HelmFluid: Learning Helmholtz Dynamics for Interpretable Fluid Prediction

Lanxiang Xing [*1] **Haixu Wu** [*1] **Yuezhou Ma** [1] **Jianmin Wang** [1] **Mingsheng Long** [1]

Abstract

Fluid prediction is a long-standing challenge due to the intrinsic high-dimensional non-linear dynamics. Previous methods usually utilize the non-linear modeling capability of deep models to directly estimate velocity fields for future prediction. However, skipping over inherent physical properties but directly learning superficial velocity fields will overwhelm the model from generating precise or physics-reliable results. In this paper, we propose the *HelmFluid* toward an accurate and interpretable predictor for fluid. Inspired by the Helmholtz theorem, we design a *HelmDynamics* block to learn *Helmholtz dynamics*, which decomposes fluid dynamics into more solvable curl-free and divergence-free parts, physically corresponding to potential and stream functions of fluid. By embedding the HelmDynamics block into a *Multi-scale Multihead Integral Architecture*, HelmFluid can integrate learned Helmholtz dynamics along temporal dimension in multiple spatial scales to yield future fluid. Compared with previous velocity estimating methods, HelmFluid is faithfully derived from Helmholtz theorem and ravels out complex fluid dynamics with physically interpretable evidence. Experimentally, HelmFluid achieves consistent state-of-the-art in both numerical simulated and real-world observed benchmarks, even for scenarios with complex boundaries. Code is available at https://github.com/thuml/HelmFluid.

1. Introduction

Fluid is one of the basic substances in the physical world. Its prediction is of immense importance in extensive real-world applications, such as atmospheric prediction for weather forecasting and airflow modeling for airfoil design, which has attracted significant attention from both science and engineering areas. However, it is quite challenging to capture and predict the intricate high-dimensional non-linear dynamics within the fluid due to imperfect observations, coupled multiscale interactions, etc. In this paper, we focus on a more practical scenario that predicts future states of fluid from *partially observed* physical quantities.

Recently, deep models have achieved impressive progress in solving complex physical systems (Karniadakis et al., 2021; Wang et al., 2023). One paradigm is learning neural operators to directly predict the future fluid field based on past observations (Lu et al., 2021a; Li et al., 2021; Wu et al., 2023). These methods focus on leveraging the non-linear modeling capacity of deep models to approximate complex mappings between past and future fluids. However, directly learning neural operators may fail to generate interpretable evidence for prediction results and incur uncontrolled errors. Another mainstreaming paradigm attempts to estimate the dynamic fields of fluid with deep models for future prediction. It is notable that the superficial dynamics are actually driven by underlying physical rules. Directly estimating the velocity fields regarding less physical properties may overwhelm the model from generating precise and plausible prediction results (Sun et al., 2018; Zhang et al., 2022). As shown in Figure 1, it is hard to directly capture the complex dynamics of fluid, where the learned dynamics will be too tanglesome to guide the fluid prediction.

To tackle the above challenges, we attempt to capture the intricate dynamics with physical insights for accurate and interpretable fluid prediction. In this paper, we dive into the physical properties of fluid and propose the *Helmholtz dynamics* as a new paradigm to represent fluid dynamics. Concretely, Helmholtz dynamics is inspired by the Helmholtz theorem (Bladel, 1959) and attributes the intricate dynamics to the potential and stream functions of fluid, which are intrinsic physical quantities of fluid and can directly derive the curl-free and divergence-free parts of fluid respectively. Compared with superficial velocity fields, our proposed Helmholtz dynamics decompose the intricate dynamics into more solvable components, thereby easing the dynamics learning process of deep models. Besides, this new dynamics requires the model to learn the inherent properties of fluid explicitly, which also empowers the prediction with endogenetic physical interpretability.

[*]Equal contribution [1]School of Software, BNRist, Tsinghua University. Lanxiang Xing <xlx22@mails.tsinghua.edu.cn>. Correspondence to: Mingsheng Long <mingsheng@tsinghua.edu.cn>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

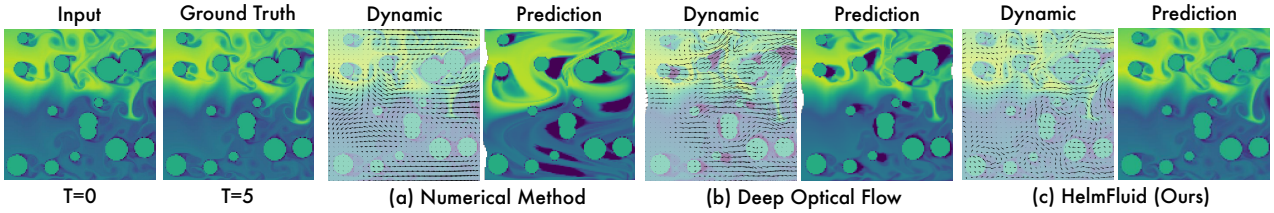


Figure 1. Comparison on dynamics and fluid modeling. Different from the numerical method (Ruzanski et al., 2011) and optical-flow-based deep model (Sun et al., 2018), HelmFluid infers the dynamics from the inherent physics quantities: potential and stream functions.

Based on the above ideas, we present the *HelmFluid* model with novel *HelmDynamics* blocks to capture the Helmholtz dynamics for interpretable fluid prediction. HelmDynamics is faithfully implemented from the Helmholtz decomposition, which can separately estimate the potential and stream functions of fluid from learned spatiotemporal correlations and further derive curl-free and divergence-free velocities. As a flexible module, HelmDynamics can conveniently encode boundary conditions into the correlation calculation process and adapt to complex boundary settings in multifarious real-world applications. Further, we design the *Multi-scale Multihead Integral Architecture* in HelmFluid to fit the multiscale nature of fluid, which can integrate Helmholtz dynamics learned by HelmDynamics blocks along temporal dimension in multiple spatial scales to predict the future fluid. Experimentally, HelmFluid achieves consistent state-of-the-art in various scenarios, covering both synthetic and real-world benchmarks with complex boundary settings. Our contributions are summarized in the following:

- Inspired by the Helmholtz theorem, we propose the *Helmholtz dynamics* to attribute intricate dynamics into inherent properties of fluid, which decomposes intricate dynamics into more solvable parts and empowers the prediction process with physical interpretability.
- We propose *HelmFluid* with the *HelmDynamics block* to capture Helmholtz dynamics. By integrating learned dynamics along temporal dimension with the *Multi-scale Multihead Integral Architecture*, HelmFluid can predict future fluid with physically plausible evidence.
- HelmFluid achieves consistent state-of-the-art in extensive benchmarks, covering both synthetic and real-world datasets, as well as various boundary conditions.

2. Related Work

As a foundation problem in science and engineering areas, fluid prediction has been widely explored. Traditional methods can solve Navier-Stokes equations with numerical algorithms, while they may fail in the real-world fluid due to imperfect observations of initial conditions and inaccurate estimation of equation parameters. Besides, these numerical methods also suffer from huge computation cost. Recently, owing to the great non-linear modeling capacity, data-driven deep models for fluid prediction have attached substantial interests, which can be roughly categorized into the following paradigms according to whether learning velocity fields explicitly or not.

Neural fluid simulator This paradigm of works attempts to directly generate future fluid with deep models. One direction is formulating partial differential equations (PDEs) along with initial and boundary conditions as loss function terms, and parameterizing the solution as a deep model (Evans, 2010; Raissi et al., 2019; 2020; Lu et al., 2021b). These approaches rely highly on exact physics equations, thereby suffering from imperfect observations and inherent randomness in real-world applications. Another branch of methods does not require the exact formulation of governing PDEs. They attempt to learn neural operators to approximate complex input-output mappings in scientific tasks, which enables the prediction of future fluid solely based on past observations. For example, (Lu et al., 2021a) proposed DeepONet in a branch-trunk framework with proven universal approximation capability. FNO (Li et al., 2021) approximates the integral operator through a linear transformation in the Fourier domain. Afterward, U-NO (Rahman et al., 2023) enhances FNO with a multi-scale framework. Later, Wu et al. (2023) proposed latent spectral models (LSM) to solve high-dimensional PDEs in the latent space by learning multiple basis operators. Still, these methods may fail to provide interpretable evidence for prediction results, such as intuitive physics quantities or visible velocity fields. Going beyond the above-mentioned methods, we propose HelmFluid as a purely data-driven model but with special designs to enhance physical interpretability.

Fluid dynamics modeling Estimating velocity fields is a direct and intuitive way of predicting the future fluid. Typically, optical flow (Horn & Schunck, 1981) is proposed to describe the motion between two successive observations. Recently, many deep models have been proposed to estimate optical flow, such as PWC-Net (Sun et al., 2018) and RAFT (Teed & Deng, 2020). However, since the optical flow was originally designed for rigid bodies, it struggles seriously in capturing fluid motion and will bring serious accumulation

errors in the prediction process. Especially for fluid, Zhang et al. incorporated physical constraints from Navier-Stokes equations to refine the velocity field predicted by PWC-Net (Sun et al., 2018) and further embedded the advection-diffusion equation into the deep model to predict the future fluid. Recently, Vortex (Deng et al., 2023) ensembles the observable Eulerian flow and the hidden Lagrangian vortical evolution to capture the intricate dynamics within the fluid. Unlike previous works, we propose to learn the inherent physical quantities of Helmholtz dynamics that explicitly derive the velocity fields, and further predict the future fluid with the Runge–Kutta temporal integral. This decomposes the intricate dynamics into more solvable components and facilitates our model with physical interpretability.

Computer graphics for fluid simulation Solving Navier-Stokes equations with learning-based computer graphics methods often uses a stream function paradigm to enforce the incompressibility condition (Ando et al., 2015). Kim et al. successfully synthesized plausible and divergence-free 2D and 3D fluid velocities from a set of reduced parameters but requiring ground truth velocity supervision, a rarity in real-world data. Recently, Liu et al. estimated the underlying physics of advection-diffusion equations, incorporating ground truth velocity and diffusion tensors supervision. Franz et al. simulated a realistic 3D density and velocity sequence from single-view sequences without 3D supervision, but it is not designed for predictive tasks as it utilizes future information to calculate current density. Unlike previous methods, our method learns the velocity field end-to-end from physical quantities observed in the past via Helmholtz dynamics, relying neither on ground truth velocity supervision nor on stream function. Such an unsupervised paradigm enables our model to capture more intricate fluid dynamics and extends its capability to a broader range of scenarios.

3. HelmFluid

In this paper, we highlight the key components of fluid prediction as providing physical interpretability and handling intricate dynamics. To achieve these objectives, we present the HelmFluid model with *HelmDynamics* blocks to capture the *Helmholtz dynamics* for 2D fluid, which is inspired by the Helmholtz theorem and attributes superficial complex dynamics to the inherent properties of fluid. Further, we design the *Multiscale Multihead Integral Architecture* to integrate the learned dynamics along the temporal dimension in multiple scales to predict the future states of fluid.

3.1. Learning Helmholtz Dynamics

Learning intricate velocity fluid directly from data may overwhelm the model. Hence, we propose to learn Helmholtz dynamics via a HelmDynamics block, which is a faithful implementation of the the Helmholtz theorem that decomposes complex fluid dynamics into more solvable components.

Helmholtz decomposition theorem Helmholtz decomposition (Bladel, 1959) plays an important role in simulating fluid dynamics, which can decompose a dynamic field into a curl-free component and a divergence-free component for simplification, and is highly related to the solvability theory of Navier-Stokes equations (Faith A., 2013).

Given a 3D dynamic field $\mathbf{F} : \mathbb{V} \to \mathbb{R}^3$ with a bounded domain $\mathbb{V} \subseteq \mathbb{R}^3$, we can obtain the following decomposition based on the Helmholtz theorem:

$$\mathbf{F}(\mathbf{r}) = \nabla\Phi(\mathbf{r}) + \nabla \times \mathbf{A}(\mathbf{r}),\ \mathbf{r} \in \mathbb{V}. \tag{1}$$

It is notable that $\Phi : \mathbb{V} \to \mathbb{R}$ denotes the *potential function*, which is a scalar field with its gradient field $\nabla\Phi$ representing the curl-free part of $\mathbf{F}$ guaranteed by $\nabla \times (\nabla\Phi) = \mathbf{0}$. And $\mathbf{A} : \mathbb{V} \to \mathbb{R}^3$ denotes the *stream function*, which is a vector field with $\nabla \times \mathbf{A}$ represents the divergence-free part of $\mathbf{F}$ guaranteed by $\nabla(\nabla \times \mathbf{A}) = \mathbf{0}$, thereby also indicating the incompressibility of the flow field.

Helmholtz dynamics for 2D fluid Following mainstream works and conventional settings (Li et al., 2021), and for conciseness of presentation, this paper presents the *HelmFluid* model on 2D fluid prediction. We project the Helmholtz theorem into 2D space by restricting the z-axis component of $\mathbf{F}$ to 0, i.e. $\mathbf{F}(\mathbf{r}) = (\mathbf{F}_x(\mathbf{r}), \mathbf{F}_y(\mathbf{r}), 0)^\mathsf{T}$. This restriction also vanishes the components of the stream function along x-axis and y-axis, namely $\mathbf{A}(\mathbf{r}) = ((0, 0, \mathbf{A}_z(\mathbf{r}))^\mathsf{T}$, indicating that the stream function degenerates to a scalar field. Generalizations and experiments on more practical 3D fluid prediction are included in Appendix A.

According to the Helmholtz decomposition theorem (Eq. 1), the fluid dynamics can be equivalently decomposed into *curl-free* and *divergence-free* parts for simplification. Thus, we define Helmholtz dynamics $\mathbf{F}_{\text{Helm}}(\Phi, \mathbf{A})$ explicitly as the function of potential and stream functions, which are inherent physics quantities of fluid. Concretely, for a 2D fluid defined in the domain $\mathbb{V} \subseteq \mathbb{R}^2$, its Helmholtz dynamics $\mathbf{F}_{\text{Helm}}$ can be formalized by potential function $\Phi : \mathbb{V} \to \mathbb{R}$ and stream function $\mathbf{A} : \mathbb{V} \to \mathbb{R}$ of fluid as follows:

$$\begin{aligned}\mathbf{F}_{\text{Helm}}(\Phi, \mathbf{A}) &= \nabla\Phi + \nabla \times \mathbf{A} \\ &= \underbrace{\left(\frac{\partial\Phi}{\partial x}, \frac{\partial\Phi}{\partial y}\right)}_{\text{Curl-free Velocity}} + \underbrace{\left(\frac{\partial\mathbf{A}}{\partial y}, -\frac{\partial\mathbf{A}}{\partial x}\right)}_{\text{Divergence-free Velocity}}.\end{aligned} \tag{2}$$

According to the Helmholtz theorem (Eq. 1), the function value of $\mathbf{F}_{\text{Helm}}$ is equivalent to the real dynamic field $\mathbf{F}$ but is more tractable. By incorporating Φ and $\mathbf{A}$, Helmholtz dynamics naturally decomposes the intricate fluid into more solvable components and ravels out the complex dynamics into intrinsic physics quantities, thus benefiting more interpretable dynamics modeling (Bhatia et al., 2013).

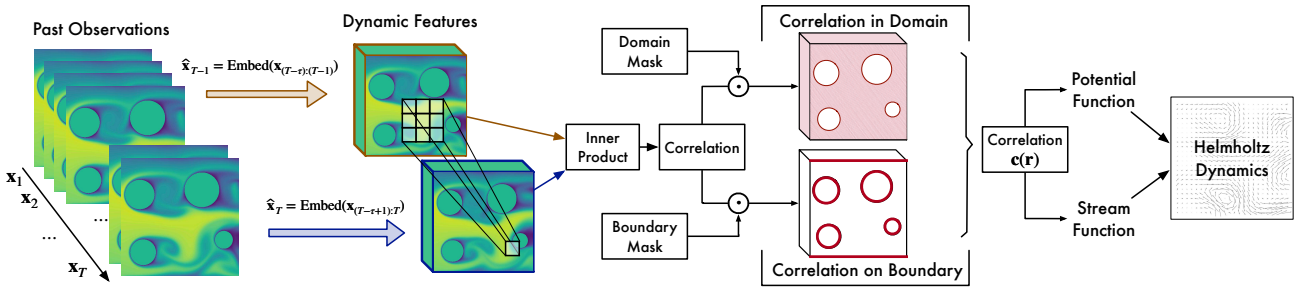


Figure 2. HelmDynamics block, which learns spatiotemporal correlations $\mathbf{c}(\mathbf{r})$ both in the domain and on the boundary to estimate potential and stream functions of fluid from past observations for composing the Helmholtz dynamics.

HelmDynamics block To learn the Helmholtz dynamics, we propose the HelmDynamics block to estimate the potential and stream functions from past observations. As shown in Figure 2, we first embed input observations into two successive deep representations to keep the temporal dynamics information explicitly. Given a sequence of T frames $\mathbf{x} = [\mathbf{x}_1, \cdots, \mathbf{x}_T], \mathbf{x}_i \in \mathbb{R}^{H\times W}$ successively observed from 2D fluid, this process can be written as

$$\begin{aligned} \widehat{\mathbf{x}}_{T-1} &= \text{Embed}\left(\mathbf{x}_{(T-\tau):(T-1)}\right) \\ \widehat{\mathbf{x}}_{T} &= \text{Embed}\left(\mathbf{x}_{(T-\tau+1):T}\right), \end{aligned} \tag{3}$$

where $\widehat{\mathbf{x}}_{T-1}, \widehat{\mathbf{x}}_T \in \mathbb{R}^{d_{\text{model}}\times H\times W}$ are the feature tensors at timestamps $T-1$ and T respectively. Here, we embed the observations from a τ lookback window to capture the spatiotemporal information, which is to project the temporal dimension τ to the channel dimension d_{model} by two convolutional layers with an in-between activation function.

Next, following the convention in dynamics modeling (Sun et al., 2018; Teed & Deng, 2020), we adopt *spatiotemporal correlations* between fluid at the previous timestamp and the current timestamp to represent the dynamics information. Especially as physics quantities of fluid are highly affected by boundary conditions, we go beyond previous approaches and propose to further include boundary conditions $\mathbb{S}$ when calculating the spatiotemporal correlations:

$$\begin{aligned} \mathbf{c}(\mathbf{r}) = \text{Concat}\Big(&\left[\widehat{\mathbf{x}}_T(\mathbf{r}) \cdot \widehat{\mathbf{x}}_{T-1}(\mathbf{r}')\right]_{\mathbf{r}'\in\mathbf{N}_\mathbf{r}}, \\ &\left[\mathbb{1}_{\mathbb{S}}(\mathbf{r}')\left(\widehat{\mathbf{x}}_T(\mathbf{r}) \cdot \widehat{\mathbf{x}}_{T-1}(\mathbf{r}')\right)\right]_{\mathbf{r}'\in\mathbf{N}_\mathbf{r}}\Big),\ \mathbf{r}\in\mathbb{V} \end{aligned} \tag{4}$$

where $\cdot$ denotes the inner-product operation and $\mathbf{N}_\mathbf{r}$ denotes the neighbors around position $\mathbf{r}$. $\mathbb{1}_{\mathbb{S}}(\cdot)$ denotes the indicator function, whose value is 1 when $\mathbf{r}' \in \mathbb{S}$ and 0 otherwise. $\mathbf{c}(\mathbf{r}) \in \mathbb{R}^{2|\mathbf{N}_\mathbf{r}|}$ represents the correlation map between the current fluid at $\mathbf{r}$ and its $|\mathbf{N}_\mathbf{r}|$ neighbors in the previous fluid, with additional consideration on the boundary conditions $\mathbb{S}$. Thus, we obtain the extracted dynamics information $\mathbf{c} \in \mathbb{R}^{2|\mathbf{N}_\mathbf{r}|\times H\times W}$. Subsequently, we can decode the potential and stream functions from the dynamics information and

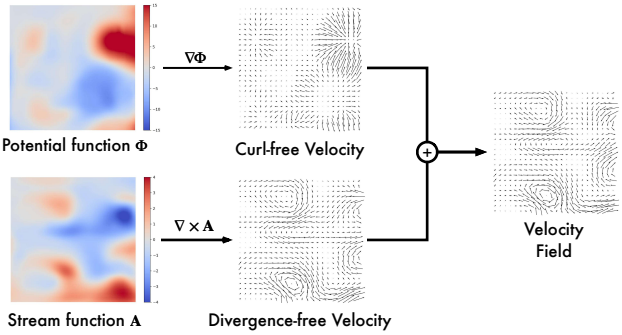


Figure 3. Transform potential and stream functions to velocity.

calculate the Helmholtz dynamics as follows:

$$\begin{aligned} &\widehat{\Phi} = \text{Decoder}_\Phi\left(\mathbf{c}\right),\ \widehat{\mathbf{A}} = \text{Decoder}_\mathbf{A}\left(\mathbf{c}\right), \\ &\widehat{\mathbf{F}}_{\text{Helm}} = \nabla\widehat{\Phi} + \nabla\times\widehat{\mathbf{A}}, \end{aligned} \tag{5}$$

where $\widehat{\Phi}, \widehat{\mathbf{A}} \in \mathbb{R}^{H\times W}$ and $\widehat{\mathbf{F}}_{\text{Helm}} \in \mathbb{R}^{2\times H\times W}$ represents the learned 2D fields of curl-free velocity, divergence-free velocity, and combined velocity respectively (Figure 3). Decoder_Φ and $\text{Decoder}_\mathbf{A}$ are learnable deep layers instantiated as two convolutional layers with an in-between activation function. We summarize the above process as

$$\widehat{\mathbf{F}}_{\text{Helm}} = \text{HelmDynamics}(\hat{\mathbf{x}}_{(T-1)}, \hat{\mathbf{x}}_T). \tag{6}$$

Note that we follow the standard practice in RAFT (Teed & Deng, 2020) and learn the fluid dynamics information from spatiotemporal correlations $\mathbf{c}$. However, rather than directly learning the velocity field from $\mathbf{c}(\mathbf{r})$ as in RAFT, we compose it from the learned potential and stream functions. Experimentally, this allows us to learn a more reasonable velocity field and enhances the accuracy of fluid prediction.

3.2. Multiscale Multihead Integral Architecture

After tackling intricate dynamics with the HelmDynamics blocks, we further present the Multiscale Multihead Integral Architecture to fuse learned dynamics along the temporal dimension for predicting future fluid, consisting of a multi-head integral block and a multiscale modeling framework.

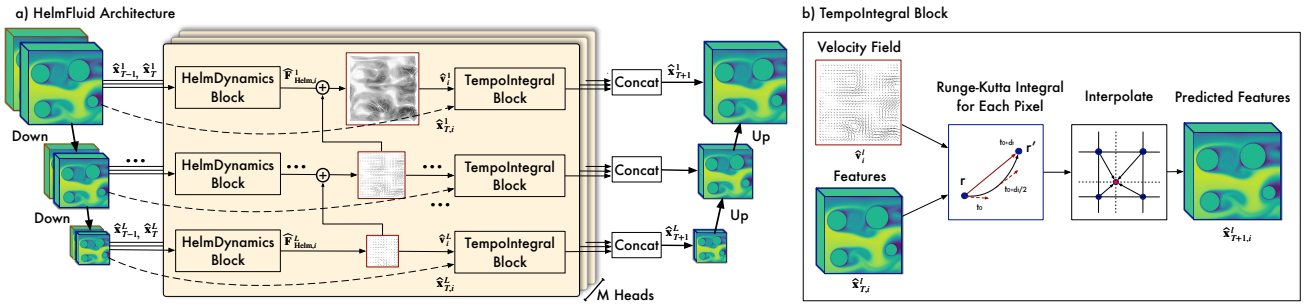


Figure 4. HelmFluid architecture (left part), which employs Runge-Kutta with BFECC (Kim et al., 2005) as a TempoIntegral Block to integrate the learned Helmholtz dynamics along the temporal dimension (right part) at multiple scales with multiheads to generate future fluid field. Especially, a residual connection across different scales is utilized to ensure the consistency of learned multiscale dynamics.

Multihead dynamics To predict the complex dynamics in fluid, we propose a multihead design for temporal integral, which is widely used in the attention mechanism to augment nonlinear capacity of deep models (Vaswani et al., 2017). The multihead design enables the model to capture different dynamic patterns via multiple Helmholtz dynamics $\widehat{\mathbf{F}}_{\text{Helm},i}$ learned within different heads. As shown in Figure 4, given the deep representations $\widehat{\mathbf{x}}_{(T-1)}, \widehat{\mathbf{x}}_T \in \mathbb{R}^{d_{\text{model}} \times H \times W}$ of two successive frames of fluid, we first split them into multiple heads along the channel dimension, with each head i as $\widehat{\mathbf{x}}_{(T-1),i}, \widehat{\mathbf{x}}_{T,i} \in \mathbb{R}^{\frac{d_{\text{model}}}{M} \times H \times W}, i \in \{1, \cdots, M\}$, where M is the number of heads. Then we compute the Helmholtz dynamics from the deep representations for each head i:

$$\widehat{\mathbf{F}}_{\text{Helm},i} = \text{HelmDynamics}(\widehat{\mathbf{x}}_{(T-1),i}, \widehat{\mathbf{x}}_{T,i}), \tag{7}$$

where $\widehat{\mathbf{F}}_{\text{Helm},i} \in \mathbb{R}^{2 \times H \times W}, i = 1, \cdots, M$.

Multiscale modeling It is known in physics that the fluid exhibits different properties at different scales. These multiscale dynamics entangle with each other, making the fluid extremely intractable. Thus, we adopt a multiscale modeling framework to enhance dynamics modeling.

Given input embeddings $\widehat{\mathbf{x}}_{(T-1)}, \widehat{\mathbf{x}}_T \in \mathbb{R}^{d_{\text{model}} \times H \times W}$, we adopt a multiscale encoder to obtain deep representations in L scales: $\widehat{\mathbf{x}}^l_{(T-1)}, \widehat{\mathbf{x}}^l_T \in \mathbb{R}^{d^l_{\text{model}} \times \lfloor \frac{H}{2^{(l-1)}} \rfloor \times \lfloor \frac{W}{2^{(l-1)}} \rfloor}, l \in \{1, \cdots, L\}$. As the dynamics at larger scales are less affected by noise and more capable of giving a reliable background velocity field for the smaller scales, we ensemble the learned dynamics from coarse to fine to ease the multiscale dynamics modeling process. As shown in Figure 4, we obtain the velocity field $\widehat{\mathbf{v}}^l_i$ at the l-th scale by

$$\widehat{\mathbf{v}}^l_i = \begin{cases} \widehat{\mathbf{F}}^l_{\text{Helm},i}, & l = L \\ \widehat{\mathbf{F}}^l_{\text{Helm},i} + \text{Upsample}(\widehat{\mathbf{v}}^{l+1}_i), & 1 \leq l < L \end{cases} \tag{8}$$

where $\widehat{\mathbf{v}}^l_i \in \mathbb{R}^{2 \times H \times W}$, and $\text{Upsample}(\cdot)$ is the bilinear interpolation to keep resolution compatible. We incorporate the idea of residual learning for multiscale dynamics to align the velocity values. This ensures the consistency of velocity field at the fine scale with that at the coarse scale.

TempoIntegral block To predict the future fluid field, we integrate the feature space along the temporal dimension. Concretely, for scale $l \in 1, 2, ..., L$ and head $i \in 1, 2, ..., M$, we integrate the deep representation $\widehat{\mathbf{x}}^l_{T,i}$ by its corresponding velocity field $\widehat{\mathbf{v}}^l_i$. As shown in Figure 4, for a position $\mathbf{r}$, we take the second-order Runge-Kutta method (DeVries & Wolf, 1994) as the numerical integral method to estimate its position in the future $\mathrm{d}t$ time: $\mathbf{r}^{l\prime}_i = \mathbf{r} + \widehat{\mathbf{v}}^l_i(\mathbf{r} + \widehat{\mathbf{v}}^l_i(\mathbf{r})\frac{\mathrm{d}t}{2})\mathrm{d}t$. This equation can directly deduce the next step representation by moving the pixel at $\mathbf{r}$ to $\mathbf{r}'$. Following the convention in temporal integral, we adopt the back-and-forth error compensation and correction (BFECC, (Kim et al., 2005)) for better position mapping, which enhances the Runge-Kutta with an ensemble of bidirectional integral. Since the mapped coordinates $\mathbf{r}^{l\prime}_i$ may not be integer positions on regular grids, we further use bilinear interpolation to yield representations on regular grids. We summarize the whole temporal integral process for each head i at each scale l as

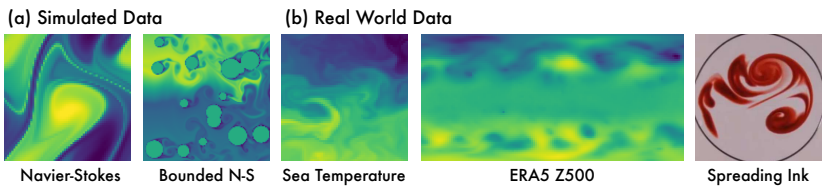
$$\begin{aligned} \widehat{\mathbf{x}}^l_{(T+1),i} &= \text{Interpolate}\left(\text{BFECC}\left(\widehat{\mathbf{x}}^l_{T,i}, \widehat{\mathbf{v}}^l_i\right)\right) \\ \widehat{\mathbf{x}}^l_{(T+1)} &= \text{Concat}\left(\left[\widehat{\mathbf{x}}^l_{(T+1),i}\right]_{i=1,\cdots,M}\right). \end{aligned} \tag{9}$$

Eventually, we progressively aggregate the predicted context features from large to small scales and obtain the final prediction of the fluid field with a projection layer. More details of the model architecture including implementation of the BFECC method are deferred to Appendix B.2.

4. Experiments

We extensively evaluate HelmFluid on five benchmarks, including both simulated and real-world observed scenarios, covering known and unknown boundary settings (see Figure 5). Extensions to 3D fluid are included in Appendix A. Descriptions of datasets, baselines, and implementation details are listed in Appendix B.

Baselines We compare HelmFluid with nine competitive baselines, including one numerical method DARTS (Ruzanski et al., 2011), four neural fluid simulators: LSM (Wu

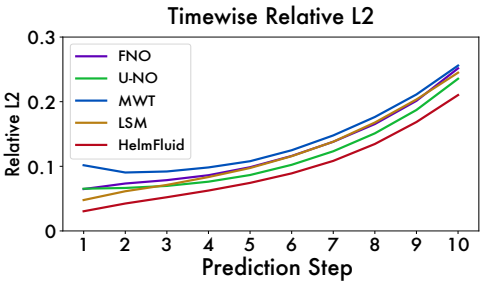


BENCHMARK	BOUNDARY	LENGTH
NAVIER-STOKES	UNKNOWN	10
BOUNDED N-S	KNOWN	10
ERA5 Z500	KNOWN	10
SEA TEMPERATURE	UNKNOWN	10
SPREADING INK	KNOWN	46-63

Figure 5. Summary of five experiment benchmarks, including (a) simulated and (b) real-world data.

Table 1. Performance comparison on the Navier-Stokes dataset under different resolutions. Relative L2 is recorded. For clarity, the best result is in bold and the second best is underlined. The relative promotion is calculated between the best and second-best models, that is $1 - \frac{\text{The best error}}{\text{The second best error}}$. The timewise error curve is recorded from the 64×64 settings.

MODEL	64×64	128×128	256×256
DARTS (RUZANSKI ET AL., 2011)	0.8046	0.7002	0.7904
U-NET (RONNEBERGER ET AL., 2015)	0.1982	0.1589	0.2953
FNO (LI ET AL., 2021)	0.1556	0.1028	0.1645
MWT (GUPTA ET AL., 2021)	0.1586	<u>0.0841</u>	<u>0.1390</u>
U-NO (RAHMAN ET AL., 2023)	<u>0.1435</u>	0.0913	0.1392
LSM (WU ET AL., 2023)	0.1535	0.0961	0.1973
HELMFLUID (OURS)	**0.1261**	**0.0807**	**0.1310**
PROMOTION	12.1%	4.0%	5.8%



et al., 2023), U-NO (Rahman et al., 2023), WMT (Gupta et al., 2021), FNO (Li et al., 2021), two fluid-dynamics-modeling solutions: Vortex (Deng et al., 2023), PWC-Net with fluid Refinement (Zhang et al., 2022), one vision backbone widely-used in fluid modeling: U-Net (Ronneberger et al., 2015), and one model specialized for weather forecasting: FourcastNet (Pathak et al., 2022). Here, LSM and U-NO are previous state-of-the-art models in fluid prediction. Note that due to the inconsistent settings in fluid prediction, some of the baselines are not suitable for all benchmarks. Thus, in the main text, we only provide comparisons to baselines on their official benchmarks. But to ensure transparency, we also provide the complete results for other baselines in Table 21.

4.1. Simulated Data

Navier-Stokes with unknown boundary This dataset is simulated from a viscous, incompressible fluid field on a two-dimensional unit torus, which obeys Navier-Stokes equations (Li et al., 2021). The task is to predict the future 10 steps based on the past 10 observations.

As presented in Table 1, HelmFluid significantly surpasses other models, demonstrating its advancement in fluid prediction. In comparison with the second-best model, HelmFluid achieves 12.1% relative error reduction (0.1261 vs. 0.1435) in the 64×64 resolution setting and achieves consistent state-of-the-art in all time steps. Besides, HelmFluid performs best for the inputs under various resolutions, verifying its capability to handle the dynamics at different scales.

To intuitively present the model capability, we also provide several showcases in Figure 6. In comparing to U-NO and LSM, HelmFluid precisely predicts the fluid motion, especially the twist parts, which involve complex interactions among several groups of fluid particles. Besides, HelmFluid also generates the learned velocity field for each step, which reflects the rotation and diffusion of fluid, empowering prediction with interpretable evidence. These results demonstrate the advantages of HelmFluid in capturing complex dynamics and endowing model interpretability.

Table 2. Model performance comparison on Bounded N-S dataset.

MODEL	RELATIVE L2
DARTS (RUZANSKI ET AL., 2011)	0.1820
U-NET (RONNEBERGER ET AL., 2015)	0.0846
FNO (LI ET AL., 2021)	0.1176
MWT (GUPTA ET AL., 2021)	0.1407
U-NO (RAHMAN ET AL., 2023)	0.1200
LSM (WU ET AL., 2023)	<u>0.0737</u>
HELMFLUID (OURS)	**0.0652**
PROMOTION	11.5%

Bounded N-S with known boundary This dataset simulates a wide pipe scenario, where the incompressible fluid moves from left to right, passing by several solid columns with position and size fixed in the dataset. The goal is to predict the future 10 steps based on past 10 observations.

HelmFluid also performs best in this challenging task and presents a consistent advantage in all prediction steps. While U-Net seems to be close to HelmFluid in averaged relative L2, it fails to capture the Karmen vortex phenomenon and results in blurry predictions (Figure 7), which will seriously

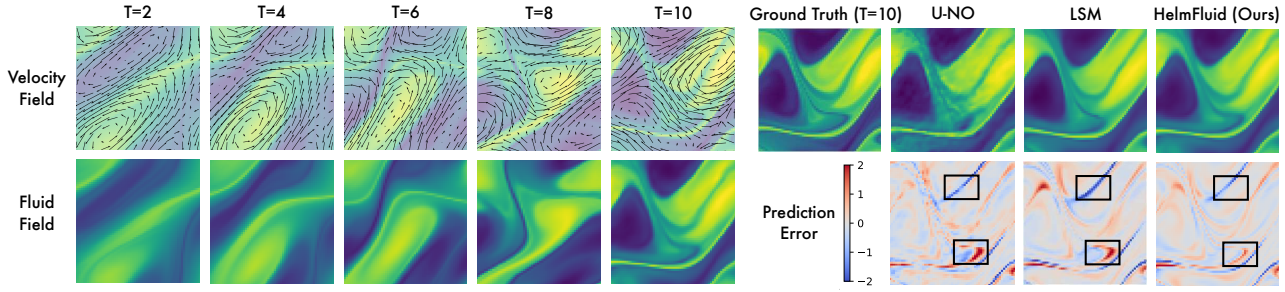


Figure 6. Showcase comparison and learned Helmholtz dynamics on the Navier-Stokes dataset under the 64×64 input resolution.

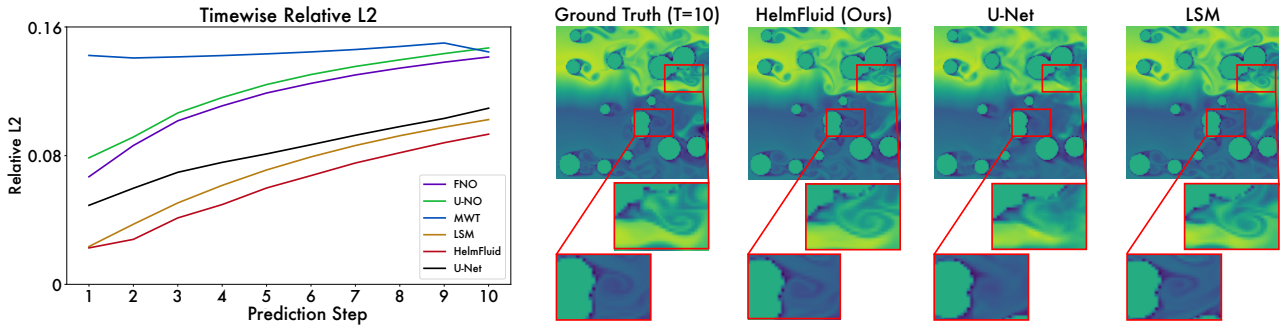


Figure 7. Timewise error and showcases on Bounded N-S dataset. For clarity, we highlight and zoom in the key parts of fluid in red boxes.

Table 3. Model comparison on the ERA5 Z500 dataset.

MODELS	RMSE
U-NET (RONNEBERGER ET AL., 2015)	632.94
FNO (LI ET AL., 2021)	596.80
MWT (GUPTA ET AL., 2021)	596.45
U-NO (RAHMAN ET AL., 2023)	596.84
LSM (WU ET AL., 2023)	<u>561.27</u>
FOURCASTNET (PATHAK ET AL., 2022)	594.49
HELMFLUID (OURS)	**521.44**
PROMOTION	7.1%

Table 4. Model comparison on the Sea Temperature dataset.

MODELS	RELATIVE L2	MSE
DARTS (RUZANSKI ET AL., 2011)	0.3308	0.1094
U-NET (RONNEBERGER ET AL., 2015)	<u>0.1735</u>	<u>0.0379</u>
FNO (LI ET AL., 2021)	0.1935	0.0456
MWT (GUPTA ET AL., 2021)	0.2075	0.0506
U-NO (RAHMAN ET AL., 2023)	0.1969	0.0472
LSM (WU ET AL., 2023)	0.1759	0.0389
HELMFLUID (OURS)	**0.1704**	**0.0368**
PROMOTION	1.8%	2.9%

impede its interpretability. In contrast, HelmFluid precisely predicts the Karmen vortex around boundaries with eidetic texture. This result benefits from the learning paradigm designed based on Helmholtz dynamics.

Besides, we provide the comparison of learned dynamics among HelmFluid, DARTS (2011) and PWC-Net (2022) in Figure 1. HelmFluid shows impressive capturing of the dynamics accurately, even for vortices around solid columns. This capability stems from HelmFluid's design in learning potential and stream functions instead of directly learning velocities, thereby mitigating overwhelming the model by intricate dynamics. It is notable that the numerical method DARTS degenerates seriously in both quantitative results (Table 2) and learned dynamics (Figure 1), which highlights challenges in this task and the advantage of HelmFluid.

4.2. Real-world Data

ERA5 Z500 with known boundary This dataset is processed from the fifth generation of ECMWF reanalysis (ERA5) data (Hersbach et al., 2020). Following the practice of weather forecasting, we choose the geopotential height at 500 hPa (Z500) with a resolution of 2.5° and a time interval of 3 hours as our data. The goal is to predict the geopotential height for 10 timesteps given 2 observations.

The Root Mean Square Error (RMSE) for Z500 is detailed in Table 3, revealing that HelmFluid consistently outperforms all other comparative baselines. Notably, it surpasses FourcastNet (2022), the pioneering model to leverage ERA5 reanalysis data for medium-range meteorological forecasts. This superior performance indicates that HelmFluid is adept at predicting real-world atmospheric fluid dynamics and has the potential to advance weather forecasting capabilities. More showcases can be found in Figure 22.

Sea Temperature with unknown boundary This dataset consists of the reanalysis ocean temperature data (MDS) provided by ECMWF. We adapted the data in four 64×64 regions located in different oceans. The goal is to predict

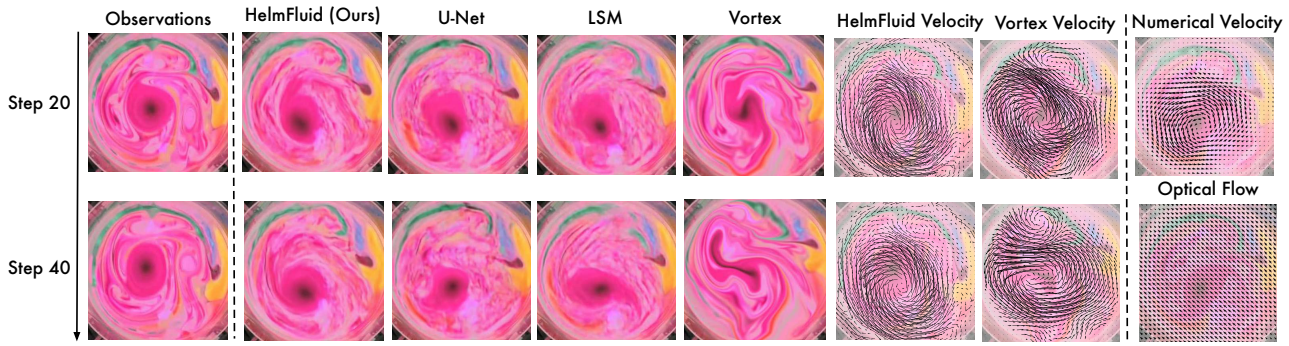


Figure 8. Showcases of prediction results (future 20 and 40 steps) and learned velocity fields (future 40 steps) on the Spreading Ink dataset. We also show velocity fields of the first timestep estimated by numerical method (DARTS) and deep optical flow (PWC-Net).

Table 5. Model comparison on Spreading Ink. Averaged Perceptual loss, Relative L2 and MSE of three sub-videos are reported.

MODELS	METRICS
U-NET (RONNEBERGER ET AL., 2015)	3.596 / 0.2620 / 0.0176
FNO (LI ET AL., 2021)	4.095 / 0.2776 / 0.0198
U-NO (RAHMAN ET AL., 2023)	5.604 / 0.2971 / 0.0227
VORTEX (DENG ET AL., 2023)	3.949 / 0.2483 / 0.0161
LSM (WU ET AL., 2023)	3.760 / 0.2698 / 0.0187
HELMFLUID (OURS)	**3.323 / 0.2183 / 0.0125**
PROMOTION	7.6% / 12.1% / 22.3%

Table 6. Ablation on the HelmDynamics block, which includes learning w/o HelmDynamics on the 64×64 Navier-Stokes dataset, and learning w/o boundary conditions on Bounded N-S dataset.

DATA	MODEL	RELATIVE L2
NAVIER-STOKES	DIRECTLY LEARNING VELOCITY	0.1412
	LEARNING HELMDYNAMICS	**0.1261**
	PROMOTION	10.7%
BOUNDED N-S	W/O BOUNDARY CONDITIONS	0.0846
	W/ BOUNDARY CONDITIONS	**0.0652**
	PROMOTION	22.9%

the temperature for the next 10 days based on 10 past days.

The results in Table 4 demonstrate that HelmFluid can handle real-world data well and outperform all baselines. It is worth noting that the test set is collected from different regions with respect to the training and validation sets, which involves the distribution shift problem. Thus, these results also verify the generality and transferability of HelmFluid.

Spreading Ink with known boundary This benchmark consists of three videos collected by Deng et al., involving more than 100 successive frames respectively. Following the experiment setting in Vortex (2023), the goal is to predict the last 1/3 frames of the video using the first 2/3.

The quantitative results are listed in Table 5. HelmFluid still performs well in this long-term forecasting task. In addition to the relative L2 and MSE, it also consistently achieves the lowest VGG perceptual loss, implying that the prediction results of HelmFluid can maintain the realistic texture and intuitive physics. As for showcases in Figure 8, we find that HelmFluid can precisely capture the diffusion of ink. Even for the future 40 frames, HelmFluid still performs well in capturing the hollow position and surpasses numerical methods, optical flow and Vortex, in learning the velocity.

4.3. Model analysis

Efficiency analysis To evaluate model practicability, we also provide efficiency analysis in Figure 9. In comparison with the second-best model U-NO, HelmFluid presents a favorable trade-off between efficiency and performance. In particular, HelmFluid surpasses U-NO by 12.1% in relative L2 with comparable running time. See Appendix D for full results and comparisons under aligned model size.

Ablations To highlight advantages of learning Helmholtz dynamics, we compare HelmFluid with its two variants: directly learning the velocity field and removing the boundary condition design. As shown in Table 6, directly estimating the velocity field will cause 10.7% performance drop. A plausible reason is that deep models can be overwhelmed by complex fluid interactions, and thus learning Helmholtz dynamics is beneficial. Also, our design in incorporating boundary condition is essential. Empowered by the HelmDynamics block, our model can conveniently utilize the boundary information, unleashing its potential in handling fluid with complex boundaries. Complete quantitative and

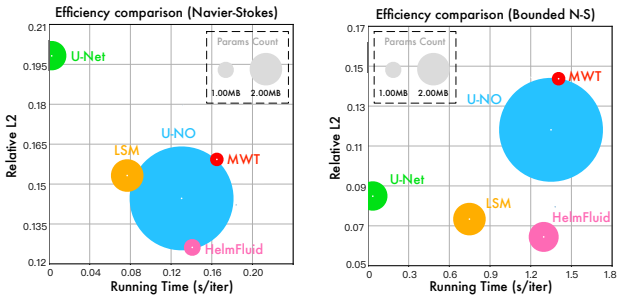


Figure 9. Efficiency comparison evaluated on the 64×64 Naiver-Stokes and 128×128 Bounded N-S averaged from 10^3 iterations.

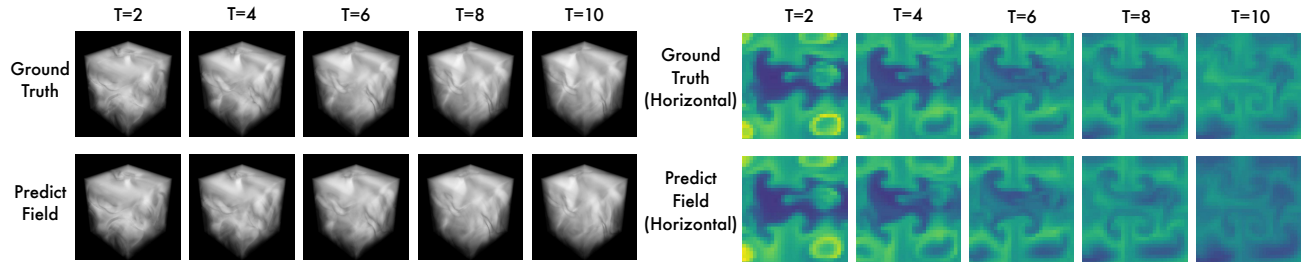


Figure 10. Showcases of the 3D Smoke Dataset, visualization from 3D perspective and 2D horizontal slice are both provided.

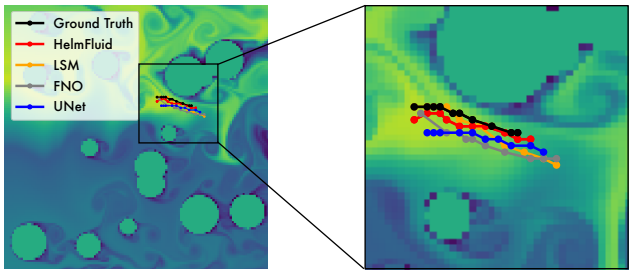


Figure 11. Tracking of the local maxima of Bounded N-S.

visual comparisons are included in Appendices C and E.

Dynamics tracking This paper is based on the Eulerian specification of fluid. As a supplement, we also provide a Lagrangian perspective to analyze model predictions. Technically, we locate the local maxima of fluid and keep tracking it in the subsequent frames. The closer the predicted trajectory of the point is to its real counterpart, the better tracking of the point is indicated. As shown in Figure 11, the trajectory predicted by HelmFluid is the closest to the ground truth, verifying the capability of HelmFluid in capturing the intricate dynamics of fluid.

Generalization on boundary conditions HelmFluid incorporates boundary conditions as additional data for correlation calculations, enhancing its predictive capabilities. To evaluate the model's performance under various boundary conditions, we undertook a modification of the Bounded Navier-Stokes dataset by altering the geometric form of the solid columns from *circular* to *square*, thereby introducing a distinct set of boundary conditions for testing. We tested on three different training-testing scenarios, including zero-shot testing on the alternative dataset, fine-tuning the model parameters on the alternative dataset followed by testing, and blending two distinct datasets for separate testing. All the models are trained with the same number of epochs.

As shown in Table 7, direct zero-shot testing on a new dataset leads to a significant degradation in model performance. However, merely fine-tuning for 3 epochs achieved comparable or even superior test results on the original dataset. In the scenario of mixed training, test metrics on both datasets improve by 2.9% and 16.7%, respectively.

Table 7. Experimental results for generalized boundary conditions, *circle* denotes the original Bounded N-S dataset, while *square* denotes the modified dataset. The underlined metric indicates training performance with the test set being identical.

Dataset	Circle	Square
Trained on Circle	<u>0.0731</u>	0.1501
Fintuned 3 epochs on Circle	-	0.0791
Fintuned 100 epochs on Circle	-	0.0781
Trained on Square	0.1503	<u>0.0855</u>
Fintuned 3 epochs on Square	0.0710	-
Fintuned 100 epochs on Square	0.0691	-
Trained on Circle and Square	**0.0618**	**0.0712**

This proves the generalizability of HelmFluid in handling various boundary conditions and demonstrates an improvement as the dataset grows in size and diversity.

4.4. Extend HelmFluid to 3D Fluid

The form of Helmholtz decomposition implies its applicability across arbitrarily high dimensions. To address fluid prediction in three-dimensional scenarios, we extend the HelmFluid model to 3D and experiment on the simulated 3D smoke buoyancy dataset with known boundary. As shown in Figure 10, HelmFluid generates smoke that closely matches the shape and position of the ground truth, effectively reflecting the variations in location and intensity of the smoke. See Appendix A for details of the dataset and implementation.

5. Conclusions and Future Work

In this paper, we present the HelmFluid model towards accurate and interpretable fluid prediction. Instead of directly learning velocity fields, we propose to learn the Helmholtz dynamics, which casts the intricate dynamics of fluid into inherent physics quantities. With HelmDynamics blocks and Multiscale Multiscale Integral Architecture, HelmFluid can precisely estimate the potential and stream functions for Helmholtz dynamics, which empowers the prediction process with physical interpretability. HelmFluid achieves consistent state-of-the-art on both simulated and real-world datasets, even for scenarios with complex boundaries. In the future, we plan to further extend HelmFluid to large-scale datasets, such as world climate and ocean current modeling.

Acknowledgements

This work was supported by the National Key Research and Development Plan (2021YFC3000905), the National Natural Science Foundation of China (U2342217 and 62022050), the BNRist Innovation Fund (BNR2024RC01010), and the National Engineering Research Center for Big Data Software.

Impact Statement

This paper presents work whose goal is to advance deep learning research for fluid prediction, which has the potential application to enhance the interpretability of atmospheric forecasts, ocean forecasts, and turbulence predictions in machine learning. Our work only focuses on the scientific problem, so there is no potential ethical risk.

References

Ando, R., Thuerey, N., and Wojtan, C. A stream function solver for liquid simulations. *ACM Trans. Graphics*, 2015.

Baukal Jr, C. E., Gershtein, V., and Li, X. J. *Computational fluid dynamics in industrial combustion*. CRC press, 2000.

Bayındır, C. and Namlı, B. Efficient sensing of von kármán vortices using compressive sensing. *Comput. Fluids*, 2021.

Bhatia, H., Norgard, G., Pascucci, V., and Bremer, P.-T. The helmholtz-hodge decomposition—a survey. *IEEE Trans. Vis. Comput. Graph.*, 2013.

Bladel, J. On helmholtz's theorem in finite regions. *IRE Trans. Anntenas Propag.*, 1959.

Deng, Y., Yu, H.-X., Wu, J., and Zhu, B. Learning vortex dynamics for fluid inference and prediction. In *ICLR*, 2023.

DeVries, P. L. and Wolf, R. P. A first course in computational physics. *Comput. Phys.*, 1994.

Evans, L. C. *Partial differential equations*. American Mathematical Soc., 2010.

Faith A., M. *An introduction to fluid mechanics*. Cambridge University Press, 2013.

Franz, E., Solenthaler, B., and Thuerey, N. Learning to estimate single-view volumetric flow motions without 3d supervision. In *ICLR*, 2023.

Gupta, G., Xiao, X., and Bogdan, P. Multiwavelet-based operator learning for differential equations. In *NeurIPS*, 2021.

Hersbach, H., Bell, B., Berrisford, P., Hirahara, S., Horányi, A., Muñoz-Sabater, J., Nicolas, J., Peubey, C., Radu, R., Schepers, D., et al. The era5 global reanalysis. *Quarterly Journal of the Royal Meteorological Society*, 146(730): 1999–2049, 2020.

Horn, B. K. and Schunck, B. G. Determining optical flow. *AI*, 1981.

Hu, Y., Li, T.-M., Anderson, L., Ragan-Kelley, J., and Durand, F. Taichi: a language for high-performance computation on spatially sparse data structures. *ACM Trans. Graphics*, 2019.

Johnson, J., Alahi, A., and Fei-Fei, L. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016.

Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., and Yang, L. Physics-informed machine learning. *Nat. Rev. Phys.*, 2021.

Kim, B., Liu, Y., Llamas, I., and Rossignac, J. Flowfixer: Using bfecc for fluid simulation. In *NPH*, 2005.

Kim, B., Azevedo, V. C., Thuerey, N., Kim, T., Gross, M., and Solenthaler, B. Deep fluids: A generative network for parameterized fluid simulations. *Comput. Graph. Forum*, 2019.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Li, Z., Kovachki, N. B., Azizzadenesheli, K., liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Fourier neural operator for parametric partial differential equations. In *ICLR*, 2021.

Li, Z., Shu, D., and Farimani, A. B. Scalable transformer for PDE surrogate modeling. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=djyn8Q0anK.

Liu, P., Tian, L., Zhang, Y., Aylward, S., Lee, Y., and Niethammer, M. Discovering hidden physics behind transport dynamics. In *CVPR*, 2021.

Lu, L., Jin, P., Pang, G., Zhang, Z., and Karniadakis, G. E. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nat. Mach. Intell.*, 2021a.

Lu, L., Meng, X., Mao, Z., and Karniadakis, G. E. Deepxde: A deep learning library for solving differential equations. *SIAM Rev.*, 2021b.

(MDS), E. C. M. S. I. C. M. D. S. Global ocean physics reanalysis. DOI: 10.48670/moi-00021 (Accessed on 23 September 2023), 2023.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

Pathak, J., Subramanian, S., Harrington, P., Raja, S., Chattopadhyay, A., Mardani, M., Kurth, T., Hall, D., Li, Z., Azizzadenesheli, K., et al. Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators. *arXiv preprint arXiv:2202.11214*, 2022.

Rahman, M. A., Ross, Z. E., and Azizzadenesheli, K. U-NO: U-shaped neural operators. *TMLR*, 2023.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.*, 2019.

Raissi, M., Yazdani, A., and Karniadakis, G. E. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science*, 2020.

Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015.

Ruzanski, E., Chandrasekar, V., and Wang, Y. The casa nowcasting system. *J. Atmos. Oceanic Technol.*, 2011.

Sun, D., Yang, X., Liu, M.-Y., and Kautz, J. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *CVPR*, 2018.

Teed, Z. and Deng, J. Raft: Recurrent all-pairs field transforms for optical flow. In *ECCV*, 2020.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. Attention is all you need. In *NeurIPS*, 2017.

Wang, H., Fu, T., Du, Y., Gao, W., Huang, K., Liu, Z., Chandak, P., Liu, S., Van Katwyk, P., Deac, A., et al. Scientific discovery in the age of artificial intelligence. *Nature*, 2023.

Wang, R., Kashinath, K., Mustafa, M., Albert, A., and Yu, R. Towards physics-informed deep learning for turbulent flow prediction. *KDD*, 2019.

Wu, H., Hu, T., Luo, H., Wang, J., and Long, M. Solving high-dimensional pdes with latent spectral models. In *ICML*, 2023.

Zhang, M., Wang, J., Tlhomole, J. B., and Piggott, M. Learning to estimate and refine fluid motion with physical dynamics. In *ICML*, 2022.

A. Extend HelmFluid to 3D Fluid

Here we present the potential extension of HelmFluid to 3D fluid prediction. According to the formalization of Helmholtz decomposition $\mathbf{F}(\mathbf{r}) = \nabla\Phi(\mathbf{r}) + \nabla \times \mathbf{A}(\mathbf{r}), \mathbf{r} \in \mathbb{V}$. For 2D cases in the main text, the velocity component on the z-axis is set to be zero, thereby $\mathbf{A}_x(\mathbf{r}) = \mathbf{A}_y(\mathbf{r}) = 0$. By extending the HelmDynamics block to learn $\widehat{\Phi} \in \mathbb{R}^{1\times D\times H\times W}$ and $\widehat{\mathbf{A}} \in \mathbb{R}^{3\times D\times H\times W}$, where D is the additional depth dimension of 3D fluid, we can adapt HelmFluid to 3D fluid prediction. Then, following the Helmholtz decomposition presented in Eq. 1, we can easily obtain the inferred 3D vector velocity field, thereby enabling HelmFluid to achieve the velocity-aware 3D fluid prediction.

To validate the capabilities of HelmFluid on 3D fluid prediction scheme, we generated 3D smoke buoyancy dataset by modifying the 3D solver (under MIT license) from https://github.com/BaratiLab/FactFormer/. 3D smoke buoyancy problem is governed by the incompressible Navier-Stokes equation coupled with advection equation (Li et al., 2023):

$$
\begin{aligned}
&\frac{\partial \mathbf{u}(\mathbf{x},t)}{\partial t} + \mathbf{u}(\mathbf{x},t)\cdot\nabla\mathbf{u}(\mathbf{x},t) = \nu\nabla^2\mathbf{u}(\mathbf{x},t) - \frac{1}{\rho}\nabla p(\mathbf{x},t) + \mathbf{f}(\mathbf{x},t), && \mathbf{x}\in(0,L)^3, t\in(0,T],\\
&\frac{\partial d(\mathbf{x},t)}{\partial t} + \mathbf{u}(\mathbf{x},t)\cdot\nabla d(\mathbf{x},t) = 0, && \mathbf{x}\in(0,L)^3, t\in(0,T],\\
&\nabla\cdot\mathbf{u}(\mathbf{x},t) = 0, && \mathbf{x}\in(0,L)^3, t\in[0,T],\\
&\mathbf{u}(\mathbf{x},0) = 0, \quad \mathbf{d}(\mathbf{x},0) = d_0(\mathbf{x}), \quad \mathbf{f}(\mathbf{x},t) = [0,0,\eta d(\mathbf{x},t)] && \mathbf{x}\in(0,L)^3, t\in(0,T],
\end{aligned}
\tag{10}
$$

where L is set 32, η is the buoyancy factor. The goal is to predict the future 10 steps based on the past 10 frames. We generated 1000 trajectories for training and 200 for testing and provided 3D showcases and 2D slices in Figure 12.

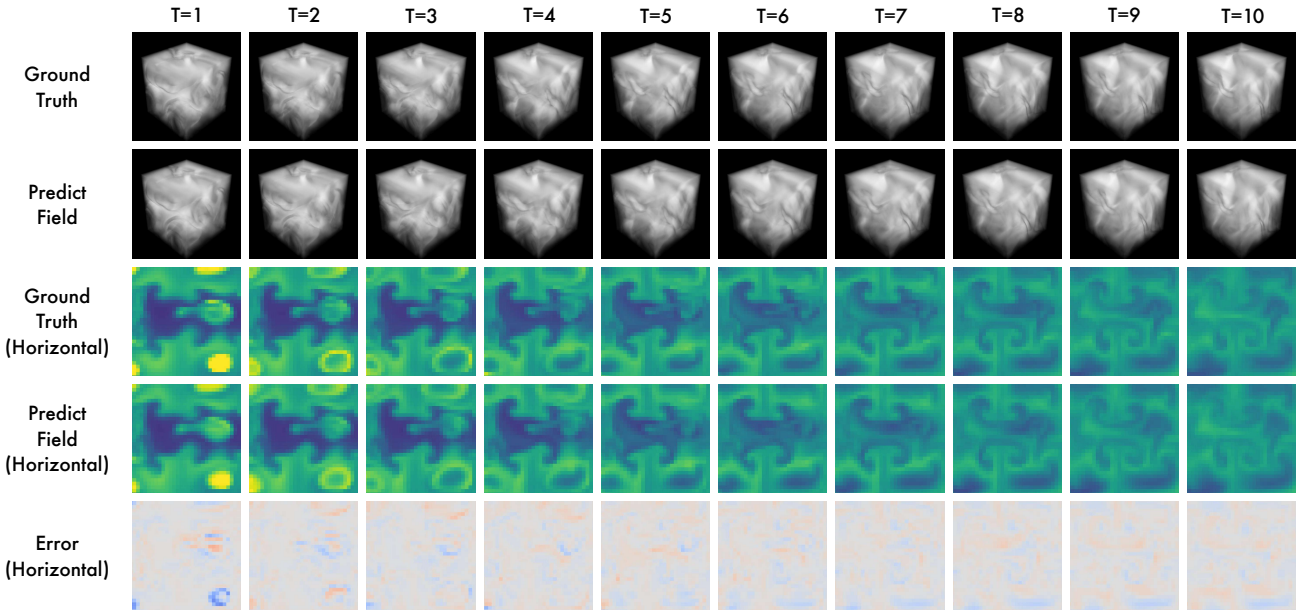


Figure 12. Showcases of the 3D Dataset, visualization from 3D perspective and 2D horizontal slice are provided.

B. Implementation Details

B.1. Dataset

We summarize the experiment datasets in Table 8. More details can be found in the following.

Table 8. A summary of experiment datasets. Note that the Spreading Ink dataset is different from other benchmarks, which only contains three video sequences. We strictly follow the Vortex (Deng et al., 2023) to split the video for training, validation and test. For example, in the training phase of video 1, we use the first 70 frames for training and the subsequent 30 frames for validation. As for the test, we use the first 100 frames as input and predict the following 50 frames.

DATASET	(INPUT, PREDICT LENGTH)	(TRAINING, VALIDATION, TEST)	OBSERVED STATE	REYNOLD NUMBERS
NAVIER-STOKES	(10,10)	(1000,200,200)	VORTICITY	$\sim 10^4$
BOUNDED N-S	(10,10)	(1000,200,200)	GRAYSCALE	~ 300
ERA5 Z500	(2,10)	(20425, 2087, 4174)	GEOPOTENTIAL	UNKNOWN
SEA TEMPERATURE	(10,10)	(170249, 17758, 65286)	TEMPERATURE	UNKNOWN
SPREADING INK VIDEO 1	(100, 50)	ONE VIDEO SEQUENCE	RGB IMAGE	UNKNOWN
SPREADING INK VIDEO 2	(126, 63)	ONE VIDEO SEQUENCE	RGB IMAGE	UNKNOWN
SPREADING INK VIDEO 3	(93, 46)	ONE VIDEO SEQUENCE	RGB IMAGE	UNKNOWN

Navier-Stokes Navier-Stokes equations describe the motion of a viscous incompressible field. In this paper, we follow (Li et al., 2021) and generate fluid on a 2D torus with the following equation:

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} - \nu \nabla^2 \mathbf{u} &= -\frac{1}{\rho}\nabla p + \mathbf{g} \\ \nabla \cdot \mathbf{u} &= 0 \\ \nabla \times \mathbf{u}(x, 0) &= \omega_0(x), x \in (0, 1)^2, \end{aligned} \tag{11}$$

where $\mathbf{u} \in \mathbb{R}^2$ represents the velocity field, p denotes the pressure, and ρ is the fluid density, which we assumed to be constant in the incompressible fluid field. $\nu \in \mathbb{R}_+$ is the kinematic viscosity representing the intrinsic nature of the fluid, which is assumed to be constant. $\mathbf{g} \in \mathbb{R}^2$ represents the summation of all the external forces applied on the fluid field. Vorticity is calculated by the velocity field, $\omega = \nabla \times \mathbf{u}$. At time zero, the initial vorticity field ω_0 is given. The goal is to predict the following vorticity fields from given observations.

We randomly sampled the initial vorticity w_0 on a two-dimensional unit torus from a Gaussian distribution and solved the equation with a numerical method to obtain the future velocity field. After generating the fluid field with 256×256 spatial resolution and 10^{-4} second temporal resolution, we downsampled it to a sequence of 1 second per frame and corresponding spatial resolution. Thus, each sequence consists of 20 frames with a total duration of 20 seconds. We fixed the viscosity $\nu = 10^{-5}$ for all three sub-datasets of different resolutions. To verify the model capacity in different resolutions, we generate three subsets ranging from 64×64 to 256×256 with 1000 training sequences, 200 validation sequences and 200 test sequences. The goal is to predict the future 10 steps based on the past 10 observations.

Bounded N-S In real-world applications, we usually need to handle the complex boundary conditions in fluid prediction. Specifically, suppose a 512×512 sized two-dimensional space with top and bottom as boundaries, with free space outside the image. We let a randomly colored fluid flow from left to right. To test the model performance under scenarios with complex boundaries, we randomly sampled fifteen circles of different sizes as obstacles and uniformly placed them in the 512×512 space. Then we used Taichi (Hu et al., 2019) as a simulator engine to generate fluid within top-down boundaries with a numerical fluid solver for the advection equation (Baukal Jr et al., 2000), and generated a sufficiently long sequence with one initial source condition. The generated fluid field contains the Karmen vortex phenomenon (Bayındır & Namlı, 2021) with many vortices of various sizes, making this problem extremely challenging.

Towards the flow field dataset, after the colored fluid field spreads over the space from left to right, we sample frames in the frequency of 60 steps and add the sampled frames into the dataset. To remove the noise from chromatic aberration, we transform all the samples into grayscale. Then we downsample the image to 128×128 and split the long sequence into disjoint subsequences with 20 timesteps. After randomly dividing them into train, validation, and test sets, we finally obtained the training, validation, and test set, which contains 1000, 200, and 200 sequences, respectively. The goal is to predict the future 10 steps based on the past 10 observations.

ERA5 Z500 We downsampled the geopotential height at 500 hPa (Z500) from the ERA5 global reanalysis data to a resolution of $2.5°$, resulting in a grid size of 72×144.For our experiments, we utilized data from 2013 to 2019 for training, 2020 as validation, and 2021 and 2022 as test sets, obtaining more than 20,000 sequences. To mitigate the influence of geographic location bias on the Z500 predictions, we implemented a normalization technique by subtracting the Z500 values

at each location from the corresponding average from 2013 to 2019. The task is to forecast the subsequent 10 frames of data, predicated on the preceding two observations. This equates to a predictive horizon of 30 hours into the future.

Sea Temperature We downloaded 20 years of daily mean sea water potential temperature on the sea surface from the reanalysis ocean data (MDS) provided by ECMWF. For experiments, we use cropped 64×64 temperature data in Atlantic, Indian, and South Pacific for training and validation, to be more exact, from 2000 to 2018 for training with 170,249 sequences and from 2019 to 2020 for validation with 17,758 sequences. Additionally, we use the sea temperature in the North Pacific from 2000 to 2020 for testing, including 65,286 sequences. For each 64×64 cropped area, we normalize it in spatial and temporal dimensions to ensure the observations are in a standard distribution, which can make the task free from the noises of sudden change and observation errors, and mainly focus on the dynamics modeling. Since there exists the region shift between training and test sets, this benchmark not only requires the model to capture complex dynamics in the ocean but also maintain good generality. The task is to predict the future 10 frames based on the past 10 observations, corresponding to predicting sea surface temperature in the 10 coming days based on 10 past days' observations.

Spreading Ink The dataset consists of three open source short videos from (Deng et al., 2023). The length of three videos are 150, 189, and 139, respectively. Following the experiment setting in Vortex (2023), we split the training and test sets in chronological order by the ratio of 2:1 for each video. Given all the training parts, the goal is to predict all the testing frames at once. For example, for the first video, we need to train our model on the first 100 frames and directly adopt this model to predict all the future 50 frames at once, namely the long-term forecasting task. Since the prediction horizon is much longer than other tasks, this problem poses special challenges in handling accumulative errors. Also, for real fluid video datasets, our concerns also include the model's portrayal of motion continuity and the realism of the generated video.

B.2. Implementations

In this section, we illustrate the concrete design for incorporating boundary conditions, the aggregation operation, and BFECC with Runge-Kutta Integral within the Multihead Multiscale Integral Architecture.

Boundary Conditions Here we detail the implementation of incorporating boundary conditions as a supplementary of Eq. 4. For given boundary conditions $\mathbb{S}$ and the position $\mathbf{r}$, we calculate the correlation on the intersection between boundary $\mathbb{S}$ and $\mathbf{r}$ neighbour $\mathbf{N}_{\mathbf{r}}$. Concretely, we multiply the boundary mask $\mathbb{1}_{\mathbb{S}}$ to embedded neighbour feature $\widehat{\mathbf{x}}_{T-1}(\mathbf{r}')$, that is,

$$\mathbb{1}_{\mathbb{S}}(\mathbf{r}')\left(\widehat{\mathbf{x}}_T(\mathbf{r}) \cdot \widehat{\mathbf{x}}_{T-1}(\mathbf{r}')\right) = \left(\widehat{\mathbf{x}}_T(\mathbf{r}) \cdot \mathbb{1}_{\mathbb{S}}(\mathbf{r}')\left(\widehat{\mathbf{x}}_{T-1}(\mathbf{r}')\right)\right), \ \mathbf{r}' \in \mathbb{V}. \tag{12}$$

This will preserve the number of neighbor correlation channels, and for $\mathbf{r}' \notin \mathbb{S}$, the correlation values will be set to zero.

Aggregation Operation Given learned deep representations of prediction $\widehat{\mathbf{x}}^{l}_{(T+1)}, \widehat{\mathbf{x}}^{l+1}_{(T+1)}$ at the $(l+1)$-th and l-th scales, the aggregation operation integrates information between different scales, which can be formalized as follows:

$$\widehat{\mathbf{x}}^{l}_{(T+1)} = \mathrm{Conv}\left(\mathrm{Concat}\left[\left(\mathrm{Upsample}\left(\widehat{\mathbf{x}}^{l+1}_{(T+1)}\right)\right), \widehat{\mathbf{x}}^{l}_{(T+1)}\right]\right), \ l \text{ from } (L-1) \text{ to } 1,$$

where we use bilinear interpolation for the operator $\mathrm{Upsample}(\cdot)$.

BFECC with Runge-Kutta Integral As mentioned in the main text, the second-order Runge-Kutta method can be expressed as $\mathrm{RK2}(\mathbf{r}, \mathbf{v}) = \mathbf{r} + \mathbf{v}(\mathbf{r} + \mathbf{v}(\mathbf{r})\frac{\mathrm{dt}}{2})\mathrm{dt}$, where $\mathbf{r} \in \mathbb{R}^{H \times W}$ and $\mathbf{v} \in \mathbb{R}^{2 \times H \times W}$. Relying solely on the Runge-Kutta method for temporal integration may result in error accumulations during advection. To address this issue, BFECC employs a combination of forward and backward integrals, which can be formulated as follows:

$$\begin{aligned} \mathbf{r}'_{\text{Forth}} &= \mathrm{RK2}(\mathbf{r}, \mathbf{v}) \\ \mathbf{r}'_{\text{Back}} &= \mathrm{RK2}(\mathbf{r}'_{\text{Forth}}, -\mathbf{v}) \\ \tilde{\mathbf{r}} &= \mathbf{r} + \frac{\mathbf{r} - \mathbf{r}'_{\text{Back}}}{2} \\ \mathrm{BFECC}(\mathbf{r}, \mathbf{v}) &= \mathrm{RK2}(\tilde{\mathbf{r}}, \mathbf{v}) \end{aligned} \tag{13}$$

For a given point $\mathbf{r}$, we initially integrate with the velocity vector $\mathbf{v}$ to obtain $\mathbf{r}'_{\text{Forth}}$. Subsequently, we utilize $\mathbf{r}'_{\text{Forth}}$ in conjunction with $-\mathbf{v}$ to perform another integral, resulting in $\mathbf{r}'_{\text{Back}}$. The disparity between $\mathbf{r}$ and $\mathbf{r}'_{\text{Back}}$ signifies the deviation between forward and backward integrals, with half of this difference employed to correct the initial position $\mathbf{r}$.

B.3. Metrics and Standard Deviations

In all four datasets, we report the mean value of relative L2 of three repeated experiments with different random seeds as a main metric. Experimentally, the standard deviations of relative L2 are smaller than 0.001 for Navier-Stokes, Bounded N-S and Sea temperature and smaller than 0.003 for Spreading Ink. For scientific rigor, we keep four decimal places for all results. For the Sea Temperature dataset, we report the MSE loss following the common practice in meteorological forecasting. For the Spreading Ink dataset, we used VGG Perceptual Loss (Johnson et al., 2016) to measure the realism of the generated fluids. Given n step predictions $\{\widehat{\mathbf{x}}_i\}_{i=1,\cdots,n}$ and corresponding ground truth $\{\mathbf{x}_i\}_{i=1,\cdots,n}$, $\widehat{\mathbf{x}}_i, \mathbf{x}_i \in \mathbb{R}^{H\times W}$, the above-mentioned metrics can be calculated as follows:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{H\times W}\|\mathbf{x}_i - \widehat{\mathbf{x}}_i\|_2^2, \quad \text{Relative L2 Loss} = \frac{\sqrt{\sum_{i=1}^{n}\|\mathbf{x}_i - \widehat{\mathbf{x}}_i\|_2^2}}{\sqrt{\sum_{i=1}^{n}\|\mathbf{x}_i\|_2^2}}.$$

Specifically, for the Spreading Ink dataset and Bounded N-S dataset with prescribed boundary conditions $\mathbb{S}$, we exclusively calculate the loss function within the specified boundary and subsequently report the average. Let $\mathbb{D}$ denote the region inside the container, and thus, MSE and Relative L2 can be computed as follows:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{|\mathbb{D}|}\sum_{(j,k)\in\mathbb{D}}(\mathbf{x}_{ijk} - \widehat{\mathbf{x}}_{ijk})^2, \quad \text{Relative L2 Loss} = \frac{\sqrt{\sum_{i=1}^{n}\sum_{(j,k)\in\mathbb{D}}(\mathbf{x}_{ijk} - \widehat{\mathbf{x}}_{ijk})^2}}{\sqrt{\sum_{i=1}^{n}\sum_{(j,k)\in\mathbb{D}}\mathbf{x}_{ijk}^2}},$$

where $\mathbf{x}_{ijk}$ represents the value at position (j,k) of $i-$th frame, and $|\mathbb{D}|$ represents the number of grid points in $\mathbb{D}$.

B.4. Model and Experiment Configurations

All the experiments are implemented in PyTorch(Paszke et al., 2019), and conducted on a single NVIDIA A100 40GB GPU. We repeat all the experiments three times with random seeds selected from 0 to 1000 and report the average results. We train the model with Adam optimizer (Kingma & Ba, 2015) for all baselines. See Table 9 for details.

Table 9. Experiment configurations in HelmFluid for different benchmarks.

BENCHMARK	LEARNING RATE	BATCH SIZE
NAVIER-STOKES	5×10^{-5}	10
BOUNDED N-S	5×10^{-5}	5
ERA5 Z500	5×10^{-5}	5
SEA TEMPERATURE	5×10^{-5}	10
SPREADING INK	5×10^{-5}	5

In this section, we provide a detailed overview of the model configurations for HelmFluid. Given that fluid dynamics vary across different resolutions, we augment the number of scales for larger inputs, as outlined in Table 10. For the Multihead Multiscale Integral Architecture, we adhere to the conventional design principles of U-Net (Ronneberger et al., 2015), incorporating downsampling, upsampling, and the aggregation of multiscale features.

Table 10. Hyperparameter configurations of HelmFluid for different resolutions.

INPUT RESOLUTIONS	HYPERPARAMETERS	VALUES
64×64	NUMBER OF SCALES L	3
	NUMBER OF HEADS M	4
	CHANNELS OF DEEP REPRESENTATIONS $\{d_{\text{MODEL}}^1,\cdots,d_{\text{MODEL}}^L\}$	$\{64,128,128\}$
	NUMBER OF NEIGHBOURS TO CALCULATE SPATIOTEMPORAL CORRELATIONS $\lvert\mathbf{N_r}\rvert$	81
128×128 256×256	NUMBER OF SCALES L	4
	NUMBER OF HEADS M	4
	CHANNELS OF DEEP REPRESENTATIONS $\{d_{\text{MODEL}}^1,\cdots,d_{\text{MODEL}}^L\}$	$\{128,256,512,512\}$
	NUMBER OF NEIGHBOURS TO CALCULATE SPATIOTEMPORAL CORRELATIONS $\lvert\mathbf{N_r}\rvert$	81

C. Ablation Study

As a supplementary analysis to the main text, we perform detailed ablations in the quantitative aspect to validate the impact of learning HelmDynamics and accounting for boundary conditions.

C.1. HelmDynamics Block

In this subsection, we will discuss the design of HelmDynamics Block from three perspectives. First, the necessity of learning velocity from HelmDynamics. Second, the effectiveness of potential and stream functions. Third, the usefulness of multilevel modeling, which we mentioned above, enhances the consistency of velocity fields at different scales. We compare the result on the 64×64 resolution Navier-Stokes dataset and report relative L2, training time, and GPU memory.

Learning HelmDynamics or Directly Learning Velocity We provide the results in Figure 13, and compare between learning velocity with HelmDynamics and learning velocity directly. We discover that directly learning the superficial velocity will overwhelm the model from capturing complex fluid interactions. As presented in Table 11, without Helmholtz dynamics, the performance decreases from 0.1261 to 0.1412, demonstrating the effectiveness of our proposed Helmholtz dynamics. In addition, the calculation of HelmDynamics only brings marginal extra computation costs.

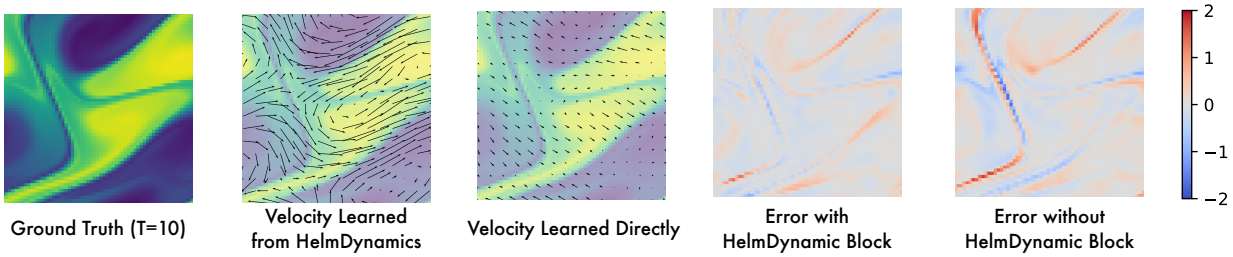


Figure 13. Velocity field and error comparison between learning by HelmDynamics Block and learning directly.

Table 11. Ablations on dynamics learning in 64×64 Navier-Stokes Dataset.

	Multihead version		Single head version	
Metrics	Velocity	HelmDynamics	Velocity	HelmDynamics
Relative L2	0.1412	**0.1261**	0.1461	**0.1344**
GPU memory (GB)	14.86	16.30	13.02	14.41
Training Time (s / epoch)	72.18	80.20	48.25	61.22

Are Both Potential and Stream Functions Effective? As presented in Table 12, only learning potential function or stream function will cause a decrease in the final performance, demonstrating the effectiveness of both components.

Table 12. Ablations on learning HelmDynamics, single potential or stream function in 64×64 Navier-Stokes Dataset.

Metrics	HelmDynamics	Only potential function	Only stream function
Relative L2	**0.1261**	0.1460	0.1305
GPU memory (GB)	16.30	16.29	16.30
Training Time (s / epoch)	80.20	79.57	79.60

Learning HelmDynamics in Multiple Scales As presented in Eq. 8, we ensemble the learned HelmDynamics in multiple scales. Here we also provide ablations on just employing HelmDynamics in one single scale in Table 13. We can find that our multiscale design can facilitate the dynamics modeling.

Number of neighbors in correlation calculation A larger regional area will provide more information for spatiotemporal correlation calculation. In this paper, we choose $|\mathbf{N_r}|$ as 9×9. Here in Table 14, we also compared with 7×7, 5×5, and 3×3. The model parameters are quite close for several neighbor sizes, but the best performance is obtained for 9×9.

Table 13. Ablations on learning HelmDynamics in multiple or single scales.

METRICS	MULTIPLE SCALES	SINGLE SCALE (BOTTOM)	SINGLE SCALE (TOP)
RELATIVE L2	**0.1261**	0.1441	0.1798
GPU MEMORY (GB)	16.30	8.80	11.68
TRAINING TIME (S / EPOCH)	80.20	30.69	45.32

Table 14. Ablations on the number of neighbors in correlation calculation.

NUMBER OF NEIGHBOURS IN CORRELATION $\lvert \mathbf{N_r} \rvert$	3×3	5×5	7×7	9×9
PARAMETER NUMBER	9,825,421	9,848,461	9,883,021	9,929,101
RELATIVE L2	0.1337	0.1273	0.1272	**0.1261**

C.2. Boundary Conditions

In this subsection, we discuss the design for boundary condition in correlation calculation, compare the result on the Bounded N-S dataset, and report relative L2, training time and GPU memory. As shown in Figure 14, while omitting boundary conditions, the learned velocities are perpendicular to the boundary, leading to discontinuous predictions. We present the quantitative results in Table 15, without input boundary condition, the performance drops seriously, indicating the necessity of our design in HelmDyanmics. It is also notable that as a flexible module, it is quite convenient to incorporate boundary conditions into the HelmDyanmics block, which is also a unique advantage of our model against others.

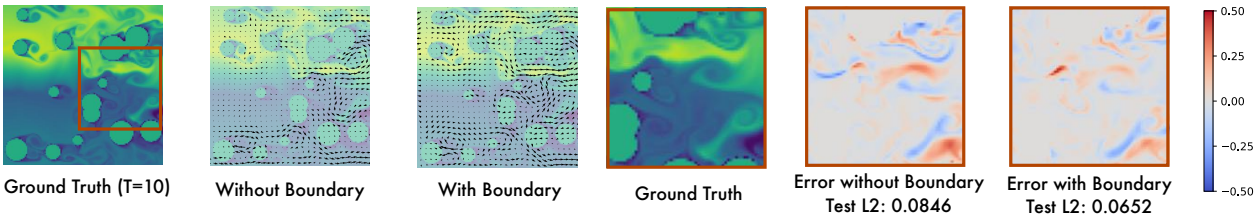


Figure 14. Velocity field and error comparison between learning by HelmDynamics Block and learning directly.

Table 15. Ablations on boundary conditions in the Bounded N-S dataset.

METRICS	OMITTING BOUNDARY CONDITIONS	USING BOUNDARY CONDITIONS
RELATIVE L2	0.0846	**0.0652**
GPU MEMORY (GB)	26.98	29.48
TRAINING TIME (S / EPOCH)	226.20	267.63

C.3. Multiscale Multihead Integral Architecture

We include a summary of ablations for multiscale multihead integral architecture in Table 16, including order of runge-kutta for temporal integral, number of heads and number of scales in multiscale multihead integral architecture. We compare the results on the Navier-Stokes dataset of 64 × 64 resolution and report relative L2, training time, and GPU memory.

Order of Runge-Kutta for temporal integral Runge-Kutta methods are widely employed for iteratively solving PDEs. The accuracy of the results increases with a higher number of orders but at the cost of additional computation time. In HelmFluid, the accuracy of prediction results is contingent on the precision of the velocity obtained through HelmDynamics blocks. According to our experiments, the second-order Runge-Kutta method is already sufficient for temporal integral, while less order leads to inaccuracies, and more order leads to about 10% more training time. Thus, we choose the second-order Runge-Kutta for integral to trade off performance and efficiency.

Number of heads in multihead modeling Adding heads is a convention to augment model capacity (Vaswani et al., 2017), and in fluid prediction, different heads can capture different dynamic patterns. In this paper, adding heads also means more operations in conducting integral. We set M as 4 for a good balance of running time and performance.

Table 16. Model performances on Navier-Stokes Dataset of 64×64 resolution with different selections for orders of Runge-Kutta, number of neighbors in correlations, number of heads and number of scales in multiscale multihead integral architecture.
The red marked hyperparameter represents the final configuration of HelmFluid.

ORDER OF RUNGE-KUTTA	1	2	3	4
RELATIVE L2	0.1298	**0.1261**	0.1268	0.1278
TRAINING TIME (S / EPOCH)	80.04	81.20	88.30	90.49
NUMBER OF HEADS M	1	4	8	16
PARAMETER NUMBER	11,063,245	9,929,101	9,812,653	9,762,205
RELATIVE L2	0.1344	<u>0.1261</u>	0.1279	**0.1249**
TRAINING TIME (S / EPOCH)	59.69	81.20	120.86	171.97
NUMBER OF SCALES L	2	3	4	5
PARAMETER NUMBER	9,283,977	9,929,101	15,906,193	29,820,309
RELATIVE L2	0.1514	**0.1261**	0.1361	<u>0.1330</u>
TRAINING TIME (S / EPOCH)	64.43	81.20	99.83	120.06

Number of scales in multiscale modeling This hyperparameter is highly related to the nature of fluid. Considering both model efficiency and fluid dynamics, we choose L as 3 for 64×64 inputs and 4 for larger inputs.

C.4. Sensitivity to the Number of Parameters

We also add the sensitivity analysis to the number of parameters on the 64×64 Navier-Stokes dataset in Table 17. We report the results of changing the channels of deep representations to a half and twice the original channels. These results show that the original configuration can achieve a favorable balance between performance of efficiency.

Table 17. Ablations on the number of parameters.

CHANNELS COMPARED TO THE OFFICIAL CONFIGURATION	1/2	1	2
RELATIVE L2	0.1380	<u>0.1261</u>	**0.1242**
GPU MEMORY (GB)	9.64	16.30	29.99
RUNNING TIME (S / EPOCH)	75.10	80.20	112.13
#PARAMETER	2,516,173	9,929,101	39,446,029

D. Additional Results

D.1. Efficiency comparision

Efficiency for different models In the main text, we presented an efficiency comparison through plots. Here, we provide detailed quantitative results in Table 18 as a supplementary reference.

Table 18. Efficiency comparison between six deep models on Naiver-Stokes 64×64 dataset, where we fixed the batch size to 10.

MODELS	HELMFLUID	U-NET	FNO	MWT	U-NO	LSM
#PARAMETER	9,929,101	17,312,650	1,188,641	7,989,593	61,157,793	19,188,033
TRAINING TIME (S / EPOCH)	80.20	46.29	18.91	90.02	103.82	44.49
RELATIVE L2	**0.1261**	0.1982	0.1556	0.1586	<u>0.1435</u>	0.1535

Align model size It's important to note that all baseline models are reproduced using their official configurations from their respective papers, which might lead to an unbalanced model size issue. To ensure a fair comparison, we also scale up

the parameters of FNO and compare it with HelmFluid. Refer to Table 19 for the results. Notably, even with an increased parameter size comparable to HelmFluid, FNO still exhibits inferior performance.

Table 19. Align model size in 64 × 64 Navier-Stokes.

RELATIVE L2	FNO	FNO (ENLARGED)	HELMFLUID
64 × 64 NAVIER-STOKES	0.1556	0.1524	**0.1261**
128 × 128 NAVIER-STOKES	0.1028	0.1025	**0.0807**
256 × 256 NAVIER-STOKES	0.1645	0.1474	**0.1310**
BOUNDED N-S	0.1176	0.1116	**0.0652**
SEA TEMPERATURE	0.1935	0.1958	**0.1704**
VIDEO 1	0.1709	0.1872	**0.1399**
VIDEO 2	0.4864	0.5250	**0.3565**
VIDEO 3	0.1756	0.1676	**0.1584**
MODEL PARAMETER	1,188,641	10,633,265	9,929,101

D.2. Full Results for Spreading Ink

We reported the averaged metrics on Spreading Ink dataset. Here, we detail the metrics on three sub-datasets respectively in Table 20. Except Relative L2 and MSE are worse than U-Net on Video3, HelmFluid consistently outperforms other models on the other metrics. However, images generated by U-Net appear fragmented, worse than HelmFluid from a visual perspective. Additionally, we present the showcases of three sub-datasets in Figure 25, 26 and 27.

Table 20. Model comparison on Spreading Ink for each video. Perceptual loss, Relative L2 and MSE are reported.

MODEL	VIDEO1	VIDEO2	VIDEO3
U-NET (RONNEBERGER ET AL., 2015)	1.500 / 0.1544 / 0.0080	3.982 / 0.4780 / 0.0330	5.307 / **0.1535** / **0.0119**
FNO (LI ET AL., 2021)	2.023 / 0.1709 / 0.0097	4.732 / 0.4864 / 0.0342	5.531 / 0.1756 / 0.0156
U-NO (RAHMAN ET AL., 2023)	4.210 / 0.1792 / 0.0106	6.204 / 0.5312 / 0.0408	6.397 / 0.1810 / 0.0166
VORTEX (DENG ET AL., 2023)	1.704 / 0.1580 / 0.0083	4.171 / 0.4150 / 0.0249	5.973 / 0.1718 / 0.0150
LSM (WU ET AL., 2023)	1.666 / 0.1592 / 0.0084	4.167 / 0.4890 / 0.0346	5.448 / 0.1611 / 0.0132
HELMFLUID (OURS)	**1.464** / **0.1399** / **0.0065**	**3.296** / **0.3565** / **0.0184**	**5.208** / 0.1584 / 0.0127

D.3. Align baselines in all benchmarks

As we stated in Section 4, some of the baselines are not suitable for part of the benchmarks, specifically Vortex (Deng et al., 2023), DARTS (Ruzanski et al., 2011), PWC-Net with fluid Refinement (Sun et al., 2018) and MWT (Gupta et al., 2021), which means their performance will degenerate seriously or the running time is extremely slow if we stiffly apply them to all benchmarks. Specifically, due to the special design for temporal information in Vortex, we only compare it in the Spreading Ink dataset in the main text. As for the DARTS, since it is designed for the mass field and not applicable for videos with RGB channels, we do not include it in the Spreading Ink dataset. Besides, PWC-Net with fluid Refinement (Zhang et al., 2022) is proposed to learn the optical flow for fluid, which suffers from the accumulative error, making it far inferior to other methods. Thus, we only compare PWC-Net in the learning velocity field in the main text.

However, we still provide the missing experiments in Table 21 to ensure transparency.

- Vortex (Deng et al., 2023) models multiple vortex trajectories as a function of time. Since different video sequences have inherently different vortex trajectories, we need to re-train Vortex to fit every video sequence. However, the other three benchmarks except Spreading Ink, have more than 1000 different video sequences. It means that we need to train 1000+ Vortex models for these benchmarks, which is unacceptable. But we still implement this experiment, where we train one vortex model on one single video sequence and generalize it to others.

- Due to the slow movement of the spreading ink dataset, DARTS (Ruzanski et al., 2011) showed outstanding quantitative results. However, it fails to predict the correct future in the other three datasets. Moreover, DARTS method solves the least squares problem in the frequency domain for every case, which will bring huge computation costs. In particular,

Table 21. Align baselines in all benchmarks, including DARTS (Ruzanski et al., 2011), adapted version of PWC-Net (Sun et al., 2018), MWT (Gupta et al., 2021), Vortex (Deng et al., 2023). We report Relative L2 for the Navier-Stokes dataset and Bounded N-S dataset, MSE and relative L2 for the Sea Temperature dataset, and Perceptual loss, Relative L2 and MSE for Spreading Ink.

	NAVIER-STOKES	BOUNDED N-S	SEA TEMPERATURE	SPREADING INK (VIDEO 3)
DARTS	0.8046	0.1820	0.3308 / 0.1094	4.940 / 0.1601 / 0.0130
PWC-UNET	0.1765	<u>0.0729</u>	<u>0.1805</u> / <u>0.0406</u>	5.341 / <u>0.1591</u> / <u>0.0128</u>
MWT	<u>0.1586</u>	0.1407	0.2075 / 0.0510	**1.521** / 0.1775 / 0.0160
VORTEX	8.1379	1.6259	4.9302 / 0.1796	5.973 / 0.1718 / 0.0150
HELMFLUID	**0.1261**	**0.0652**	**0.1704 / 0.0368**	<u>5.208</u> / **0.1584 / 0.0127**

the other deep methods predict the whole sequence in less than 0.1 seconds, while DARTS takes more than 10 seconds. Also, the changes in estimated velocity are very slight with the change of time, which leads to incorrect location estimation. And the extrapolation causes blurring in long-term prediction.

- PWC-Net (Sun et al., 2018) specifically focuses on estimating velocity between adjacent observations. Initially, we attempted to extrapolate predictions using the estimated velocity field, but this approach resulted in severe distortion. To harness the estimated velocity more effectively, we input both the velocity and observations into a U-Net (Ronneberger et al., 2015), yielding improved results denoted as PWC-UNet in Table 21. Despite the enhancement provided by the estimated velocity from PWC-Net, U-Net still falls short compared to HelmFluid.

- MWT (Gupta et al., 2021) predict the future frames based on wavelet analysis. It fails in long-term prediction. The prediction on video 3 of Spreading Ink (Figure 15) shows that as the prediction time gets longer, the prediction image stays at the same position and appears weird texture.

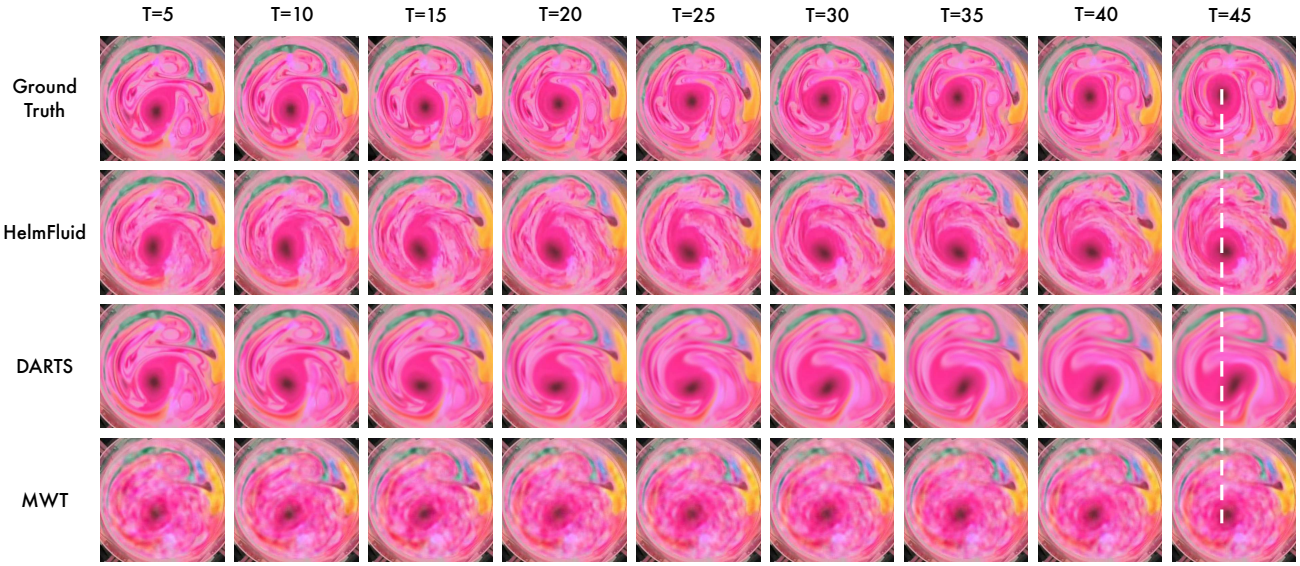


Figure 15. Showcases of HelmFluid, DARTS, and MWT on the Spread Ink dataset .

D.4. Performance on turbulence dataset

To effectively show the model performance in handling turbulent fluid, we assess HelmFluid and other baseline models using a turbulence dataset with dimensions of 64×64 (Wang et al., 2019). This dataset comprises 6000 sequences for training, 1700 for validation, and 2100 for testing. The objective is to predict subsequent velocity fields based on preceding observations. For optimal performance on the dataset, all models are trained with an input sequence of 25 timesteps and evaluated over 20 timesteps. The results are presented in Table 22.

Table 22. Performance on turbulence dataset.

TURBULENCE DATASET	MSE
U-NET (RONNEBERGER ET AL., 2015)	1062.13
TF-NET (WANG ET AL., 2019)	<u>1061.78</u>
FNO (LI ET AL., 2021)	1187.44
U-NO (RAHMAN ET AL., 2023)	3276.09
LSM (WU ET AL., 2023)	1069.26
HELMFLUID	**1042.38**

E. More Showcases

As a supplement to the main text, we provide more showcases here for comparison (Figure 16-27). Videos are provided in Supplementary Materials.

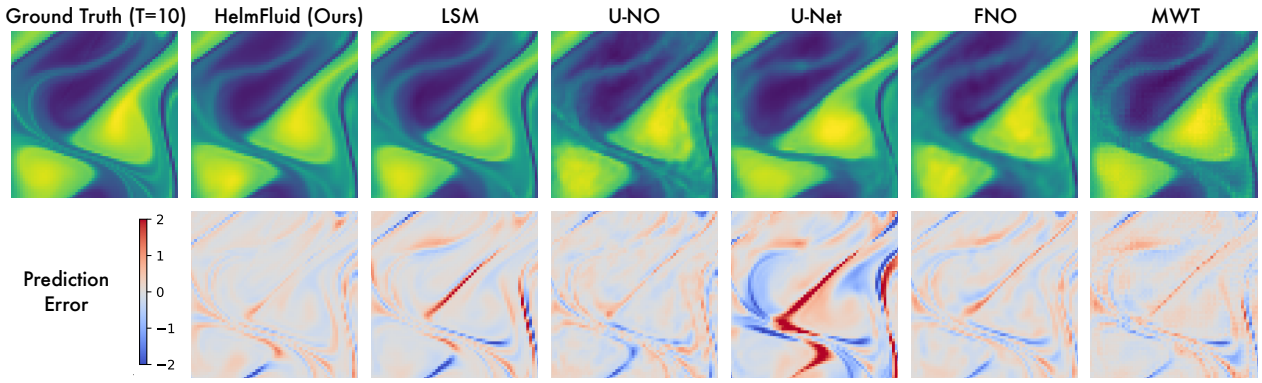


Figure 16. Showcases of the Navier-Stokes dataset with resolution of 64×64.

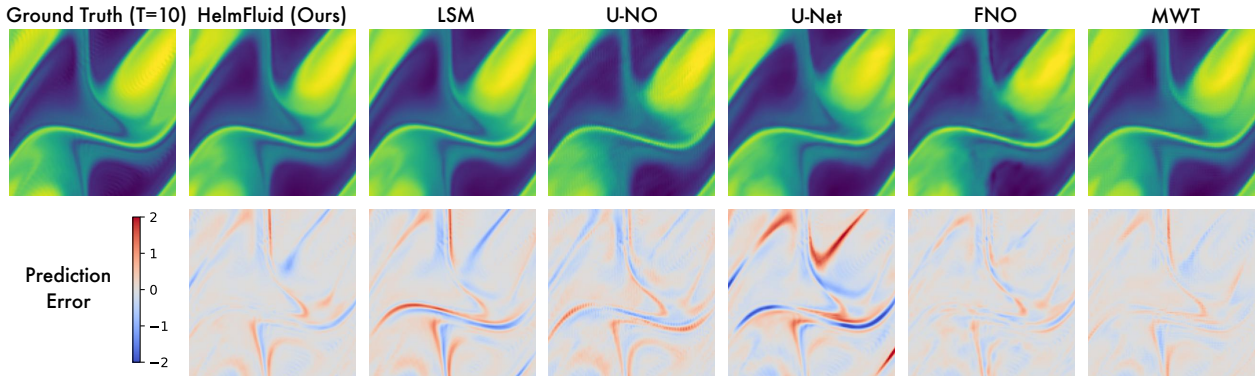


Figure 17. Showcases of the Navier-Stokes dataset with resolution of 128×128.

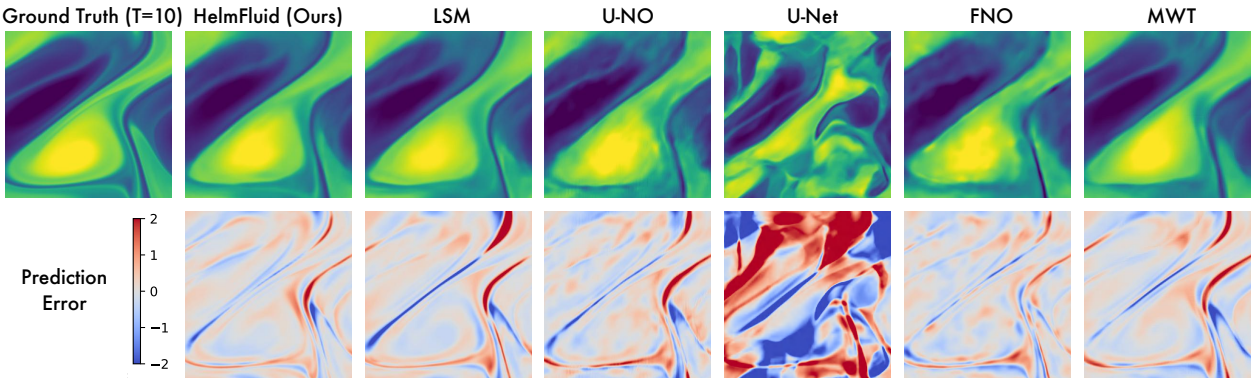


Figure 18. Showcases of the Navier-Stokes dataset with resolution of 256×256.

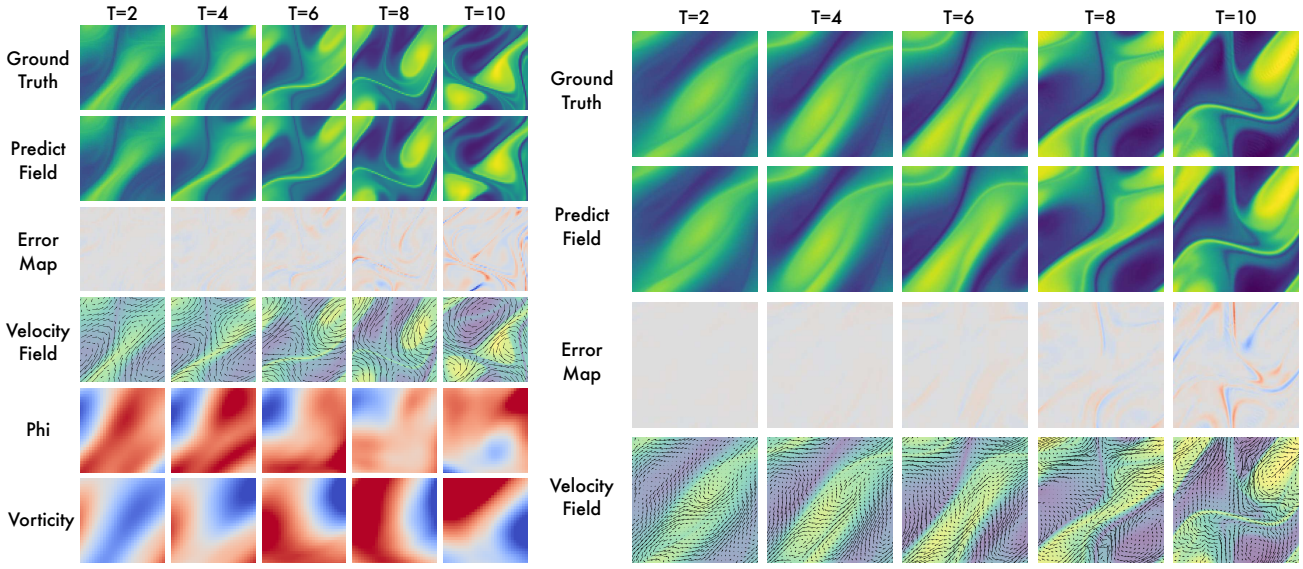


Figure 19. Showcases of HelmFluid on Navier-Stokes dataset with resolution 64×64 and 128×128.

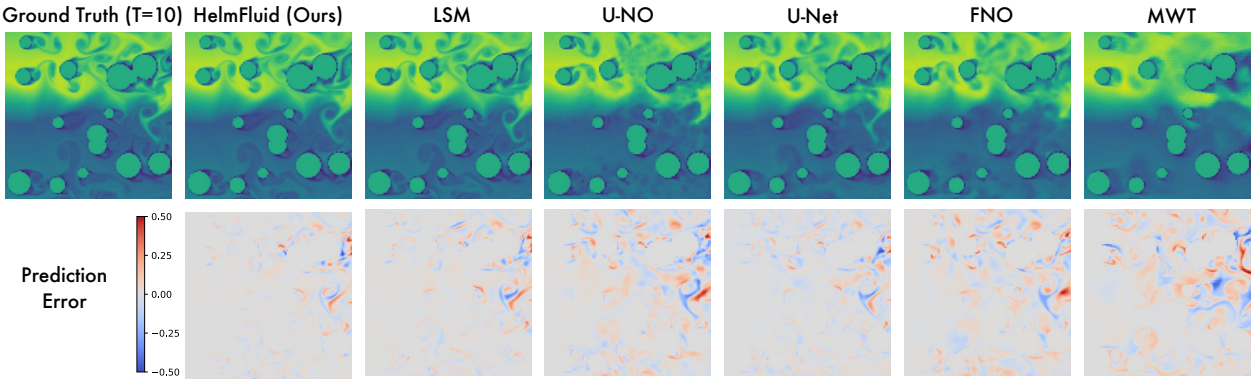


Figure 20. Showcases of the Bounded N-S dataset.

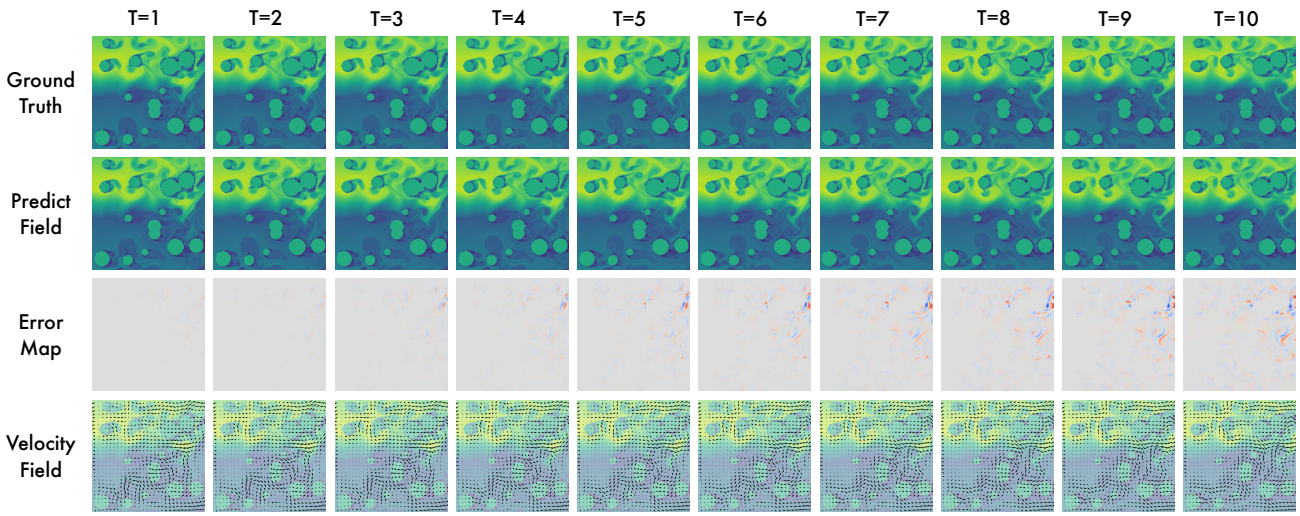


Figure 21. Showcases of HelmFluid on the Bounded N-S dataset.

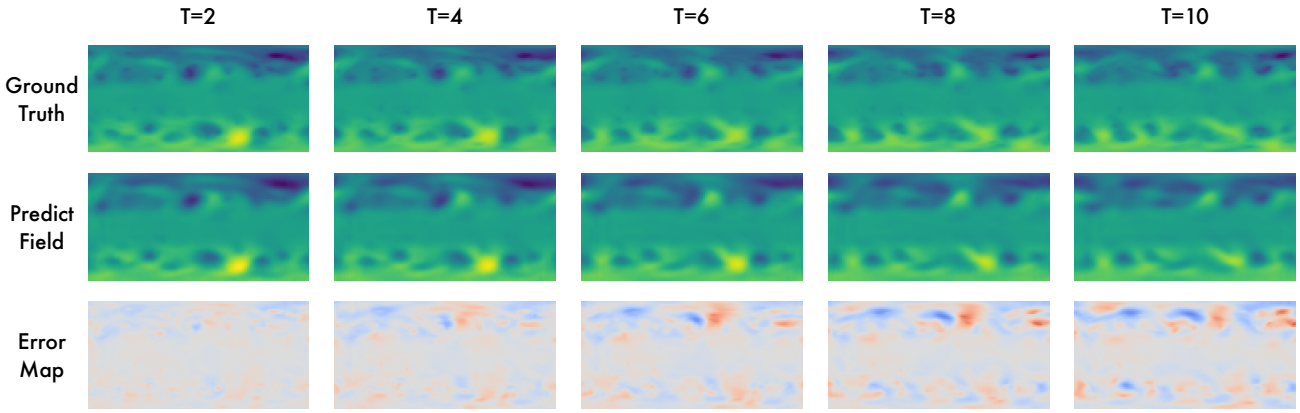


Figure 22. Showcases of the ERA5 Z500 dataset.

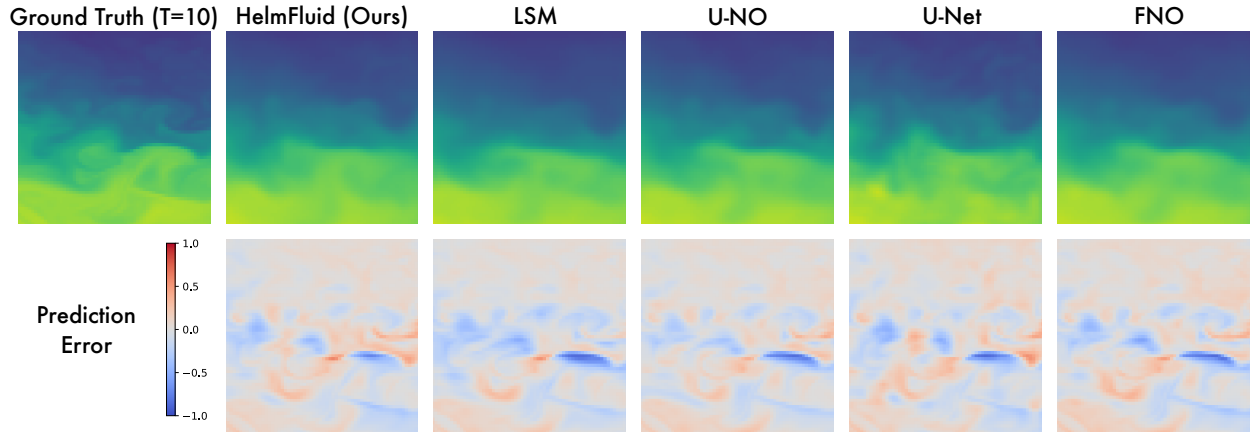


Figure 23. Showcases of the Sea Temperature dataset.

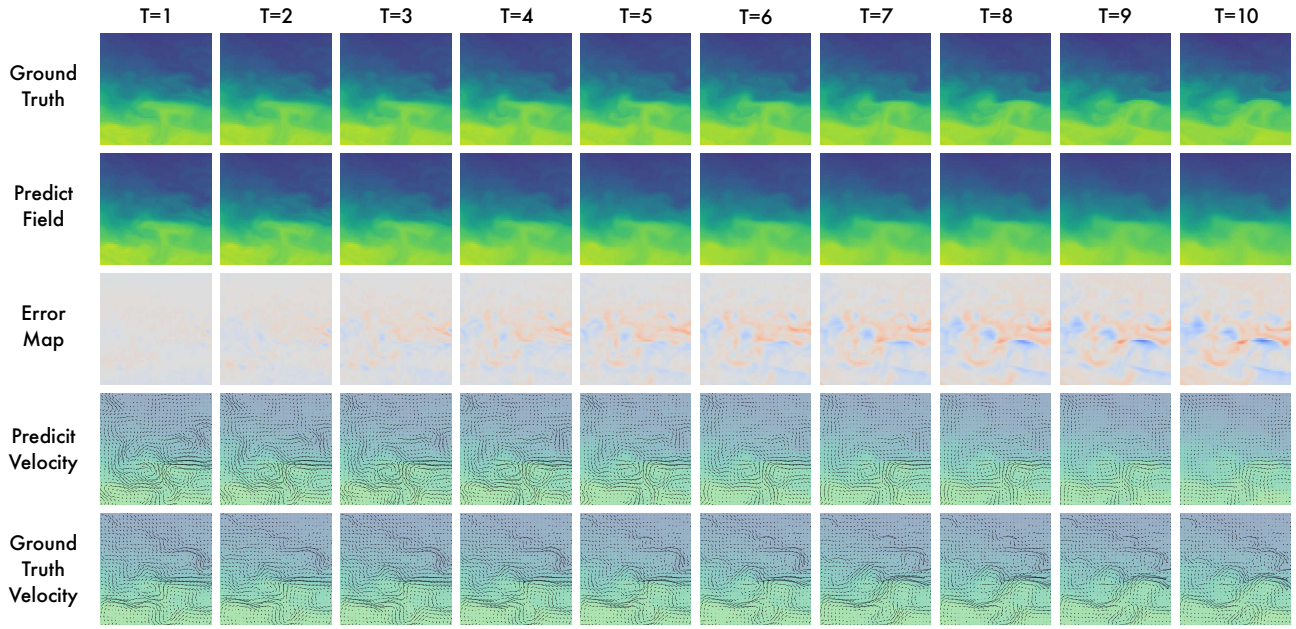


Figure 24. Showcases of HelmFluid on the Sea Temperature dataset.

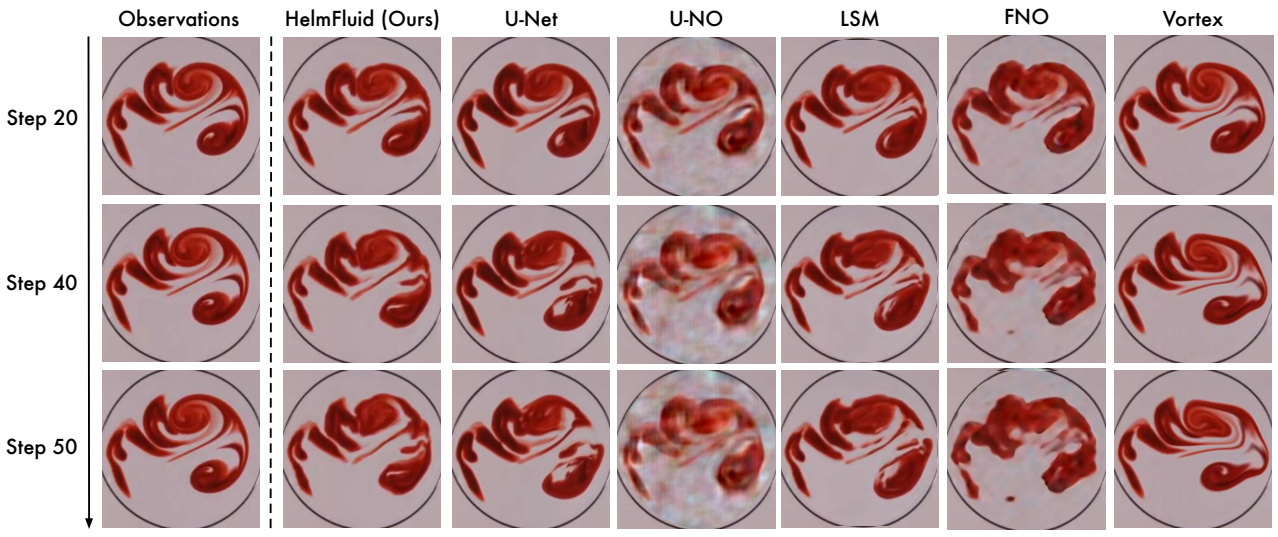


Figure 25. Showcases of the Spreading Ink dataset (Video 1).

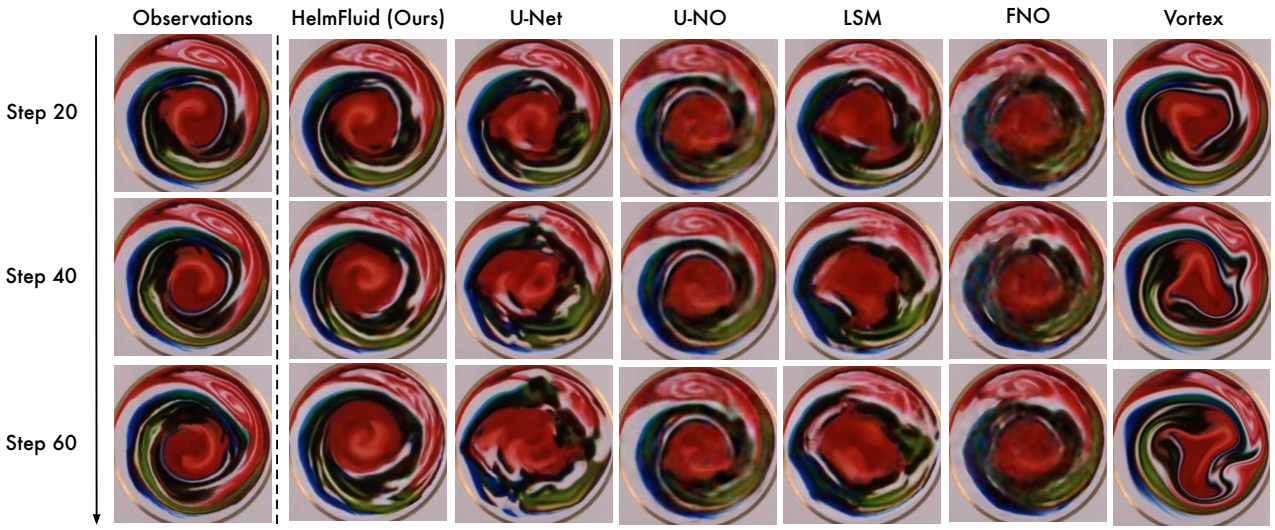


Figure 26. Showcases of the Spreading Ink dataset (Video 2).

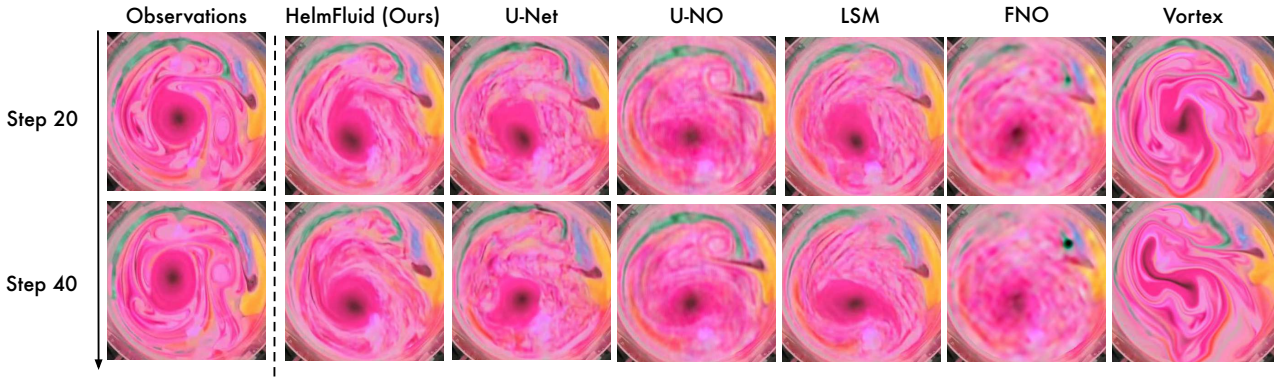


Figure 27. Showcases of the Spreading Ink dataset (Video 3).