of Scientific Computing* 35 (2008), 350–371.

Andrew Selle, Nick Rasmussen, and Ronald Fedkiw. 2005. A vortex particle method for smoke, water and explosions. In *ACM SIGGRAPH 2005 Papers*. 910–914.

Rajsekhar Setaluri, Mridul Aanjaneya, Sean Bauer, and Eftychios Sifakis. 2014. SPGrid: A sparse paged grid structure applied to adaptive smoke simulation. *ACM Transactions on Graphics (TOG)* 33, 6 (2014), 1–12.

Shang-En Shen and Yi-Cheng Huang. 2024. Application of reinforcement learning in controlling quadrotor UAV flight actions. *Drones* 8, 11 (2024), 660.

Ratnesh K Shukla, Mahidhar Tatineni, and Xiaolin Zhong. 2007. Very high-order compact finite difference schemes on non-uniform grids for incompressible Navier–Stokes equations. *J. Comput. Phys.* 224, 2 (2007), 1064–1094.

Yousuf Soliman, Marcel Padilla, Oliver Gross, Felix Knöppel, Ulrich Pinkall, and Peter Schröder. 2024. Going with the Flow. *ACM Transactions on Graphics (TOG)* 43, 4 (2024), 1–12.

Wenbin Song, Heng Zhang, Yang Wang, and Xiaopei Liu. 2025. Creating Fluid-Interactive Virtual Agents by an Efficient Simulator with Local-domain Control. *ACM Transactions on Graphics (TOG)* 44, 4 (2025).

Yunlong Song, Mats Steinweg, Elia Kaufmann, and Davide Scaramuzza. 2021. Autonomous drone racing with deep reinforcement learning. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 1205–1212.

Jos Stam. 1999. Stable fluids. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '99)*. ACM Press/Addison-Wesley Publishing Co., USA, 121–128. doi:10.1145/311535.311548

Brian L Stevens, Frank L Lewis, and Eric N Johnson. 2015. *Aircraft control and simulation: dynamics, controls design, and autonomous systems*. John Wiley & Sons.

John C Strikwerda. 2004. *Finite difference schemes and partial differential equations*. SIAM.

Barna Szabó and Ivo Babuška. 2021. Finite element analysis: Method, verification and validation. (2021).

Farhang Tahmasebi, Robert Zach, Matthias Schuß, and Ardeshir Mahdavi. 2012. Simulation model calibration: An optimization-based approach. IBPSA.

Shintaro Takeuchi, Takahiro Yamazaki, and Takeo Kajishima. 2006. Study of solid-fluid interaction in body-fixed non-inertial frame of reference. *Journal of Fluid Science and Technology* 1, 1 (2006), 1–11.

Jie Tan, Yuting Gu, Greg Turk, and C Karen Liu. 2011. Articulated swimming creatures. *ACM Transactions on Graphics (TOG)* 30, 4 (2011), 1–12.

Michael Tao, Christopher Batty, Mirela Ben-Chen, Eugene Fiume, and David IW Levin. 2022. VEMPIC: particle-in-polyhedron fluid simulation for intricate solid boundaries. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 1–22.

Shi Tao, Qing He, Baiman Chen, Xiaoping Yang, and Simin Huang. 2018. One-point second-order curved boundary condition for lattice Boltzmann simulation of suspended particles. *Computers & Mathematics with Applications* 76, 7 (2018), 1593–1607.

The OpenFOAM Foundation. 2024. *OpenFOAM User Guide*. https://www.openfoam.com/documentation/guides/latest/doc/ Accessed: 2024-12-31.

TITAN DYNAMICS. 2025. Hornet VTOL. https://titandynamics.aero/free/p/hornet-vtol

Niels Troldborg. 2009. Actuator line modeling of wind turbine wakes. (2009).

Dimosthenis C Tsouros, Stamatia Bibi, and Panagiotis G Sarigiannidis. 2019. A review on UAV-based applications for precision agriculture. *Information* 10, 11 (2019), 349.

Jiyuan Tu, Guan Heng Yeoh, Chaoqun Liu, and Yao Tao. 2023. *Computational fluid dynamics: a practical approach*. Elsevier.

Long Luong Tuan and Sangchul Won. 2013. PID based sliding mode controller design for the micro quadrotor. In *2013 13th International Conference on Control, Automation and Systems (ICCAS 2013)*. IEEE, 1860–1865.

D Vallespin, KJ Badcock, A Da Ronch, MD White, P Perfect, and M Ghoreyshi. 2012. Computational fluid dynamics framework for aerodynamic model assessment. *Progress in Aerospace Sciences* 52 (2012), 2–18.

Gerhard Venter. 2010. Review of optimization techniques. (2010).

Alejandro F Villaverde, Dilan Pathirana, Fabian Fröhlich, Jan Hasenauer, and Julio R Banga. 2022. A protocol for dynamic model calibration. *Briefings in Bioinformatics* 23, 1 (2022), bbab387.

Mengdi Wang, Fan Feng, Junlin Li, and Bo Zhu. 2025. Cirrus: Adaptive Hybrid Particle-Grid Flow Maps on GPU. *ACM Transactions on Graphics (TOG)* 44, 4 (2025), 1–17.

Jens Wittenburg. 2013. *Dynamics of systems of rigid bodies*. Vol. 33. Springer-Verlag.

Jie Wu and Chang Shu. 2009. Implicit velocity correction-based immersed boundary-lattice Boltzmann method and its applications. *J. Comput. Phys.* 228, 6 (2009), 1963–1979.

Kui Wu, Nghia Truong, Cem Yuksel, and Rama Hoetzlein. 2018. Fast fluid simulations with sparse volumes on the GPU. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library, 157–167.

Kun Xiao, Yao Meng, Xunhua Dai, Haotian Zhang, and Quan Quan. 2021. A lifting wing fixed on multirotor UAVs for long flight ranges. In *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 1605–1610.

Xiaoyu Xiao, Ding Lin, Yiheng Wu, Kai Bai, and Xiaopei Liu. 2025. Simulating Two-phase Fluid-rigid Interactions with an Overset-Grid Kinetic Solver. *IEEE Transactions on Visualization and Computer Graphics* (2025).

Haoran Xie, Takeo Igarashi, and Kazunori Miyata. 2018. Precomputed panel solver for aerodynamics simulation. *ACM Transactions on Graphics (TOG)* 37, 2 (2018), 1–12.

Jie Xu, Tao Du, Michael Foshey, Beichen Li, Bo Zhu, Adriana Schulz, and Wojciech Matusik. 2019. Learning to fly: computational controller design for hybrid uavs with reinforcement learning. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 1–12.

Yuan-Qing Xu, Fang-Bao Tian, and Yu-Lin Deng. 2013. An efficient red blood cell model in the frame of IB-LBM and its application. *International Journal of Biomathematics* 6, 01 (2013), 1250061.

Jinpeng Yang, Zhihao Cai, Qing Lin, and Yingxun Wang. 2013. Self-tuning PID control design for quadrotor UAV based on adaptive pole placement control. In *2013 Chinese Automation Congress*. IEEE, 233–237.

Sen Yang and Bin Xian. 2019. Energy-based nonlinear adaptive control design for the quadrotor UAV system with a suspended payload. *IEEE Transactions on Industrial Electronics* 67, 3 (2019), 2054–2064.

Xiaolei Yang and Fotis Sotiropoulos. 2018. A new class of actuator surface models for wind turbines. *Wind Energy* 21, 5 (2018), 285–302.

Dazhi Yu, Renwei Mei, and Wei Shyy. 2003. A unified boundary treatment in lattice boltzmann method. In *41st aerospace sciences meeting and exhibit*. 953.

Xinjie Yu and Mitsuo Gen. 2010. *Introduction to evolutionary algorithms*. Springer Science & Business Media.

Cem Yuksel. 2015. Sample elimination for generating poisson disk sample sets. In *Computer Graphics Forum*, Vol. 34. Wiley Online Library, 25–32.

Omar Zarifi. 2020. Sparse smoke simulations in Houdini. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Talks*. 1–2.

Jonas Zehnder, Rahul Narain, and Bernhard Thomaszewski. 2018. An advection-reflection solver for detail-preserving fluid simulation. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–8.

Weifeng Zhao and Wen-An Yong. 2017. Single-node second-order boundary schemes for the lattice Boltzmann method. *J. Comput. Phys.* 329 (2017), 1–15.

Xiaohui Zhou, Ke Xie, Kai Huang, Yilin Liu, Yang Zhou, Minglun Gong, and Hui Huang. 2020. Offsite aerial path planning for efficient urban scene reconstruction. *ACM Transactions on Graphics (TOG)* 39, 6 (2020), 1–16.

Olek C Zienkiewicz, Robert L Taylor, and Perumal Nithiarasu. 2013. *The finite element method for fluid dynamics*. Butterworth-Heinemann.