FD-BENCH: A MODULAR AND FAIR BENCHMARK FOR DATA-DRIVEN FLUID SIMULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Data-driven modeling of fluid dynamics has advanced rapidly with neural PDE solvers, yet a fair and strong benchmark remains fragmented due to the absence of unified PDE datasets and standardized evaluation protocols. Although architectural innovations are abundant, fair assessment is further impeded by the lack of a clear disentanglement between spatial, temporal and loss modules. In this paper, we introduce **FD-Bench**, the first fair, modular, comprehensive and reproducible benchmark for data-driven fluid simulation. FD-Bench systematically reviews and decomposes 89 baseline models reported across recent publications, extracting and standardizing their key architectural and training components for fair, unified comparison across 10 representative flow scenarios. It provides four key contributions: (1) a modular design enabling fair comparisons across spatial, temporal, and loss function modules; (2) the first systematic framework for direct comparison with traditional numerical solvers; (3) fine-grained generalization analysis across resolutions, initial conditions, and prediction time window]; and (4) a user-friendly, extensible codebase to support future research. Through rigorous empirical studies, FD-Bench establishes the most comprehensive leaderboard to date, resolving long-standing issues in reproducibility and comparability, and laying a foundation for robust evaluation of future data-driven fluid models. The code is open-sourced at https://anonymous.4open.science/r/FD-Bench-15BC.

1 INTRODUCTION

The accurate simulation of complex fluid dynamical systems governed by partial differential equations (PDEs) is a cornerstone of scientific and engineering applications, ranging from aerodynamics (O'Connell et al., 2022; Deng et al., 2023; Mufti et al., 2024), chemical engineering (Cao & Li, 2018; Cao et al., 2019; Zhang et al., 2022), biology (Yin et al., 2022; Voorter et al., 2023; Shen Wong et al., 2023), and environmental science (Wen et al., 2022; Pathak et al., 2022; Bi et al., 2023; Rajagopal et al., 2023; Zhang et al., 2024). Traditional numerical solvers have achieved remarkable success in offering high-fidelity solutions grounded in well-established mathematical formulations and rigorous convergence guarantees. Recent advancements in machine learning have introduced numerous neural solvers for data-driven fluid simulation (Wang et al., 2024a). Data-driven approaches enable efficient prediction of complex dynamics with reduced computational cost. These methods have shown strong potential in capturing nonlinear behaviors beyond the reach of traditional solvers.

Despite rapid progress, the field faces significant challenges in establishing consistent evaluation standards. Researchers in both the field of AI and traditional Computational Fluid Dynamics (CFD) often face three pressing questions. **First**, how effective are current neural solvers, which methods demonstrate the best performance, and how do they compare against traditional numerical solvers in terms of accuracy? **Second**, how efficient are current neural solvers, and to what extent can they accelerate simulations compared to traditional numerical methods? **Third**, and perhaps most critically, how well do these neural solvers generalize when applied to more practical and realistic scenarios? Therefore, as recently emphasized by (Brandstetter, 2025), recent analyses reveal persistent challenges in making fair comparisons to prior methods. Advancing the field will require stronger and more consistent benchmark problems.

To date, the three aforementioned questions remain unanswered, largely due to the inherent infinite-dimensional continuity and variability of fluid dynamics, which makes definitive analysis challenging. Significant limitations remain in three key areas illustrated in Figure 1.

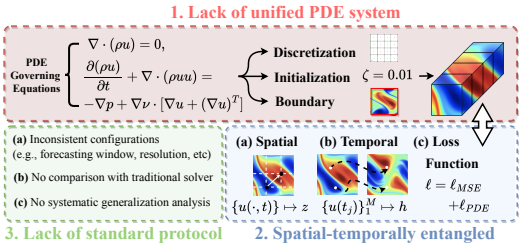


Figure 1: Limitations identified in three key areas.

(1) Lack of unified PDE systems. From a data perspective, the diversity of benchmark settings, ranging in governing equations, domain geometries, spatial resolutions, temporal discretizations, and flow conditions, hinders fair comparison across methods. This variability often results in fragmented and non-reproducible findings. As shown in Table 1, no two benchmarks employ the same dataset, making it difficult to draw reliable conclusions about model performance. **(2) Entangled spatial-temporal modeling and loss design** Capturing fluid dynamics requires architectures that jointly model spatial structures and temporal evolution while maintaining physical consistency. However, many neural solvers entangle components from diverse domains without clear attribution, and lack systematic ablations, making it difficult to isolate the source of performance gains. For example, FNO (Li et al., 2021a) learns spatiotemporal features by stacking the time dimension in the frequency domain, while MP-PDE (Brandstetter et al., 2022) introduces temporal bundling to enhance temporal evolution. These differing design choices make it nearly impossible to directly compare their individual contributions. **(3) Lack of standardized evaluation protocol.** From an evaluation standpoint, the dynamical nature of fluid systems, characterized by sensitivity to initial conditions and parameter settings, makes it challenging to derive robust conclusions or establish reproducible baselines. Table 1 presents a comparison of key aspects from our systematically constructed evaluation. This reveals three primary limitations: **(3.1)** Inconsistent experimental configurations. It is common for different studies to adopt distinct experimental setups, including variations in discretization, flow field conditions, forecasting windows, and other hyper-parameters. Moreover, the evaluation metrics are inconsistent across studies, which further limits the comparability of different methods. **(3.2)** Limited benchmarking against traditional solvers. Many neural solvers lack rigorous comparisons with classical numerical solvers, particularly under settings where accuracy, stability, and computational efficiency must be jointly evaluated. **(3.3)** Absence of systematic generalization analysis. Few studies examine how models generalize across different resolutions, conditions, or parameter regimes. This limits our understanding of whether models can extend to real-world applications.

To overcome these limitations, we introduce **FD-Bench**, a rigorously designed and fully reproducible benchmark. Rather than merely re-implementing prior work, FD-Bench systematically reviews and decomposes 89 baseline models reported across recent publications (see Appendix C), extracting and standardizing their key architectural and training components for fair, unified comparison across 10 representative flow scenarios. FD-Bench brings three key innovations to the field, each directly addressing the limitations above. **First**, we collect and generate 10 representative fluid flow scenarios that span a diverse range of physical conditions shown in Figure 2. Building on these datasets, we enable fair and direct comparisons among diverse neural solvers as well as against traditional numerical solvers for the first time. Besides, this unified system makes it possible to conduct fine-grained generation analysis, revealing how variations in initial conditions, spatial resolution, prediction time window, and model capacity affect performance. **Second**, FD-Bench enables modular and fair comparisons by isolating spatial, temporal, and loss function modules, thereby eliminating confounding factors introduced by implementation-specific details. We construct the most rigorous and comprehensive leaderboard to date, providing a strong foundation for fair and reproducible evaluation in the field. **Third**, FD-Bench addresses the long-standing lack of standardized evaluation protocols by providing a fully modular, easy-to-use, and reproducible codebase. Researchers can simply specify the dataset and select the desired combination to automatically run experiments under a unified and controlled setup. This design not only ensures that all baseline methods can be faithfully reproduced with minimal configuration but also enables seamless extension to novel architectures or loss functions, thereby significantly lowering the barrier to entry and promoting fair, reproducible, and extensible evaluation across the community.

This work not only resolves long-standing comparability issues but also delivers a flexible toolkit, laying the foundation for more transparent and reproducible research in data-driven fluid dynamics.

2 Related Work and Position of FD-Bench

Comparisons with existing benchmarks. Table 1 provides an overview of existing benchmarks for data-driven simulation. PDEBench (Takamoto et al., 2022) first significantly advanced the field by

Benchmark	**1.** Different grid discretization	**2.** Long-term rollout	**3.** Numerical method comp.	**4.** Temporal modular comp.	**5.** Spacial modular comp.	**6.** Loss modular comp.	**7.** Generalization study on conditions	# Baselines
PDEBench (Takamoto et al., 2022)	✗	✓	✓	✗	✗	✗	✗	3
CFDBench (Luo et al., 2023c)	✗	✓	✗	✗	✗	✗	✓	9
DiffBench (Kohl et al., 2023)	✗	✓	✗	✗	✗	✗	✗	6
PDENNEval (Wei et al., 2024)	✗	✓	✗	✗	✗	✓	✓	12
PINNacle (Hao et al., 2023b)	✗	✓	✗	✗	✗	✗	✓	12
Well (Ohana et al., 2024)	✗	✓	✗	✗	✗	✗	✗	4
Posiden (Herde et al., 2024)	✗	✓	✗	✗	✓	✗	✓	7
ML-PDE (McGreivy et al., 2024)	✗	✗	✓	✗	✗	✗	✗	10
InverseBench (Zheng et al., 2025)	✗	✓	✗	✗	✗	✗	✓	14
FD-Bench	✓	✓	✓	✓	✓	✓	✓	**89**

Table 1: A comprehensive comparison between our proposed **FD-Bench** and existing benchmarks. Each column represents a key evaluation dimension, where "comp." stands for "comparison". # Baselines indicates the number of existing methods reproduced and evaluated within each benchmark.

introducing standardized datasets. CFDBench (Luo et al., 2023c) extended this idea to a wider set of CFD problems and emphasized generalization across flow conditions. More specialized efforts such as PINNacle (Hao et al., 2023b) and PDENNEval (Wei et al., 2024) target physics-informed neural networks. More recent efforts such as Well (Ohana et al., 2024) and Posiden (Herde et al., 2024) further enriched the landscape by introducing their own curated datasets and conducting detailed experimental analyses, offering deeper insights into model behavior. However, most of them remain limited in scope: they typically focus on a narrow class of PDE systems or solver families. They just treat each baseline method as a monolithic whole, without probing the sources of its contributions, and consequently lack a fully modular decomposition of spatial, temporal, and loss components. Moreover, they seldom provide rigorous head-to-head comparisons with classical numerical solvers (McGreivy et al., 2024) .

FD-Bench aims to investigate four key questions in fluid simulation:

Q1. Does an optimal neural architecture exist? Modeling complex fluid dynamical systems often suffers from the "coupling curse," (Wu et al., 2024d) where spatio-temporal flow conditions are inextricably intertwined, making it difficult to pinpoint the true source of performance gains. FD-Bench proposes to systematically decouple fluid dynamical system modeling into three orthogonal dimensions to break the coupling curse, enabling direct attribution of model behavior to specific design choices. It also yields taxonomy-driven insights by aligning baselines within a unified schema.

Q2. Can neural solvers truly replace traditional numerical solvers? To obtain a fair comparison of both computational efficiency and predictive accuracy, we propose benchmarking neural solvers against traditional numerical solvers operating on coarser discretizations or using lower-order time-integration schemes (*e.g.*, Euler's method instead of fourth-order Runge–Kutta) that produce equivalent error. Thus, we extend the idea in McGreivy et al. (2024) to compare the rollout error of neural solvers to that of these coarse-grid solvers. We perform experiments on three subsets with varying parameters, evaluate their long-horizon rollout performance, and ensure a fair comparison under identical system environments.

Q3. Which spatial discretization scheme better supports their representative neural solvers? Neural solvers employ one of two discretization schemes: Eulerian or Lagrangian. In Eulerian approaches (grid- or mesh-based) (Subramaniam, 2013; Stam, 1999), the spatial domain is discretized by a fixed arrangement of nodes. In Lagrangian (particle-based) methods (Müller et al., 2003), discretization is realized through material points that move with the local deformation of the continuum. Existing works only compare models with the same discretization schemes. To this end, we generate three subsets to train their representative neural solvers and compare their rollout performance.

Q4. Whether knowledge learned from different systems boost generalization? In practical applications, evaluating a neural solver's performance in out-of-distribution scenarios is of greater significance. We provide a detailed investigation into how different modular designs perform under varying initial conditions, resolutions, model parameters, and longer rollout horizons. Specifically, we examine the zero-shot generalization ability of the models to unseen initial conditions, their convergence behavior when trained at different resolutions, their scalability with increased parameter size, and their ability to extrapolate in zero-shot rollout over extended time steps.

3 PDE Datasets Curation

3.1 Data Selection Criteria

To ensure a comprehensive evaluation, our benchmark incorporates datasets that rigorously satisfy the following requirements. (1) **Diverse PDE types**: Inclusion of PDE families with real-world physical significance, such as the N-S equations for fluid dynamics and Burgers' equation for shock waves.

Name	Shape	Size
Incompressible N-S	{1200, 1000, 256, 256}	2.26T
Compressible N-S	{42k, 4, 4, 128, 128 }	222G
Stochastic N-S	{100, 1k, 2, 128,128}	12G
Kolmogorov Flow	{20k, 21, 5, 128, 128}	17G
Diffusion-Reaction	{1k, 100, 2, 128,128}	13G
Taylor-Green Vortex	{204, 126, 2, 10000}	1.8G
Reverse Poiseuille Flow	{1, 30000, 2, 12800}	2.9G
Advection	{1200, 1000, 256, 256}	2.26T
Lid-driven Cavity Flow	{1, 20000, 2, 11236}	3.81T
Burgers	{1200, 1000, 2, 256, 256}	3.72T
Total	-	**8.59T**

Table 2: **Statistics** of flows. The "shape" column represents, in order: the trajectory sample number, time steps, feature channels, and the flow field resolution.

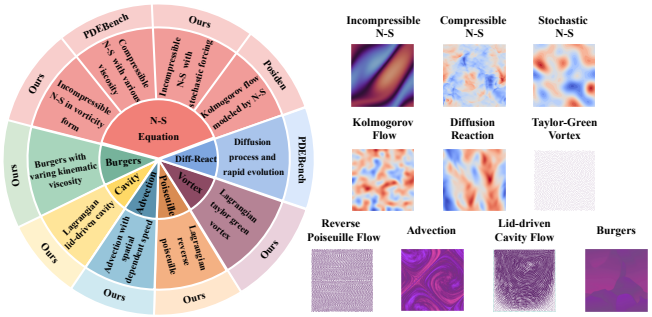


Figure 2: We collect and generate 10 representative fluid flow scenarios that span a diverse range of physical conditions. We also present the corresponding visualizations.

This ensures the universality of tested methods across mathematical formulations. (2) **Heterogeneous data distributions**: Coverage of varying initial conditions (*e.g.,* smooth, discontinuous, stochastic) and boundary conditions to rigorously evaluate generalization capabilities under distribution shifts. (3) **Multi-scale field settings**: Datasets spanning coarse-to-fine spatial resolutions and temporal discretizations, coupled with diverse numerical discretization schemes to assess resolution scalability and discretization invariance. (4) **Complex temporal dynamics**: Inclusion of both short-term transient behaviors (*e.g.,* rapid instabilities) and long-term equilibrium states, as well as systems with stiff dynamics or temporal discontinuities, to test robustness in temporal representation. (5) **Irregular geometries**: Domains with non-Cartesian boundaries (*e.g.,* curved and fractured).

3.2 Data Collection & Generation

For modular comparison, we collect three representative flow problems from existing benchmarks, namely `Compressible N-S` (Takamoto et al., 2022), `Diffusion-Reaction` (Takamoto et al., 2022), and `Kolmogorov Flow` (Takamoto et al., 2022) for fair comparison. In addition, we generate a `Stochastic N-S` subset, driven by incompressible flow with stochastic forcing and initialized with 10 distinct initial conditions, to further enhance the diversity of our experiments. For more details, please refer to Appendix J.1.

For comparison with traditional numerical solvers, we generate three subsets with varying parameters using a pseudo-spectral solver (Canuto et al., 1988). To simulate `Incompressible N-S` in vorticity form, we sample five Reynolds numbers: 50, 200, 500, 1k, and 2k. To simulate `Burgers`, we use five kinematic viscosities: 5×10^{-4}, 1×10^{-3}, 5×10^{-3}, 1×10^{-3}, and 5×10^{-2}. Finally, to simulate `Advection`, we consider five spatially varying advection speeds. Details on the solver implementation and the PDE formulations are provided in Appendix J.2.

For comparison on different discretizations, we use Smoothed Particle Hydrodynamics (SPH) (Gingold & Monaghan, 1977) to generate three Lagrangian particle subsets, `Taylor-Green vortex` (Re = 100), `Lid-driven cavity flow` (Re = 100), and `Reverse Poiseuille flow` (Re = 10), each governed by the compressible N-S equations. Particle positions are recorded every 100 time steps, and neural models are trained to predict the flow evolution over these intervals. We convert Lagrangian particle datasets into Eulerian form. For grid representations, we generate uniformly spaced nodes and aggregate particle velocity information onto these nodes. For mesh representations, we randomly sample Eulerian nodes and perform the same velocity-aggregation procedure. Aggregation is performed with a quintic smoothing kernel. See Appendix J.3 for additional details.

4 Decoupling in Modular Design

To enable a unified and fair comparison across 89 baselines, each originally proposed under different datasets and modeling paradigms, we creatively decompose each method into four key design dimensions inspired by a Taylor expansion: Design = "**Spatial representation**" + "**Temporal representation**" + "**Loss function**" + Additional Technique, where the "Additional Technique" captures the distinctive architectural or training innovation introduced by each work.

4.1 Spatial Representation

A key challenge is how to encode the continuous spatial field $u\colon \Omega \times [0,T] \to \mathbb{R}^d$. Let $\Omega \subset \mathbb{R}^d$ denote the physical domain, and let $u(\boldsymbol{x},t) \in \mathbb{R}^c$ be the physical variable (*e.g.,* velocity, pressure), so that $u(\boldsymbol{x},t) \in \mathbb{R}^c$ represents the value of the field at spatial location $\boldsymbol{x} \in \Omega$ and time t. $c \in \mathbb{N}$ is the number of channels (for example, three velocity components and pressure). We collect N

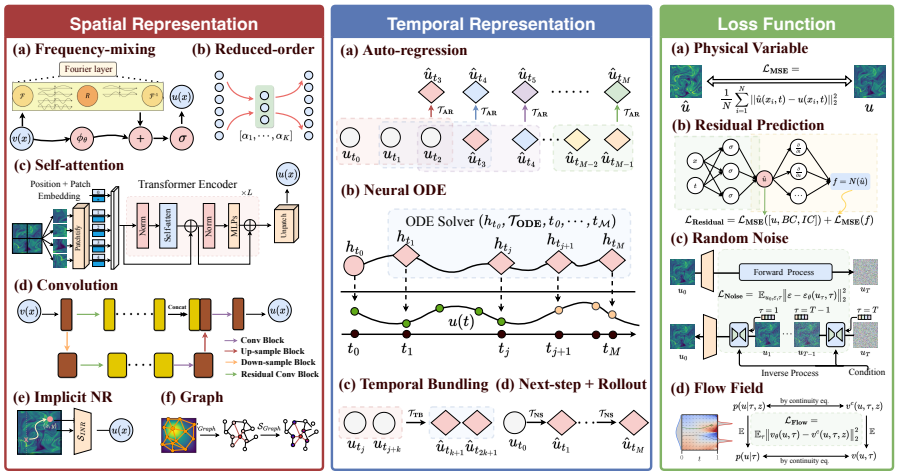


Figure 3: A schematic illustration of common approaches for each key module in data-driven neural PDE solvers. Note that self-attention is also applicable to temporal representation.

spatial samples $\{\boldsymbol{x}_i\}_{i=1}^N \subset \Omega$ (*e.g.,* grid points, mesh nodes, or particles) and denote $u(\cdot,t) = \big[u(\boldsymbol{x}_1,t),\, u(\boldsymbol{x}_2,t),\, \ldots,\, u(\boldsymbol{x}_N,t)\big]^\top \in \mathbb{R}^{N\times c}$. Spatial representation is any map that encodes the continuous field $u(\boldsymbol{x},t)$ at time t into a finite-dimensional feature z as $\mathcal{S}_\theta : \big\{u(\cdot,t)\big\} \mapsto z \in \mathbb{R}^D$. Here, we consider the commonly used spatial representation paradigms.

Fourier. It transforms the spatial field to the spectral domain using frequency transform (Li et al.; 2023a; Zhang et al., 2025) (*e.g.,* Discrete Fourier Transform), yielding spectral coefficients $\hat{u}(\mathbf{k},t) = \mathcal{F}(u)$, where $\mathbf{k} = (k_1,\ldots,k_d)$ denotes the discrete frequency. A learnable spectral filter $\phi_\theta(\mathbf{k})$ is applied by element-wise multiplication to the Fourier coefficients $\hat{u}(\mathbf{k},t) = \mathcal{F}(u(\cdot,t))$, thereby efficiently capturing long-range correlations via global mixing in physical space while decoupling spectral modes to learn coherent structures. $\mathcal{S}_{\text{Fourier}}(u(\cdot,t)) = \mathcal{F}^{-1}(\phi_\theta \odot \mathcal{F}(u(\cdot,t))) \in \mathbb{R}^{N\times c}$.
Self-attention. It allows each spatial location to attend to all others by computing weighted combinations of features based on learned similarity scores (Wu et al., 2024a; Geneva et al., 2022), thereby capturing global interactions adaptively. Given learnable projections $\mathbf{W}_Q, \mathbf{W}_K \in \mathbb{R}^{c\times d_k}$ and $\mathbf{W}_V \in \mathbb{R}^{c\times d_v}$: $\mathcal{S}_{\text{SA}}\left(u\left(\cdot,t\right)\right) = \text{softmax}\big(\frac{1}{\sqrt{d_k}}\left(u(\cdot,t)\,\mathbf{W}_Q\right)\left(u(\cdot,t)\,\mathbf{W}_K\right)^\top\big)\left(u(\cdot,t)\,\mathbf{W}_V\right) \in \mathbb{R}^{N\times d_v}$. **Convolution.** Convolution (Raonic et al., 2023) aggregates information from a local neighborhood using a shared kernel, leveraging spatial locality and translation equivariance. Let $\mathcal{H} \subset \mathbb{Z}^d$ be the set of stencil offsets. Let $c_{\text{in}} = c$ and c_{out} be the number of input and output channels. For each offset $h \in \mathcal{H}$, the kernel weight is $\mathbf{W}_h \in \mathbb{R}^{c_{\text{out}}\times c_{\text{in}}}$. Then, for each spatial index i: $\mathcal{S}_{\text{Conv}}(u(\cdot,t))_i = \sum_{h\in\mathcal{H}} \mathbf{W}_h\; u\big(\boldsymbol{x}_{i+h},t\big) \in \mathbb{R}^{c_{\text{out}}}$. **Graph.** Graph convolutions (Brandstetter et al., 2022; Lino et al., 2022; Wang et al., 2024b) generalize standard convolutional filtering to irregular domains by aggregating neighbor features weighted by graph connectivity, enabling modeling on meshes or point clouds. A typical form is: $\mathcal{S}_{\text{Graph}}(u(\cdot,t)) = \text{softmax}(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}\,u(\cdot,t)\,W) \in \mathbb{R}^{N\times D}$, where A is the binary adjacency, $\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}$ symmetrically normalizes the adjacency to control spectral properties, and $\sigma(\cdot)$ is a pointwise nonlinearity. **Reduced-Order Modeling (ROM).** Typical work like Proper Orthogonal Decomposition (POD) (Rojas et al., 2021; Wentland et al., 2023) approximates $u(\cdot,t)$ by projecting onto the K leading modes $\{\phi_k\}$ found via the covariance operator $C(\boldsymbol{x},\boldsymbol{x}') = \frac{1}{M}\sum_{i=1}^M u(\boldsymbol{x},t_i)\,u(\boldsymbol{x}',t_i)$, and solving $\int_\Omega C(\boldsymbol{x},\boldsymbol{x}')\phi_k(\boldsymbol{x}')\,\mathrm{d}\boldsymbol{x}' = \lambda_k\phi_k(\boldsymbol{x})$ under the inner product $\langle f,g\rangle = \int_\Omega f(\boldsymbol{x})g(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}$. The projection coefficients $\alpha_k(t) = \langle u(\cdot,t),\phi_k\rangle$ give $\mathcal{S}_{\text{ROM}}(u(\cdot,t)) = [\alpha_1(t),\ldots,\alpha_K(t)]^\top \in \mathbb{R}^K$, where M is the number of snapshots, N the spatial samples, and $K \ll N$ the retained modes. **Implicit Neural Representation.** It yields a continuous, differentiable approximation (Chen et al., 2023) of the field with sub-grid resolution and compact storage. The continuous field $u(\cdot,t)$ is modeled by implementing a coordinate-to-value mapping with sampled 2D coordinates $\{(x_i,y_i)\}_{i=1}^N$ as $\mathcal{S}_{\text{INR}}\big(u(\cdot,t)\big) = \arg\min_{\theta\in\mathbb{R}^D}\frac{1}{N}\sum_{i=1}^N\big\|f_\theta((x_i,y_i)) - u((x_i,y_i)),t)\big\|^2$.

4.2 Temporal Representation

Temporal representation encodes the evolution of the fluid field over time, providing essential features for predicting the future state of the flow field. Following the definition in Section 4.1, a

discrete sequence of M successive snapshots $\{u(t_j)\}_{j=1}^{M} \subset \mathbb{R}^{N\times c}$ are learned with a mapping $\mathcal{T}_\theta\big(\{u(t_j)\}_{j=1}^{M}\big) = h \in \mathbb{R}^D$, where θ parameterizes the encoder and h is a D-dimensional feature.

Autoregression. Autoregressive approaches (Li et al., 2021a; 2020) decompose the sequence model into successive predictions using a sliding history window of length k. If the input window at step j is $\big[u(t_{j-k}), u(t_{j-k+1}), \ldots, u(t_{j-1})\big] \in \mathbb{R}^{k\times N\times c}$. Then for j from $k+1$ to M, $\hat{u}(t_{j+1}) = \mathcal{T}_{\mathrm{AR}}\big(\big[u(t_{j-k+1}), \ldots, u(t_{j-1}), \hat{u}(t_j)\big]\big)$, so that each new prediction is appended to the window while the oldest state is discarded. **Next-Step.** Unlike the continuous prediction used in autoregression, next-step prediction (Cao et al., 2023b; Fortunato et al., 2022) predicts only a single step ahead based on the most recent state and is not affected by teacher forcing. It aims to generate long-horizon forecasts as $\hat{u}(t_j) = \mathcal{T}_{\mathrm{Next}}\big(u(t_{j-1})\big), j = \{1, \ldots, M\}$. **Temporal Bundling.** It is inspired by MP-PDE (Brandstetter et al., 2022), which stacks a fixed window of k frames and applies a feed-forward or convolutional encoder to predict k future steps. Thus a new vector $\mathbf{d} = \mathcal{T}_{\mathrm{TB}}(\mathbf{d}^0) = (\mathbf{d}^1, \cdots, \mathbf{d}^k)$ is used to update the solution $u\big(t_{k+j}\big) = u\big(t_k\big) + \big(t_{k+j} - t_k\big)\,\mathbf{d}^j, j = \{1, \ldots, k\}$. **Self-attention.** It parallels the spatial representation mechanism; however, whereas attention scores are originally computed over spatial patches, they are here computed along the temporal dimension. **Neural ODE.** Neural ordinary differential equations (Chen et al., 2018; Sun et al., 2024) provide a continuous-time framework for modeling temporal evolution in a latent space (Han et al., 2024; Wang et al., 2025b). We define a hidden state $h(t) \in \mathbb{R}^D$ that evolves according to the parameterized dynamics: $\frac{\mathrm{d}h}{\mathrm{d}t} = \mathcal{T}_{\mathrm{ODE}}\big(h(t), u(t_{0:t-1}), t\big), h(t_0) = \zeta(u(t_0)), u(t) = \xi(h(t))$, where $\zeta(\cdot)$ and $\xi(\cdot)$ are parameterized to encode the dynamics of the hidden state. The sequence of latent states $\{h(t_j)\}_{j=1}^{M}$ is obtained by solving ODE with a numeric integrator, and the final state is $h(t_1), \cdots, h(t_j) = \mathrm{ODESolver}(\mathcal{T}_{\mathrm{ODE}}, h(t_0), (t_0, \cdots, t_j))$. This continuous formulation naturally captures irregular time sampling and allows for arbitrary-horizon forecasting.

4.3 Loss Function

Physical Variable Prediction Loss. The mean squared error (MSE) (Cao et al., 2023a; Hao et al., 2024; Choubineh et al., 2023; Fortunato et al., 2022) between predicted and ground truth fields is commonly used in neural solvers given by $\mathcal{L}_{\mathrm{MSE}} = \frac{1}{N}\sum_{i=1}^{N}\big\|\hat{u}(\boldsymbol{x}_i, t) - u(\boldsymbol{x}_i, t)\big\|_2^2$. **Diffusion Denoising Loss.** Under the diffusion framework (Ho et al., 2020; Song et al., 2020), one perturbs the true field u_0 by a noise schedule $u_\tau = \sqrt{\bar{\alpha}_\tau}\,u_0 + \sqrt{1-\bar{\alpha}_\tau}\,\varepsilon, \varepsilon \sim \mathcal{N}(0, I)$, and trains a neural network ε_θ to estimate the added noise. The corresponding loss is $\mathcal{L}_{\mathrm{Noise}} = \mathbb{E}_{u_0,\varepsilon,\tau}\big\|\varepsilon - \varepsilon_\theta(u_\tau, \tau)\big\|_2^2$. This approach captures the full distribution of flow fields, enabling stochastic sampling and modeling of complex outcomes. **Flow Matching Loss.** Flow matching (Lipman et al., 2022) directly learns the instantaneous vector field by minimizing the discrepancy between a predicted velocity and a numerical approximation of the true flow with a simple (invertible) affine map ψ as $\mathcal{L}_{\mathrm{Flow}} = \mathbb{E}_{u,\tau}\Big\|v_\theta\big(\psi_\tau(u_0), \tau\big) - \frac{d}{d\tau}\psi_\tau(u_0)\Big\|_2^2$. $\psi_\tau(u)$ can be $\mu_\tau(u_1) + \sigma_\tau(u_1)u, \tau \in [0, 1]$. By training v_θ to match true dynamics, this loss facilitates stable multi-step and continuous forecasting, often producing sharper trajectories. **Physics-Informed Residual Loss.** Physics-informed neural networks (Raissi et al., 2019; Wang & Zhong, 2024) enforce the governing PDE by penalizing the residual of its differential operator. Denoting the learned field u_θ, the residual is $\mathcal{R}[u_\theta](x, t) = \partial_t u_\theta + \mathcal{N}(u_\theta) - g(x, t)$, and the loss reads $\mathcal{L}_{\mathrm{Residual}} = \mathbb{E}_{x,t}\big\|\mathcal{R}[u_\theta](x, t)\big\|_2^2$. Here $\mathcal{N}$ represents spatial operators (*e.g.*, advection) and g is any source term. It enforces exact satisfaction of physical laws in the continuous limit and often generalizes well beyond training data.

5 Experiments

5.1 Experimental Setup

Evaluation Metrics. To ensure a comprehensive and objective evaluation, we employ a diverse suite of metrics. First, we assess the global performance using the root-mean-squared-error (RMSE) and its normalized version (nRMSE). However, these measures do not capture local performance nuances. We further incorporate additional metrics that focus on specific failure modes, the RMSE in Fourier space (fRMSE) evaluated separately in low, middle, and high-frequency regions. Moreover, we extend our assessment to include efficiency evaluations, such as computational time cost and memory consumption. More details can be seen in Appendix H.

Experimental Setting. We create a brand-new codebase covering all of the decompositions. We use PyTorch to implement experiments on 8 × NVIDIA A6000 GPUs. Each dataset is by default split into 80% for training, 10% for testing, and 10% for validation, with a default spatial resolution

Model	Param	Compressible N-S RMSE↓	Compressible N-S nRMSE↓	Compressible N-S fRMSE↓	Diffusion-Reaction RMSE↓	Diffusion-Reaction nRMSE↓	Diffusion-Reaction fRMSE↓	Kolmogorov Flow RMSE↓	Kolmogorov Flow nRMSE↓	Kolmogorov Flow fRMSE↓	Efficiency Mem	Efficiency GFLOPs
$\mathcal{X}$ + Next-step + Physical variable												
Graph rep	18.6M	0.2648	0.3013	0.0097	0.0128	0.0251	0.0011	0.0709	0.0240	0.0044	1.17G	79.38
ROM	89.0M	0.1043	0.1542	0.0088	0.0523	0.0208	0.0019	0.0151	0.0151	0.0012	1.52G	81.96
Fourier	62.0M	0.0779	0.1019	0.0085	0.0077	0.0150	0.0008	0.0049	0.0049	0.0005	1.67G	79.46
Self-attention	76.2M	0.0589	0.0714	0.0042	0.0065	0.0150	0.0008	0.0032	0.0031	0.0002	1.41G	82.62
Convolution	64.8M	0.1325	0.1528	0.0074	0.0123	0.0274	0.0017	0.0675	0.0227	0.0041	1.39G	80.60
$\mathcal{X}$ + Next-step + Noise												
Fourier	62.0M	0.3387	0.3610	0.0962	0.0822	0.0948	0.0249	0.1161	0.1108	0.0424	2.10G	85.44
Self-attention	79.8M	0.3002	0.3354	0.0598	0.0302	0.0594	0.0051	0.0841	0.0816	0.0176	1.82G	82.59
Convolution	48.9M	0.1981	0.3348	0.0242	0.0437	0.0814	0.0089	0.1279	0.1155	0.0496	0.99G	74.92
ROM	75.9M	0.3518	0.3762	0.1006	0.0759	0.0962	0.0238	0.1345	0.1283	0.0559	1.07G	80.72
Self-attention representation + $\mathcal{Y}$ + Physical variable												
Self-attention	30.9M	0.2319	0.3094	0.0280	0.0443	0.1083	0.0067	0.0137	0.0145	0.0021	1.07G	91.37
Neural ODE	32.9M	0.3362	0.4137	0.0399	0.0135	0.0311	0.0014	0.0297	0.0307	0.0038	1.09G	87.52
Autoregression	24.9M	0.3943	0.4576	0.0478	0.0785	0.1398	0.0087	0.0182	0.0198	0.0026	1.24G	87.58
Next + Rollout	36.6M	0.2563	0.3177	0.0309	0.0454	0.1382	0.0041	0.0085	0.0086	0.0008	0.82G	79.15
Temporal Bund	36.3M	0.1357	0.1729	0.0122	0.0081	0.0205	0.0006	0.0049	0.0051	0.0005	1.71G	97.33
Fourier representation + $\mathcal{Y}$ + Physical variable												
Neural ODE	8.42M	0.2172	0.2884	0.0293	0.0124	0.0381	0.0010	0.0545	0.0589	0.0022	0.38G	29.16
Autoregression	18.0M	0.2032	0.2676	0.0291	0.0734	0.2089	0.0113	0.0849	0.0871	0.0095	1.69G	16.61
Next + Rollout	22.1M	0.1942	0.2581	0.0277	0.0165	0.0531	0.0014	0.0073	0.0081	0.0008	1.21G	20.45
Temporal Bund	17.4M	0.1842	0.2468	0.0187	0.0091	0.0193	0.0008	0.0072	0.0074	0.0007	1.14G	20.85
Self-attention representation + Next-step + $\mathcal{Z}$												
Physical var	24.7M	0.0997	0.0891	0.0094	0.0145	0.0277	0.0019	0.0082	0.0075	0.0009	0.72G	33.74
Noise	46.4M	0.3211	0.3367	0.0769	0.0310	0.0602	0.0048	0.0916	0.0859	0.0200	0.97G	35.46
Flow	42.6M	0.1232	0.1494	0.0118	0.0629	0.1039	0.0098	0.0609	0.0613	0.0050	1.15G	20.19
PDE Residual	29.6M	0.1370	0.1752	0.0118	0.1364	0.1644	0.0102	0.0302	0.0306	0.0021	0.67G	31.53

Table 3: The performance arises from three design factors: spatial representation ($\mathcal{X}$), temporal representation ($\mathcal{Y}$), and the loss function ($\mathcal{Z}$). Thus, we decouple them into modular designs and conduct comparisons under controlled variables on three datasets. MEM is the maximum memory.

of 128×128. We standardize the tuning process across all module families. Specifically, we fix the optimizer type, search ranges for learning rate and weight decay, and scheduler choices. Each result is tuned within the same grid-search budget, and we provide the full set of tuned configuration files, including random seeds, in the released codebase.

5.2 How Should We Define and Search for Optimal Neural Architectures?

We employ a modular decomposition combined with a controlled-variable methodology for comparison. To ensure fairness, all results are reported under approximately matched computational cost (*e.g.,* GFLOPs). Furthermore, all comparative experiments are performed on three representative 128×128 2D flow problems to ensure fairness.

To investigate the effect of spatial representations, we fix the prediction modes to $\mathcal{Y}$=`next-step`, and compare the performance of different methods. As shown in Table 3, the self-attention-based representation consistently achieves the best results across multiple tasks, followed by the Fourier-based representation. It is worth noting that due to the high computational cost of training convolutional models, we were unable to evaluate their performance under larger parameter budgets. Additionally, for the graph-based model, we apply sampling strategies to accommodate the specific data topology before training. More analysis details (*e.g.,* results on Stochastic N-S) are in Appendix F.

To investigate temporal evolution, we fix the spatial encoder to either $\mathcal{X}$=`Self-attention` or `Fourier`. As shown in Table 3, temporal bundling consistently achieves the best performance across tasks, while maintaining relatively low computational overhead. This highlights the importance of developing modules that explicitly capture the temporal dynamics of fluid evolution. Notably, neural ODEs approach the performance of temporal bundling on several tasks, suggesting a promising direction for future research. More generation analysis is in Section 5.5

Lastly, with respect to the loss function, physical variable prediction remains the mainstream approach. However, generative models such as diffusion-based and flow-based methods also demonstrate competitive performance, indicating promising directions for future investigation.

In conclusion, our results show that **(1)** self-attention is particularly effective for spatial encoding, and its performance can likely be improved further via techniques such as sparse attention, receptive field modulation, or transformer regularization. Meanwhile, frequency-based methods offer strong performance at minimal cost, highlighting the promise of embedding mathematical priors into neural solvers. **(2)** Simple yet effective temporal bundling strategies achieve strong performance under constrained compute budgets. This suggests a valuable research direction: accelerating temporal evolution modules like neural ODE through lightweight designs, potentially yielding a more favorable trade-off between predictive fidelity and efficiency in real-world applications.

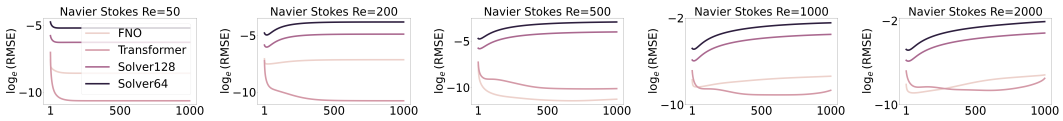


Figure 4: We compare neural solvers against traditional solver (*i.e.,* pseudo-spectral solver) operating at lower resolutions on N-S equation across five Reynolds numbers, spanning laminar to turbulent regimes. Solver x implies a pseudo-spectral solver operating at $x \times x$ resolution.

5.3 Can neural solvers truly replace traditional numerical solvers?

In terms of both performance and efficiency, the answer may be yes. Our experiments are conducted under carefully controlled and reproducible settings on incompressible N-S with results shown in Figure 4. Additional results are in Appendix E. Across most benchmarks, Transformer achieves substantially lower prediction error, especially at longer rollout steps. In terms of runtime, which is shown in Figure 5, Transformer outperforms both the high-fidelity simulator and coarse-grid solvers, delivering $10\times$ to $48\times$ speedups in the Burgers' equation experiments. Data generated at 256×256 is considered ground truth. All traditional methods, including those on coarser grids, employ adaptive time stepping governed by the Courant–Friedrichs–Lewy (CFL) condition to maintain numerical stability. On a coarser grid like 64×64, the solution can become less smooth and more prone to numerical instability. This can lead to smaller allowed Δt to maintain stability. As a result, even though each prediction with step size Δt is cheaper, the total number of steps required to reach the final simulation time can increase. We attribute the speedup of neural solvers to their ability to predict solutions with larger step size, allowing them to take fewer prediction steps than conventional numerical solvers. To examine this property in detail, we include extensive experiments in the Appendix E.1, analyzing the neural solver's performance across varying step sizes. The results indicate that, unlike numerical solvers where increasing Δt often causes severe numerical instabilities, neural solvers experience only minor performance degradation when trained to predict with a larger step size. They can still surpass numerical solvers operating at coarser resolution, enabling speedups of several hundred times. We refer the readers to Appendix J.2 for details of solvers and Appendix H for details of neural solvers.

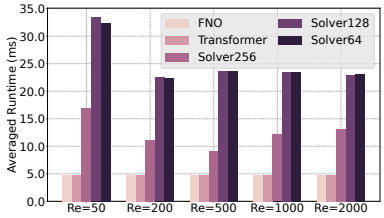


Figure 5: Runtime of neural solvers against numerical solver operating at lower resolutions on the incompressible N-S for predicting up to $T = \frac{1}{32}$. Solver x implies $x \times x$ resolution of pseudo-spectral solver.

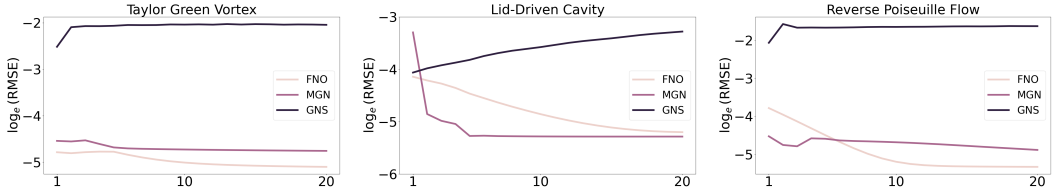


Figure 6: Comparison of rollout performance between FNO (grid data), MeshGraphNets (mesh data), and GNS (particle data). We transform particle and mesh predictions to grid data to evaluate.

5.4 Which discretization better supports their representative neural solvers?

To ensure a rigorous and fair comparison, we evaluate each solver on the discretization scheme it is specifically designed for: FNO (Li et al., 2021a) is trained on regular Eulerian grids, MeshGraphNets (MGN) (Pfaff et al., 2021) on unstructured meshes, and GNS (Sanchez et al., 2020) on particle-based Lagrangian representations. All experiments are conducted under matched grid resolutions, comparable computational budgets, and carefully tuned hyperparameters (see Appendix H) to isolate the effect of discretization rather than model capacity or training regime. Figure 6 reports the rollout MSE of the three solvers. Across all experiments, the Eulerian discretizations achieve significantly lower rollout MSE than the Lagrangian scheme.

In conclusion, we observe that models employing Eulerian schemes more accurately capture fundamental fluid dynamics. In contrast, Lagrangian representations must model fine-grained details, which impedes their ability to learn robust representations and renders them susceptible to error accumulation over extended rollout time steps. Their emphasis on simulating high-resolution local

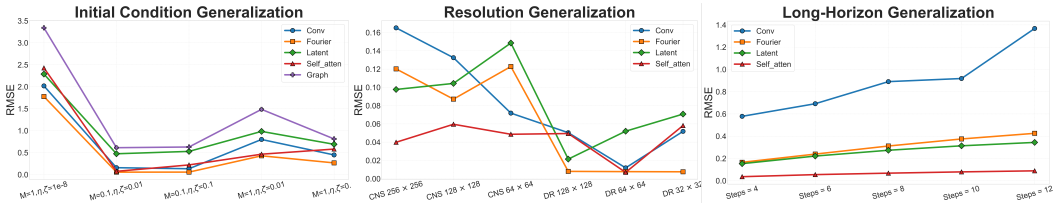


Figure 7: Extrapolation evaluation on zero-shot generalization across diverse initial conditions, resolution generalization across spatial scales, and long-horizon generalization. All evaluations are performed on the Compressible N-S system using different methods for next-step prediction.

interactions leads to severe error accumulation and propagation across particles over time. To mitigate this, future directions may involve hierarchical or multi-scale architectures, such as U-Net-based designs, that can dynamically balance fine-grained resolution with global structural coherence, thus improving both stability and generalization. We refer the readers to Appendix H for more details of model and training hyperparameters.

5.5 Whether knowledge from different systems can boost generalization?

Generalization on zero-shot initial conditions. We first pre-train all candidate models on Compressible N–S with a moderate Mach number $M = 0.1$, where both shear and bulk viscosities are fixed at $\eta = \zeta = 10^{-8}$. Then we evaluate the trained models directly on test data with OOD initial conditions, which exhibit qualitatively different flow structures and turbulence patterns. This setting is designed to mimic real-world scenarios where the governing dynamics remain the same, but the flow configurations encountered may deviate significantly from those seen during training.

As shown in the first panel of Figure 7, all models experience varying degrees of performance degradation under these OOD shifts, confirming the challenge of prediction across diverse initial conditions. Interestingly, models leveraging global spatial representations, such as Fourier-based operators and self-attention mechanisms, exhibit significantly better robustness. Their ability to capture long-range correlations and encode global flow structures appears to confer a clear advantage when extrapolating to turbulent regimes with unseen coherent structures.

Generalization on different resolutions. We also train models using various methods on both the Compressible N-S and Diffusion-Reaction subsets across different spatial resolutions, and evaluate them on their corresponding test resolutions. This controlled setup allows us to disentangle how each model family benefits from increasing grid fidelity and whether their inductive biases are well-aligned with the underlying physics at different scales. As shown in the middle panel of Figure 7, self-attention and latent-based models benefit from higher-resolution training, leveraging richer spatial information to improve performance. In contrast, Fourier-based representations exhibit stronger performance at lower resolutions, suggesting superior inductive bias for coarse-scale generalization.

Generalization on long-horizon rollout. We also evaluate the long-horizon generalization ability of different methods under the setting $\mathcal{Z}$ = `variable`, by rolling out predictions over extended time steps. This experiment is designed to probe whether the learned representations can remain numerically stable and physically consistent when error accumulation becomes a dominant factor. As shown in the right panel of Figure 7, all methods experience some degree of error accumulation as the rollout length increases. However, the spatial representation based on self-attention remains the most stable, exhibiting the lowest performance degradation over time.

Additionally, models trained with a denoising loss and subsequently sampled exhibit substantially greater stability compared to deterministic models. Detailed comparison is in Appendix G.1.

Generalization on scaling ability. Please refer to Appendix G.2 for comparisons of scaling ability.

Qualitative study. Additional qualitative results and detailed visualization analyses are provided in Appendix K to further illustrate the model behavior and support our quantitative findings.

6 Conclusion

Taken together, findings of FD-Bench not only establish a rigorous and fair benchmark but also chart a forward-looking research agenda for the community. We envision that FD-Bench will serve as a catalyst for developing neural PDE solvers that are not only more accurate and efficient but also more robust and scalable, paving the way toward practical, real-time, and general-purpose simulators.

Reproducibility Statement

We have made significant efforts to ensure the reproducibility of FD-Bench. All datasets used in this study are either collected from public benchmarks (Compressible N-S, Diffusion-Reaction, Kolmogorov Flow) or generated using open-source pseudo-spectral solvers under well-specified parameter settings; detailed dataset curation procedures are provided in Section 3 and Appendix J. You can also find it in the README of https://anonymous.4open.science/r/FD-Bench-15BC. Our benchmark codebase is fully modular and publicly released in an anonymous repository https://anonymous.4open.science/r/FD-Bench-15BC, including scripts for data preprocessing, training, evaluation, and hyperparameter tuning. To ensure fair comparisons, we standardize grid-search budgets for optimizers, learning rates, weight decay, and schedulers across all module families, as detailed in Section 5 and Appendix H. Random seeds for all experiments are fixed and included in the configuration files. Complete results, including additional ablation studies, discretization comparisons, and generalization analyses, are reported in the main paper (Sections 5) and expanded in Appendices D-K. Together, these materials are intended to enable full replication of our experiments and facilitate future extensions of FD-Bench.

References

Benedikt Alkin, Andreas Fürst, Simon Lucas Schmid, Lukas Gruber, Markus Holzleitner, and Johannes Brandstetter. Universal physics transformers: A framework for efficiently scaling neural operators. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=oUXiNX5KRm.

Jan-Hendrik Bastek, WaiChing Sun, and Dennis Kochmann. Physics-informed diffusion models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=tpYeermigp.

Kaifeng Bi, Lingxi Xie, Hengheng Zhang, Xin Chen, Xiaotao Gu, and Qi Tian. Accurate medium-range global weather forecasting with 3d neural networks. *Nature*, 619(7970):533–538, 2023.

Suresh Bishnoi, Ravinder Bhattoo, Jayadeva Jayadeva, Sayan Ranu, and N M Anoop Krishnan. Enhancing the inductive biases of graph neural ODE for modeling physical systems. In *The Eleventh International Conference on Learning Representations*, 2023.

Oussama Boussif, Yoshua Bengio, Loubna Benabbou, and Dan Assouline. Magnet: Mesh agnostic neural pde solver. *Advances in Neural Information Processing Systems*, 35:31972–31985, 2022.

Johannes Brandstetter. Envisioning better benchmarks for machine learning pde solvers. *Nature Machine Intelligence*, 7(1):2–3, 2025.

Johannes Brandstetter, Daniel E. Worrall, and Max Welling. Message passing neural PDE solvers. In *International Conference on Learning Representations*, 2022.

Johannes Brandstetter, Rianne van den Berg, et al. Clifford neural layers for PDE modeling. In *The Eleventh International Conference on Learning Representations*, 2023.

Andrey Bryutkin, Jiahao Huang, Zhongying Deng, Guang Yang, Carola-Bibiane Schönlieb, and Angelica I Aviles-Rivero. HAMLET: Graph transformer neural operator for partial differential equations. In *International Conference on Machine Learning*, 2024.

Claudio Canuto, M. Yousuff Hussaini, Alfio Quarteroni, and Thomas A. Zang. *Spectral Methods in Fluid Dynamics*. Springer-Verlag, Berlin, Heidelberg, 1988.

Qianying Cao, Somdatta Goswami, and George Em Karniadakis. Lno: Laplace neural operator for solving differential equations. *Arxiv*, 2023a.

Shuhao Cao. Choose a transformer: Fourier or galerkin. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=ssohLcmn4-r.

Shuhao Cao, Francesco Brarda, Ruipeng Li, and Yuanzhe Xi. Spectral-refiner: Accurate fine-tuning of spatiotemporal fourier neural operator for turbulent flows. In *The Thirteenth International Conference on Learning Representations*, 2025.

Yadi Cao and Ri Li. A liquid plug moving in an annular pipe—flow analysis. *Physics of Fluids*, 30 (9), 2018.

Yadi Cao, Xuan Gao, and Ri Li. A liquid plug moving in an annular pipe–heat transfer analysis. *International Journal of Heat and Mass Transfer*, 139:1065–1076, 2019.

Yadi Cao, Menglei Chai, Minchen Li, and Chenfanfu Jiang. Efficient learning of mesh-based physical simulation with bi-stride multi-scale graph neural network. In *International Conference on Machine Learning*, pp. 3541–3558. PMLR, 2023b.

Honglin Chen, Rundi Wu, Eitan Grinspun, Changxi Zheng, and Peter Yichen Chen. Implicit neural spatial representations for time-dependent pdes. In *International Conference on Machine Learning*, pp. 5162–5177. PMLR, 2023.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Zhao Chen, Yang Liu, and Hao Sun. Physics-informed learning of governing equations from scarce data. *Nature communications*, 12(1):6136, 2021.

Chaoran Cheng, Boran Han, Danielle C Maddix, Abdul Fatir Ansari, Andrew Stuart, Michael W Mahoney, and Yuyang Wang. Gradient-free generation for hard-constrained systems. *arXiv preprint arXiv:2412.01786*, 2024.

Abouzar Choubineh, Jie Chen, David A Wood, Frans Coenen, and Fei Ma. Fourier neural operator for fluid flow in small-shape 2d simulated porous media dataset. *Algorithms*, 16(1):24, 2023.

Zhiwen Deng, Jing Wang, Hongsheng Liu, Hairun Xie, BoKai Li, Miao Zhang, Tingmeng Jia, Yi Zhang, Zidong Wang, and Bin Dong. Prediction of transonic flow over supercritical airfoils using geometric-encoding and deep-learning strategies. *Physics of Fluids*, 35(7), 2023.

Meire Fortunato, Tobias Pfaff, Peter Wirnsberger, Alexander Pritzel, and Peter Battaglia. Multiscale meshgraphnets. *Arxiv*, 2022.

Nicholas Geneva et al. Transformers for modeling physical systems. *Neural Networks*, 146:272–289, 2022.

Paul Ghanem, Ahmet Demirkaya, Tales Imbiriba, Alireza Ramezani, Zachary Danziger, and Deniz Erdogmus. Learning physics informed neural odes with partial measurements. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 16799–16807, 2025.

R.A. Gingold and J.J. Monaghan. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society*, 181(3):375–389, 1977.

John Guibas, Morteza Mardani, Zongyi Li, Andrew Tao, Anima Anandkumar, and Bryan Catanzaro. Efficient token mixing for transformers via adaptive fourier neural operators. In *International conference on learning representations*, 2021.

Gaurav Gupta, Xiongye Xiao, and Paul Bogdan. Multiwavelet-based operator learning for differential equations. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=LZDiWaC9CGL.

Kaiqiao Han, Yi Yang, Zijie Huang, Xuan Kan, Ying Guo, Yang Yang, Lifang He, Liang Zhan, Yizhou Sun, Wei Wang, et al. Brainode: Dynamic brain signal analysis via graph-aided neural ordinary differential equations. In *2024 IEEE EMBS International Conference on Biomedical and Health Informatics (BHI)*, pp. 1–8. IEEE, 2024.

XU HAN, Han Gao, Tobias Pfaff, Jian-Xun Wang, and Liping Liu. Predicting physics in mesh-reduced space with temporal attention. In *International Conference on Learning Representations*, 2022.

Zhongkai Hao, Zhengyi Wang, Hang Su, Chengyang Ying, Yinpeng Dong, Songming Liu, Ze Cheng, Jian Song, and Jun Zhu. Gnot: A general neural operator transformer for operator learning. In *International Conference on Machine Learning*, pp. 12556–12569. PMLR, 2023a.

Zhongkai Hao, Jiachen Yao, Chang Su, Hang Su, Ziao Wang, Fanzhi Lu, Zeyu Xia, Yichi Zhang, Songming Liu, Lu Lu, et al. Pinnacle: A comprehensive benchmark of physics-informed neural networks for solving pdes. *Arxiv*, 2023b.

Zhongkai Hao, Chang Su, Songming Liu, Julius Berner, Chengyang Ying, Hang Su, Anima Anandkumar, Jian Song, and Jun Zhu. Dpot: Auto-regressive denoising operator transformer for large-scale pde pre-training. *ICML*, 2024.

Xiaodong He, Yinan Wang, and Juan Li. Flow completion network: Inferring the fluid dynamics from incomplete flow information using graph neural networks. *Physics of Fluids*, 34(8), 2022.

Jacob Helwig, Xuan Zhang, Cong Fu, Jerry Kurtin, Stephan Wojtowytsch, and Shuiwang Ji. Group equivariant fourier neural operators for partial differential equations. *ICML*, 2023.

Maximilian Herde, Bogdan Raonic, Tobias Rohner, Roger Käppeli, Roberto Molinaro, Emmanuel de Bézenac, and Siddhartha Mishra. Poseidon: Efficient foundation models for pdes. *Advances in Neural Information Processing Systems*, 37:72525–72624, 2024.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Jiahe Huang, Guandao Yang, Zichen Wang, and Jeong Joon Park. DiffusionPDE: Generative PDE-solving under partial observation. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024a. URL https://openreview.net/forum?id=z0I2SbjN0R.

Xiang Huang, Zhanhong Ye, Hongsheng Liu, Shi Ji, Zidong Wang, Kang Yang, Yang Li, Min Wang, Haotian Chu, Fan Yu, et al. Meta-auto-decoder for solving parametric partial differential equations. *Advances in Neural Information Processing Systems*, 35:23426–23438, 2022.

Zijie Huang, Yizhou Sun, and Wei Wang. Learning continuous system dynamics from irregularly-sampled partial observations. In *Advances in Neural Information Processing Systems*, 2020.

Zijie Huang, Wanjia Zhao, Jingdong Gao, Ziniu Hu, Xiao Luo, Yadi Cao, Yuanzhou Chen, Yizhou Sun, and Wei Wang. Physics-informed regularization for domain-agnostic dynamical system modeling, 2024b. URL https://arxiv.org/abs/2410.06366.

Jinsung Jeon, Hyundong Jin, Jonghyun Choi, Sanghyun Hong, Dongeun Lee, Kookjin Lee, and Noseong Park. Pac-fno: Parallel-structured all-component fourier neural operators for recognizing low-quality images. *Arxiv*, 2024.

Pengzhan Jin, Shuai Meng, and Lu Lu. Mionet: Learning multiple-input operators via tensor product. *SIAM Journal on Scientific Computing*, 44(6):A3490–A3514, 2022.

Xiaowei Jin, Shengze Cai, Hui Li, and George Em Karniadakis. Nsfnets (navier-stokes flow nets): Physics-informed neural networks for the incompressible navier-stokes equations. *Journal of Computational Physics*, 426:109951, 2021.

Gavin Kerrigan, Giosue Migliorini, and Padhraic Smyth. Functional flow matching. *arXiv preprint arXiv:2305.17209*, 2023.

G.A. Klaasen and W.C. Troy. Stationary wave solutions of a system of reaction-diffusion equations derived from the fitzhugh–nagumo equations. *SIAM Journal on Applied Mathematics*, 44(1): 96–110, 1984. doi: 10.1137/0144008.

Georg Kohl, Li-Wei Chen, and Nils Thuerey. Benchmarking autoregressive conditional diffusion models for turbulent flow simulation. *arXiv preprint arXiv:2309.01745*, 2023.

Katiana Kontolati, Somdatta Goswami, George Em Karniadakis, and Michael D Shields. Learning in latent spaces improves the predictive accuracy of deep neural operators. *Arxiv*, 2023.

Jae Yong Lee, SungWoong CHO, and Hyung Ju Hwang. HyperdeepONet: learning operator with complex target function space using the limited resources via hypernetwork. In *The Eleventh International Conference on Learning Representations*, 2023.

Sangseung Lee and Donghyun You. Data-driven prediction of unsteady flow over a circular cylinder using deep learning. *Journal of Fluid Mechanics*, 879:217–254, 2019.

Tianyi Li, Luca Biferale, Fabio Bonaccorso, Martino Andrea Scarpolini, and Michele Buzzicotti. Synthetic lagrangian turbulence by generative diffusion models. *Nature Machine Intelligence*, pp. 1–11, 2024a.

Zijie Li and Amir Barati Farimani. Graph neural network-accelerated lagrangian fluid simulation. *Computers & Graphics*, 103:201–211, 2022.

Zijie Li, Dule Shu, and Amir Barati Farimani. Scalable transformer for pde surrogate modeling. *Advances in Neural Information Processing Systems*, 36, 2024b.

Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, Anima Anandkumar, et al. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Andrew Stuart, Kaushik Bhattacharya, and Anima Anandkumar. Multipole graph neural operator for parametric partial differential equations. *Advances in Neural Information Processing Systems*, 33:6755–6766, 2020.

Zongyi Li, Nikola Borislavov Kovachki, et al. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021a.

Zongyi Li, Hongkai Zheng, Nikola Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-informed neural operator for learning partial differential equations. *Arxiv*, 2021b.

Zongyi Li, Daniel Zhengyu Huang, Burigede Liu, and Anima Anandkumar. Fourier neural operator with learned deformations for pdes on general geometries. *Journal of Machine Learning Research*, 24(388):1–26, 2023a.

Zongyi Li, Nikola Borislavov Kovachki, et al. Geometry-informed neural operator for large-scale 3d PDEs. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023b.

Guang Lin, Christian Moya, and Zecheng Zhang. B-deeponet: An enhanced bayesian deeponet for solving noisy parametric pdes using accelerated replica exchange sgld. *Journal of Computational Physics*, 473:111713, 2023.

Mario Lino, Stathi Fotiadis, Anil A Bharath, and Chris D Cantwell. Multi-scale rotation-equivariant graph neural networks for unsteady eulerian fluid dynamics. *Physics of Fluids*, 34(8), 2022.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Jinxian Liu, Ye Chen, Bingbing Ni, Wei Ren, Zhenbo Yu, and Xiaoyang Huang. Fast fluid simulation via dynamic multi-scale gridding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 1675–1682, 2023a.

Ning Liu, Siavash Jafarzadeh, and Yue Yu. Domain agnostic fourier neural operators. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023b.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature machine intelligence*, 3(3):218–229, 2021.

Xiao Luo, Haixin Wang, Zijie Huang, Huiyu Jiang, Abhijeet Sadashiv Gangan, Song Jiang, and Yizhou Sun. Care: Modeling interacting dynamics under temporal environmental variation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023a.

Xiao Luo, Jingyang Yuan, Zijie Huang, Huiyu Jiang, Yifang Qin, Wei Ju, Ming Zhang, and Yizhou Sun. HOPE: High-order graph ODE for modeling interacting dynamics. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 23124–23139. PMLR, 23–29 Jul 2023b. URL https://proceedings.mlr.press/v202/luo23f.html.

Xiao Luo, Yiyang Gu, Huiyu Jiang, Hang Zhou, Jinsheng Huang, Wei Ju, Zhiping Xiao, Ming Zhang, and Yizhou Sun. Pgode: Towards high-quality system dynamics modeling, 2024. URL https://arxiv.org/abs/2311.06554.

Yining Luo, Yingfa Chen, and Zhen Zhang. Cfdbench: A comprehensive benchmark for machine learning methods in fluid dynamics. *Arxiv*, 2023c.

Andrew J Majda, Andrea L Bertozzi, and A Ogawa. Vorticity and incompressible flow. cambridge texts in applied mathematics. *Appl. Mech. Rev.*, 55(4):B77–B78, 2002.

Nick McGreivy et al. Weak baselines and reporting biases lead to overoptimism in machine learning for fluid-related partial differential equations. *Nature Machine Intelligence*, 6(10):1256–1269, 2024.

R. Mikulevicius and B. L. Rozovskii. Stochastic navier–stokes equations for turbulent flows. *SIAM Journal on Mathematical Analysis*, 35(5):1250–1310, 2004. doi: 10.1137/S0036141002409167. URL https://doi.org/10.1137/S0036141002409167.

F. Moukalled, L. Mangani, and M. Darwish. *The Finite Volume Method in Computational Fluid Dynamics*. Springer, 1 edition, 2016. doi: 10.1007/978-3-319-16874-6.

Bilal Mufti, Anindya Bhaduri, Sayan Ghosh, Liping Wang, and Dimitri N Mavris. Shock wave prediction in transonic flow fields using domain-informed probabilistic deep learning. *Physics of Fluids*, 36(1), 2024.

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

Ruben Ohana, Michael McCabe, Lucas Meyer, Rudy Morel, Fruzsina Agocs, Miguel Beneitez, Marsha Berger, Blakesly Burkhart, Stuart Dalziel, Drummond Fielding, et al. The well: a large-scale collection of diverse physics simulations for machine learning. *Advances in Neural Information Processing Systems*, 37:44989–45037, 2024.

Michael O'Connell, Guanya Shi, Xichen Shi, et al. Neural-fly enables rapid learning for agile flight in strong winds. *Science Robotics*, 7(66), 2022.

Jaideep Pathak, Shashank Subramanian, et al. Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators. *Arxiv*, 2022.

Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter Battaglia. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations*, 2021.

Yuan Qiu, Nolan Bridges, and Peng Chen. Derivative-enhanced deep operator network. *Advances in Neural Information Processing Systems*, 37:20945–20981, 2024.

Md Ashiqur Rahman, Zachary E Ross, and Kamyar Azizzadenesheli. U-NO: U-shaped neural operators. *Transactions on Machine Learning Research*, 2023.

Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

Ellery Rajagopal, Anantha NS Babu, et al. Evaluation of deep neural operator models toward ocean forecasting. In *OCEANS*, pp. 1–9. IEEE, 2023.

Chengping Rao, Pu Ren, Qi Wang, Oral Buyukozturk, Hao Sun, and Yang Liu. Encoding physics to learn reaction–diffusion processes. *Nature Machine Intelligence*, 5(7):765–779, 2023.

Bogdan Raonic, Roberto Molinaro, Tobias Rohner, Siddhartha Mishra, and Emmanuel de Bezenac. Convolutional neural operators. In *ICLR 2023 Workshop on Physics for Machine Learning*, 2023.

Carlos JG Rojas, Andreas Dengel, and Mateus Dias Ribeiro. Reduced-order model for fluid flows via neural ordinary differential equations. *arXiv preprint arXiv:2102.02248*, 2021.

Alexander Rudikov, Vladimir Fanaskov, et al. Neural operators meet conjugate gradients: The FCG-NO method for efficient PDE solving. In *Forty-first International Conference on Machine Learning*, 2024.

Cristopher Salvi, Maud Lemercier, and Andris Gerasimovics. Neural stochastic pdes: Resolution-invariant learning of continuous spatiotemporal dynamics, 2022. URL https://arxiv.org/abs/2110.10249.

Alvaro Sanchez, Jonathan Godwin, et al. Learning to simulate complex physics with graph networks. In *International conference on machine learning*, pp. 8459–8468. PMLR, 2020.

Jacob Seidman, Georgios Kissas, Paris Perdikaris, and George J Pappas. Nomad: Nonlinear manifold decoders for operator learning. *Advances in Neural Information Processing Systems*, 35:5601–5613, 2022.

Yidi Shao, Chen Change Loy, and Bo Dai. Transformer with implicit edges for particle-based physics simulation. In *European Conference on Computer Vision*, pp. 549–564. Springer, 2022.

Hong Shen Wong, Wei Xuan Chan, Bing Huan Li, and Choon Hwai Yap. Multiple case physics-informed neural network for biomedical tube flows. *Arxiv*, 2023.

Aleksei Sholokhov, Yuying Liu, Hassan Mansour, and Saleh Nabi. Physics-informed neural ode (pinode): embedding physics into models using collocation points. *Scientific Reports*, 13(1):10166, 2023.

Aliaksandra Shysheya, Cristiana Diaconu, Federico Bergamin, Paris Perdikaris, José Miguel Hernández-Lobato, Richard E. Turner, and Emile Mathieu. On conditional diffusion models for PDE simulations. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=nQl8EjyMzh.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

Jos Stam. Stable fluids. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, pp. 121–128, USA, 1999. ACM Press/Addison-Wesley Publishing Co. ISBN 0201485605. doi: 10.1145/311535.311548. URL https://doi.org/10.1145/311535.311548.

Shankar Subramaniam. Lagrangian–eulerian methods for multiphase flows. *Progress in Energy and Combustion Science*, 39(2-3):215–245, 2013.

Fang Sun, Zijie Huang, Haixin Wang, Yadi Cao, Xiao Luo, Wei Wang, and Yizhou Sun. Graph fourier neural odes: Bridging spatial and temporal multiscales in molecular dynamics. *arXiv preprint arXiv:2411.01600*, 2024.

Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Daniel MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. Pdebench: An extensive benchmark for scientific machine learning. *Advances in Neural Information Processing Systems*, 35:1596–1611, 2022.

Maryam Toloubidokhti, Yubo Ye, Ryan Missel, Xiajun Jiang, Nilesh Kumar, Ruby Shrestha, and Linwei Wang. Dats: Difficulty-aware task sampler for meta-learning physics-informed neural networks. In *The Twelfth International Conference on Learning Representations*, 2023.

F Arend Torres, Marcello Massimo Negri, Marco Inversi, Jonathan Aellen, and Volker Roth. Lagrangian flow networks for conservation laws. *Arxiv*, 2023.

A. Tran, A. Mathews, et al. Factorized fourier neural operators. In *ICLR*, 2023.

Mario Lino Valencia, Tobias Pfaff, and Nils Thuerey. Learning distributions of complex fluid simulations with diffusion graph networks. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=uKZdlihDDn.

Simone Venturi et al. Svd perspectives for augmenting deeponet flexibility and interpretability. *Computer Methods in Applied Mechanics and Engineering*, 403:115718, 2023.

Paulien HM Voorter, Walter H Backes, et al. Improving microstructural integrity, interstitial fluid, and blood microcirculation images from multi-b-value diffusion mri using physics-informed neural networks in cerebrovascular disease. *Magnetic Resonance in Medicine*, 2023.

Haixin Wang, Yadi Cao, Zijie Huang, Yuxuan Liu, Peiyan Hu, Xiao Luo, Zezheng Song, Wanjia Zhao, Jilin Liu, Jinan Sun, et al. Recent advances on machine learning for computational fluid dynamics: A survey. *arXiv preprint arXiv:2408.12171*, 2024a.

Haixin Wang, LI Jiaxin, Anubhav Dwivedi, Kentaro Hara, and Tailin Wu. BENO: Boundary-embedded neural operators for elliptic PDEs. In *The Twelfth International Conference on Learning Representations*, 2024b.

Rui Wang, Robin Walters, and Rose Yu. Incorporating symmetry into deep dynamics models for improved generalization. *Arxiv 2020*.

Rui Wang, Karthik Kashinath, Mustafa Mustafa, Adrian Albert, and Rose Yu. Towards physics-informed deep learning for turbulent flow prediction. In *KDD*, pp. 1457–1466, 2020.

Rui Wang et al. Approximately equivariant networks for imperfectly symmetric dynamics. In *International Conference on Machine Learning*, 2022.

Sifan Wang, Hanwen Wang, and Paris Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed deeponets. *Science advances*, 7(40), 2021.

Sifan Wang, Zehao Dou, Tong-Rui Liu, and Lu Lu. Fundiff: Diffusion models over function spaces for physics-informed generative modeling. *arXiv preprint arXiv:2506.07902*, 2025a.

Xiaoda Wang, Yuji Zhao, Kaiqiao Han, Xiao Luo, Sanne van Rooij, Jennifer Stevens, Lifang He, Liang Zhan, Yizhou Sun, Wei Wang, et al. Conditional neural ode for longitudinal parkinson's disease progression forecasting. In *Abstract in the Organization for Human Brain Mapping Annual Meeting*, 2025b.

Yifan Wang and Linlin Zhong. Nas-pinn: neural architecture search-guided physics-informed neural network for solving pdes. *Journal of Computational Physics*, 496:112603, 2024.

Ping Wei, Menghan Liu, Jianhuan Cen, Ziyang Zhou, Liao Chen, and Qingsong Zou. Pdenneval: A comprehensive evaluation of neural network methods for solving pdes. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pp. 5181–5189, 2024.

Gege Wen, Zongyi Li, Kamyar Azizzadenesheli, Anima Anandkumar, and Sally M Benson. U-fno—an enhanced fourier neural operator-based deep-learning model for multiphase flow. *Advances in Water Resources*, 163:104180, 2022.

Christopher R Wentland, Karthik Duraisamy, and Cheng Huang. Scalable projection-based reduced-order models for large multiscale fluid systems. *AIAA Journal*, 61(10):4499–4523, 2023.

Henning Wessels, Christian Weißenfels, and Peter Wriggers. The neural particle method–an updated lagrangian physics informed neural network for computational fluid dynamics. *Computer Methods in Applied Mechanics and Engineering*, 368:113127, 2020.

Haixu Wu, Tengge Hu, Huakun Luo, Jianmin Wang, and Mingsheng Long. Solving high-dimensional pdes with latent spectral models. In *International Conference on Machine Learning*, 2023a.

Haixu Wu, Huakun Luo, et al. Transolver: A fast transformer solver for PDEs on general geometries. In *Forty-first International Conference on Machine Learning*, 2024a.

Hao Wu, Changhu Wang, Fan Xu, Jinbao Xue, Chong Chen, Xian-Sheng Hua, and Xiao Luo. Pure: Prompt evolution with graph ode for out-of-distribution fluid dynamics modeling. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 104965–104994. Curran Associates, Inc., 2024b. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/bd92debabb5e6eb881ef81d88e0f22ae-Paper-Conference.pdf.

Hao Wu, Huiyuan Wang, Kun Wang, Weiyan Wang, ChanganYe, Yangyu Tao, Chong Chen, Xian-Sheng Hua, and Xiao Luo. Prometheus: Out-of-distribution fluid dynamics modeling with disentangled graph ODE. In *Forty-first International Conference on Machine Learning*, 2024c. URL https://openreview.net/forum?id=JsPvL6ExK8.

Hao Wu, Fan Xu, Yifan Duan, Ziwei Niu, Weiyan Wang, Gaofeng Lu, Kun Wang, Yuxuan Liang, and Yang Wang. Spatio-temporal fluid dynamics modeling via physical-awareness and parameter diffusion guidance. *arXiv preprint arXiv:2403.13850*, 2024d.

Tailin Wu, Takashi Maruyama, Qingqing Zhao, Gordon Wetzstein, and Jure Leskovec. Learning controllable adaptive simulation for multi-resolution physics. In *The Eleventh International Conference on Learning Representations*, 2023b.

Xiongye Xiao, Defu Cao, Ruochen Yang, Gaurav Gupta, Gengshuo Liu, Chenzhong Yin, Radu Balan, and Paul Bogdan. Coupled multiwavelet operator learning for coupled differential equations. In *The Eleventh International Conference on Learning Representations*, 2022.

Zipeng Xiao, Zhongkai Hao, Bokai Lin, Zhijie Deng, and Hang Su. Improved operator learning by orthogonal attention. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 54288–54299. PMLR, 21–27 Jul 2024a. URL https://proceedings.mlr.press/v235/xiao24c.html.

Zipeng Xiao, Siqi Kou, Hao Zhongkai, Bokai Lin, and Zhijie Deng. Amortized fourier neural operators. *Advances in Neural Information Processing Systems*, 37:115001–115020, 2024b.

Wei Xiong, Xiaomeng Huang, Ziyang Zhang, Ruixuan Deng, Pei Sun, and Yang Tian. Koopman neural operator as a mesh-free solver of non-linear partial differential equations. *Arxiv*, 2023.

Liu Yang, Siting Liu, Tingwei Meng, and Stanley J. Osher. In-context operator learning with data prompts for differential equation problems. *Proceedings of the National Academy of Sciences*, 120 (39), 2023.

Jiachen Yao, Abbas Mammadov, Julius Berner, Gavin Kerrigan, Jong Chul Ye, Kamyar Azizzadenesheli, and Anima Anandkumar. Guided diffusion sampling on function spaces with applications to pdes. *arXiv preprint arXiv:2505.17004*, 2025.

Minglang Yin, Ehsan Ban, Bruno V Rego, Enrui Zhang, Cristina Cavinato, Jay D Humphrey, and George Em Karniadakis. Simulating progressive intramural damage leading to aortic dissection using deeponet: an operator–regression neural network. *Journal of the Royal Society Interface*, 19 (187):20210670, 2022.

Yuan Yin, Matthieu Kirchmeyer, Jean-Yves Franceschi, Alain Rakotomamonjy, and patrick gallinari. Continuous PDE dynamics forecasting with implicit neural representations. In *The Eleventh International Conference on Learning Representations*, 2023.

Jingyang Yuan, Gongbo Sun, Zhiping Xiao, Hang Zhou, Xiao Luo, Junyu Luo, Yusheng Zhao, Wei Ju, and Ming Zhang. EGODE: An event-attended graph ODE framework for modeling rigid dynamics. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=js5vZtyoIQ.

Qianru Zhang, Haixin Wang, Cheng Long, Liangcai Su, Xingwei He, Jianlong Chang, Tailin Wu, Hongzhi Yin, Siu-Ming Yiu, Qi Tian, et al. A survey of generative techniques for spatial-temporal data mining. *arXiv preprint arXiv:2405.09592*, 2024.

Qianru Zhang, Peng Yang, Honggang Wen, Xinzhu Li, Haixin Wang, Fang Sun, Zezheng Song, Zhichen Lai, Rui Ma, Ruihua Han, et al. Beyond the time domain: Recent advances on frequency transforms in time series analysis. *arXiv preprint arXiv:2504.07099*, 2025.

Rui Zhang, Qi Meng, Rongchan Zhu, Yue Wang, Wenlei Shi, Shihua Zhang, Zhi-Ming Ma, and Tie-Yan Liu. Monte carlo neural operator for learning pdes via probabilistic representation. *Arxiv*, 2023.

Tianhan Zhang, Yuxiao Yi, Yifan Xu, Zhi X Chen, Yaoyu Zhang, E Weinan, and Zhi-Qin John Xu. A multi-scale sampling method for accurate and robust deep neural network to predict combustion chemical kinetics. *Combustion and Flame*, 245:112319, 2022.

Qingqi Zhao, Xiaoxue Han, Ruichang Guo, and Cheng Chen. A computationally efficient hybrid neural network architecture for porous media: Integrating cnns and gnns for improved permeability prediction. *Arxiv*, 2023.

Zhiyuan Zhao, Xueying Ding, and B. Aditya Prakash. PINNsformer: A transformer-based framework for physics-informed neural networks. In *The Twelfth International Conference on Learning Representations*, 2024.

Hongkai Zheng, Wenda Chu, Bingliang Zhang, Zihui Wu, Austin Wang, Berthy T Feng, Caifeng Zou, Yu Sun, Nikola Kovachki, Zachary E Ross, et al. Inversebench: Benchmarking plug-and-play diffusion priors for inverse problems in physical sciences. *arXiv preprint arXiv:2503.11043*, 2025.

Su Zheng, Zhengqi Gao, Fan-Keng Sun, Duane Boning, Bei Yu, and Martin D Wong. Improving neural ode training with temporal adaptive batch normalization. *Advances in Neural Information Processing Systems*, 37:95875–95895, 2024.

APPENDIX

A THE USE OF LARGE LANGUAGE MODELS (LLMS)

LLMs were not involved in the research ideation or the writing of this paper.

B NOTATION

Table 4 summarizes the key mathematical symbols and their meanings used throughout our benchmark framework and avoids symbol ambiguity across different methods, covering core components such as spatial representation, temporal modeling, and loss functions.

C MODULAR DECOMPOSITION

We systematically review 89 representative data-driven fluid simulation methods reported across major venues (ICLR, NeurIPS, ICML, JMLR, Nat. Mach. Intell., etc.) and decompose each method into four key design dimensions under a Taylor-expansion-inspired formulation: Design = "**Spatial representation**" + "**Temporal representation**" + "**Loss function**" + Additional Technique, where the "Additional Technique" captures the distinctive architectural or training innovation introduced by each work.

This decomposition allows us to (i) standardize diverse neural solver designs into a comparable format, (ii) isolate the contribution of each design choice (*e.g.*, spatial vs. temporal module), and (iii) facilitate cross-method analysis across 10 canonical flow scenarios under a unified lens. Instead of re-running all models, we extract the essential architectural and training components from each publication and categorize them in Table 5, thereby enabling a principled, reproducible, and cost-efficient comparison framework.

Table 5: We propose that the design of data-driven fluid simulation models can be analogized to a Taylor expansion. Design = "**Spatial representation**" + "**Temporal representation**" + "**Loss function**" + Additional Technique, where the additional item represents the key innovations introduced in each method.

#	Method	Publication	Spatial Repre.	Temporal Repre.	Loss	Additional Technique
1	FNO (Li et al.)	ICLR 2021	Fourier	Autoregression	Variable	Fourier integral operator
2	AFNO (Guibas et al., 2021)	ICLR 2022	Fourier	Autoregression	Variable	Mixing of tokens
3	Geo-FNO (Li et al., 2023a)	JMLR	Fourier	Autoregression	Variable	Data deformation
4	PINO (Li et al., 2021b)	ACM Data Sci.	Fourier	Autoregression	PDE loss	Efficient derivative compute
5	U-NO (Rahman et al., 2023)	TMLR	Fourier	Autoregression	Variable	U-net architecture
6	F-FNO (Tran et al., 2023)	ICLR 2023	Fourier	Autoregression	Variable	Factorize Fourier transform
7	CFNO (Brandstetter et al., 2023)	ICLR 2023	Fourier	Autoregression	Variable	Clifford algebras

Continued on next page

Table 5 – continued from previous page

#	Method	Publication	Spatial Repre.	Temporal Repre.	Loss	Additional Technique
8	CMWNO (Xiao et al., 2022)	ICLR 2023	Fourier	Autoregression	Variable	Multiwavelet decomposition
9	*G*-FNO (Helwig et al., 2023)	ICML 2023	Fourier	Autoregression	Variable	Group equivariant layers
10	GINO (Li et al., 2023b)	NeurIPS 2023	Graph/Fourier	Autoregression	Variable	Discretization convergence
11	DAFNO (Liu et al., 2023b)	NeurIPS 2023	Fourier	Autoregression	Variable	Smoothed characteristic func
12	PAC-FNO (Jeon et al., 2024)	ICLR 2024	Fourier	Autoregression	Variable	Parallel structure
13	AM-FNO (Xiao et al., 2024b)	NeurIPS 2024	Fourier	Autoregression	Variable	Amortized parameterization
14	DE-DON (Qiu et al., 2024)	NeurIPS 2024	MLP/Fourier	Autoregression	Derivative	Dimension reduction
15	MCNP (Zhang et al., 2023)	TPAMI	Fourier	Autoregression	Neural MC	Mathematical expectation
16	ST-FNO (Cao et al., 2025)	ICLR 2025	Fourier	Time-depth Conv	Variable	Spectral fine-tuning
17	scOT (Herde et al., 2024)	NeurIPS 2024	Self-atten	Temporal Bund	Variable	Large-scale pretraining
18	DiffPDE (Huang et al., 2024a)	NeurIPS 2024	Convolution	Autoregression	Noise	Guided Diffusion
19	Shysheya et al. (2024)	NeurIPS 2024	Convolution	Autoregression	Noise	Flexible pre- and post-training
20	Valencia et al. (2025)	ICLR 2025	Graph	Autoregression	Noise	Latent multi-scale GNN diffusion
21	Bastek et al. (2025)	ICLR 2025	Convolution	Autoregression	Noise	Physics informed diffusion
22	TA-BN (Zheng et al., 2024)	NeurIPS 2024	Convolution	Neural ODE	Variable	Temporal adaptive BatchNorm
23	Sholokhov et al. (2023)	Sci. Rep.	MLP	Neural ODE	variable	Collocation Points
24	Ghanem et al. (2025)	AAAI 2025	MLP	Neural ODE	Variable	Physical-informed loss
25	PGODE (Luo et al., 2024)	ICML2024	Self-atten	Neural ODE	Variable	Environment Parameter
26	Treat (Huang et al., 2024b)	NeurIPS 2024	Self-atten	Neural ODE	Vairable	Physical-informed regularization
27	Pure(Wu et al., 2024b)	NeurIPS 2024	Self-atten	Neural ODE	Vairable	Multi-view context information
28	EGODE(Yuan et al., 2024)	NeurIPS 2024	Self-atten	Neural ODE	Vairable	Event-attended information
29	Prometheus(Wu et al., 2024c)	ICML 2024	Self-atten	Neural ODE	Vairable	disentangled representations
30	HOPE(Luo et al., 2023b)	ICML 2023	Self-atten	Neural ODE	Vairable	High order information
31	LGODE(Huang et al., 2020)	NeurIPS 2020	Self-atten	Neural ODE	Vairable	Temporal Self-attention
32	Brandstetter et al. (2022)	ICLR 2022	Graph	Temporal Bund	Variable	Model training stability
33	BENO(Wang et al., 2024b)	ICLR 2024	Graph	Autoregression	Variable	Complex geometry embedding
34	UPT(Alkin et al., 2024)	NeurIPS 2024	Self-atten	Autoregression	Variable	Scalability across discretization
35	Transolver(Wu et al., 2024a)	ICML 2024	Self-atten	Autoregression	Variable	Physics-Attention
36	OMO(Xiao et al., 2024a)	ICML 2024	Self-atten	Autoregression	Variable	Orthogonal Attention
37	LSM(Wu et al., 2023a)	ICML 2023	Self-atten	Autoregression	Variable	Latent Propagation
38	GNOT(Hao et al., 2023a)	ICML 2023	Self-atten	Autoregression	Variable	Linear Attention
39	Galerkin-Trans(Cao, 2021)	NeurIPS 2021	Self-atten	Autoregression	Variable	Petrov-Galerkin projection
40	MWT(Gupta et al., 2021)	NeurIPS 2021	Convolution	Autoregression	Variable	Multi-wavelet filters
41	CARE (Luo et al., 2023a)	NeurIPS 2023	Graph	ODE	Variable	Context acquirement
42	Bryutkin et al. (2024)	ICML 2024	Graph+SA	Next-step	Variable	Modular input encoders
43	LAMP (Wu et al., 2023b)	ICLR 2023	Graph	Next-step	Variable	GNN-based actor-critic for policy
44	DINo (Yin et al., 2023)	ICLR 2023	Implicit	Neural ODE	Variable	Extrapolates at arbitrary loc
45	Zhao et al. (2023)	Adv Water Resour	Graph Conv	Next-step	Variable	GNN Grad-CAM
46	FCN (He et al., 2022)	PHYS FLUIDS.	Graph	Next-step	Variable	Vortex force contribution
47	GNODE (Bishnoi et al., 2023)	ICLR 2023	Graph	Neural ODE	Variable	Encode the constraints explicitly
48	MAgNet (Boussif et al., 2022)	NeurIPS 2022	Graph	Next-step	Variable	Implicit neural representation
49	TIE (Shao et al., 2022)	ECCV 2022	Self-atten	Next-step	Variable	Implicit Edges
50	HAN et al. (2022)	ICLR 2022	Self-atten	Self-atten	Variable	Encoder-decoder structure
51	MGN (Pfaff et al., 2021)	ICLR 2021	Graph	Next-step	Variable	Mesh graph representation
52	GNS (Sanchez et al., 2020)	ICML 2022	Graph	Nest-step	Variable	Message passing layers
53	RSteer (Wang et al., 2022)	ICLR 2023	Graph	Next-step	Variable	Approximately equivariant networks
54	EquNet (Wang et al.)	ICLR 2021	Conv	Temporal Bund	Variable	Incorporate symmetry
55	TF-Net (Wang et al., 2020)	KDD 2020	Conv	Next-step	Variable	Marry two simulation
56	DPUF (Lee & You, 2019)	J. Fluid Mech.	Conv	Next-step	Physical	Conservation of mass, momentum
57	Wessels et al. (2020)	CoM Appl M	MLP	Next-step	Variable	Updated Lagrangian
58	FGN (Li & Farimani, 2022)	Comput Graph	Graph	Next-step	Variable	Node-focused and edge-focused
59	MCC (Liu et al., 2023a)	AAAI 2023	Conv	Next-step	Variable	Dynamic multi-scale gridding
60	LFlows (Torres et al., 2023)	ICLR 2024	MLP	Temporal Bund	Flow	Satisfy the continuity equation
61	Li et al. (2024a)	Nat. Mach. Intell.	Conv	Next-step	Noise	Two different classes of DM
62	DeepONet (Lu et al., 2021)	Nat. Mach. Intell.	MLP	Next-step	Variable	Two sub-nets
63	Wang et al. (2021)	Sci. Advection	MLP	Next-step	Physics	Soft penalty constraints for law
64	MIONet (Jin et al., 2022)	Siam J Sci Comput	MLP	Next-step	Physics	A low-rank approximation
65	Geneva et al. (2022)	Neural Networks	Self-atten	Self-atten	Variable	Koopman dynamics
66	NOMAD (Seidman et al., 2022)	NeurIPS 2022	MLP	Next-step	Variable	Nonlinear decoder map
67	HyperDON (Lee et al., 2023)	ICLR 2023	MLP	Next-step	Variable	Use a hypernetwork
68	B-DON (Lin et al., 2023)	J. Comput. Phy.	MLP	Next-step	Variable	Bayesian network
69	SVD-DON (Venturi et al., 2023)	CoM Appl M	MLP	Next-step	Variable	Orthogonal decomposition
70	L-DON (Kontolati et al., 2023)	Arxiv 2023	MLP	Next-step	Variable	Low-dimensional latent space
71	GNOT (Hao et al., 2023a)	ICML 2023	Self-atten	Next-step	Variable	Heterogeneous normalized atten
72	CNO (Raonic et al., 2023)	NeurIPS 2023	Conv	Autoregression	Variable	Adaptations for convolution
73	FactFormer (Li et al., 2024b)	NeurIPS 2023	Self-atten	Self-atten	Variable	Axial factorized kernel
74	LNO (Cao et al., 2023a)	Nat. Mach. Intell.	Laplace	Autoregression	Variable	Pole-residue relationship
75	KNO (Xiong et al., 2023)	APL ML	Koopman	Autoregression	Variable	Approximate the Koopman operator
76	ICON (Yang et al., 2023)	PNAS	Self-atten	Self-atten	variable	In-context learning

Continued on next page

Table 5 – continued from previous page

#	Method	Publication	Spatial Repre.	Temporal Repre.	Loss	Additional Technique
77	Transolver (Wu et al., 2024a)	ICML 2024	Self-atten	Self-atten	Variable	Intrinsic physical states
78	FCG-NO (Rudikov et al., 2024)	ICML 2024	Frequency	Autoregression	Variable	Flexible conjugate gradient
79	PINN (Raissi et al., 2019)	J. Comput. Phy.	MLP	Next-step	Residual	-
80	PINN-SR (Chen et al., 2021)	Nat. Commun.	MLP	Next-step	Residual	Regularization loss
81	NSFnet (Jin et al., 2021)	J. Comput. Phys.	MLP	Next-step	Residual	For N-S Eq.
82	MAD (Huang et al., 2022)	NeurIPS 2022	MLP	Next-step	Variable	Meta-learning
83	Toloubidokhti et al. (2023)	ICLR 2024	MLP	Next-step	Residual	Difficulty-aware task sampler
84	Zhao et al. (2024)	ICLR 2024	Self-atten	Next-step	Residual	Pseudo sequences
85	PeRCNN (Rao et al., 2023)	Nat. Mach. Intell.	Conv	Next-step	Variable	Forcibly encodes physics structure
86	FFM (Kerrigan et al., 2023)	AISTATS 2024	Conv	Next-step	Flow	Function space generation
87	ECI (Cheng et al., 2024)	ICLR 2025	Fourier	Next-step	Flow	Integration of constraint
88	FunDS (Yao et al., 2025)	Arxiv 2025	Fourier	Next-step	Noise	Function-space posterior sampling
89	FunDIff (Wang et al., 2025a)	Arxiv 2025	Self-atten	Next-step	Noise	Latent diffusion process

D Supplementary Results on Additional Techniques

In fact, when conducting experiments for each method, researchers often incorporate techniques from the field of deep learning to achieve better performance, smoother optimization, and more stable training. Considering that, in practice, different users may add different additional techniques, we provide examples in our code showing how to quickly add any desired technique. When designing the code repository, we reserve kwargs for each model to support various extensions. Using '$\mathcal{X}$ = `self-attention`' as an example, we list common techniques, including `GlobalFilter`, `Long-short term attention`, and `Adaptive Fourier attention`. Specifically, we show the experimental results on `Compressible N-S` with '$\mathcal{Y}$ = `next-step`' and '$\mathcal{Z}$ = `variable`' are shown below.

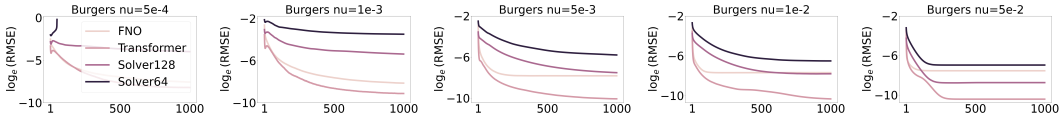


Figure 8: We compare FNO and Transformer against pseudo-spectral solver operating at lower resolutions on Burgers' equation across five kinematics viscosity. Solver x implies pseudo-spectral solver operating at $x \times x$ resolution.

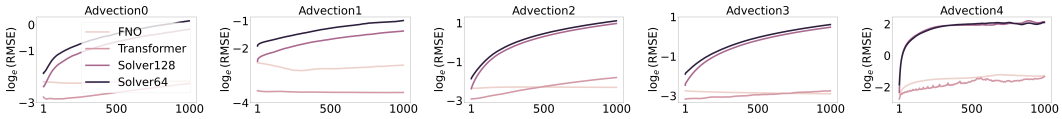


Figure 9: We compare FNO and Transformer against pseudo-spectral solver operating at lower resolutions on Advection equation across five advection speed types. Solver x implies pseudo-spectral solver operating at $x \times x$ resolution.

E Supplementary Results on Comparison with Traditional Solvers

To further demonstrate the performance of neural operators relative to traditional numerical solvers, we present additional rollout results and runtime comparisons. Figure 8 reports the evolution of the rollout error for Burgers' equation under five viscosity regimes, ranging from $\nu = 5 \times 10^{-4}$ to $\nu = 5 \times 10^{-2}$. We observe that neural operator models, particularly FNO, consistently achieve lower error over long rollouts compared to traditional finite-difference solvers at multiple resolutions (Solver64, Solver128). This advantage becomes more pronounced as the viscosity increases, where the numerical dissipation in coarse solvers leads to early error saturation. Similar trends are observed for the Advection equation (Figure 9), confirming the generality of this behavior across PDE types.

In terms of efficiency, Figure 11 compares the averaged per-step runtime across models and solver resolutions. FNO and Transformer-based models offer significant speedups over Solver 128 and

Symbol	Meaning
Ω	Spatial domain with dimension d, $\Omega \subset \mathbb{R}^d$
T	Temporal horizon, $t \in [0, T]$
$u(\boldsymbol{x}, t)$	Field variable at location $\boldsymbol{x}$ and time t with c channels
$\{\boldsymbol{x}_i\}_{i=1}^N$	Spatial locations of N samples, $\boldsymbol{x}_i \in \Omega$
M	Number of temporal snapshots
$\mathcal{S}_\theta, \mathcal{T}_\theta$	Spatial and temporal encoder parameterized by θ
z, h	Feature representation in spatial and temporal domain
$\mathcal{F}, \mathcal{F}^{-1}$	Fourier and inverse Fourier transforms
$\hat{u}(\mathbf{k}, t)$	Spectral coefficient at frequency $\mathbf{k}$
$\phi_\theta(\mathbf{k})$	Learnable spectral filter of frequency index $\mathbf{k}$
$\odot$	Elementwise (Hadamard) product
$\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V$	Query, key and value projection matrices, $\in \mathbb{R}^{c \times d_k}$
d_k, d_k, d_v	Query/key and Value dimension in self-attention
$\mathcal{H}$	Set of stencil offsets for convolution
$c_{\text{in}}, c_{\text{out}}$	Number of input and output channels to convolution
$\mathbf{W}_h$	Convolution kernel at offset h, $\in \mathbb{R}^{c_{\text{out}} \times c_{\text{in}}}$
$\tilde{A}, \tilde{D}$	Adjacency with self-loops, $\tilde{A} = A + I$, and degree matrix of $\tilde{A}$
K	Number of retained modes in ROM, $K \ll N$
$C(\boldsymbol{x}, \boldsymbol{x}')$	Covariance kernel between spatial locations
$\phi_k(\boldsymbol{x})$	k-th POD basis function
λ_k	Eigenvalue associated with ϕ_k
$\alpha_k(t)$	POD projection coefficient at time t
$\langle f, g \rangle$	L^2 inner product over Ω
$\mathbf{d}^j$	Predicted increment for bundling step j
$h(t)$	Latent state in Neural ODE
$\zeta(\cdot), \xi(\cdot)$	Encoder/decoder for latent ODE state
ε_θ	Network estimating noise in diffusion models
$u_\tau, \bar{\alpha}_\tau$	Noisy version of field and noise schedule coefficient in diffusion
$v_\theta, \psi_\tau(u)$	Predicted velocity field and interpolated affine flow
$\mathcal{R}[u_\theta](x, t)$	Residual of the PDE operator
$g(x, t)$	Source term in PDE

Table 4: Unified notation across spatial, temporal, and loss components in our paper.

Additional Technique	**RMSE**	**nRMSE**	**fRMSE**
Self-attention + Next-step + Physical variable + ***Additional Techniques***			
Vanilla Self-attention	0.0571	0.0695	0.0035
GlobalFilter	0.0642	0.0711	0.0052
Long-short term attention	0.0559	0.0709	0.0043
Adaptive Fourier attention	0.0597	0.0699	0.0047

Table 6: Performance comparison of various additional techniques.

Solver 256, especially when considering high-resolution solvers that incur large computational overhead. This speed advantage, combined with improved predictive accuracy, highlights the practical benefits of neural operators as surrogates for long-horizon PDE simulation. These results suggest that data-driven neural surrogates can provide a favorable trade-off between accuracy and computational cost, making them attractive for real-time or large-scale simulation scenarios. In conclusion, FNO not only achieves better predictive accuracy but also faster speed.

E.1 Neural Solver with Various Step Sizes

We train FNO to make predictions with different step sizes. *FNO-x* denotes FNO trained to predict with step size $x\Delta t^*$, where $\Delta t^* = \frac{1}{32}$. The results are shown in Figure 12, and they demonstrate that increasing the step sizes leads to only minor performance degradation. Figure 10 reports the runtime comparison for predicting up to $T = \frac{1}{2}$, where we observe speedups of several hundred times for *FNO-16*. The experiment is conducted on incompressible N-S dataset.

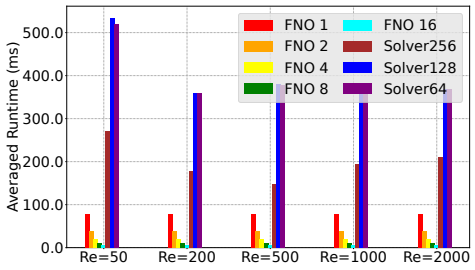
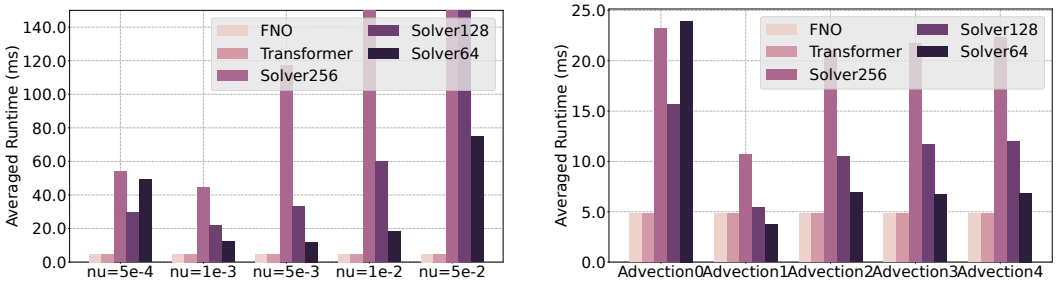


Figure 10: We evaluate the runtime performance of the FNO when trained to predict various stepsize against pseudo-spectral solver operating at lower resolutions for predicting up to $T = \frac{1}{2}$. Solver x implies pseudo-spectral solver operating at $x \times x$ resolution. Data generated at 256×256 are considered ground truth. To ensure numerical stability, adaptive time-stepping governed by the CFL condition is adopted even for solvers at low resolution. *FNO-*x denotes FNO trained to predict with stepsize $x\Delta t^*$, where $\Delta t^* = \frac{1}{32}$.



Burgers' Equation *Advection Equation*

Figure 11: We compare the runtime of the FNO and Transformer against pseudo-spectral solver operating at lower resolutions for predicting up to $T = \frac{1}{32}$. Solver x implies pseudo-spectral solver operating at $x \times x$ resolution. Data generated at 256×256 are considered ground truth. Note that to ensure numerical stability, adaptive time-stepping governed by CFL condition is adopted even for solvers at low-resolution.

F Supplementary Results on Different Modules

In this section, we provide additional experimental results on the evaluation of different modular designs on prediction performance and computational efficiency across Stochastic N-S. The quantitative results are summarized in Table 7.

From the first block of the table, which compares different spatial representations $\mathcal{X}$ under a fixed next-step predictor and physical variable decoding head, we observe that Fourier-based representations achieve the lowest RMSE (0.0398) despite having a relatively large parameter count (72M), demonstrating their strong inductive bias for capturing global structures in stochastic flows. Convolutional representations, while requiring fewer parameters (7.63M), exhibit competitive nRMSE and fRMSE, suggesting that local receptive fields are still effective when combined with sufficient depth. Interestingly, graph-based representations lag behind all other variants, highlighting their limited expressiveness in handling high-resolution stochastic turbulence.

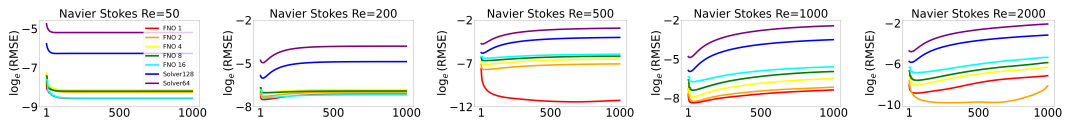


Figure 12: We compare FNO when trained to predict with various stepsize against pseudo-spectral solver operating at lower resolutions on incompressible N-S across five Reynold numbers. Solver x implies pseudo-spectral solver operating at $x \times x$ resolution. *FNO-*x denotes FNO trained to predict with stepsize $x\Delta t^*$, where $\Delta t^* = \frac{1}{32}$.

In the second block, we fix the spatial representation to self-attention and vary the temporal modeling module $\mathcal{Y}$. Neural ODE-based temporal evolution provides the most stable long-horizon predictions, achieving the lowest errors across all three metrics, confirming its ability to model continuous-time dynamics more faithfully. Temporal bundling also improves over simple next-step prediction by reducing error accumulation over time, while pure self-attention temporal modeling performs the worst, potentially due to overfitting to short-term dependencies.

Overall, these results indicate that (1) Fourier-based spatial encoders are well-suited for stochastic PDE modeling when computational resources allow, (2) convolutional encoders strike a good balance between efficiency and accuracy, and (3) Neural ODE is the most robust choice for temporal evolution, especially under stochastic forcing. These findings justify the adoption of Fourier representation with Neural ODE temporal modeling in our final architecture.

		Stochastic N-S		
MODEL	PARAM	RMSE↓	NRMSE↓	FRMSE↓
$\mathcal{X}$ + Next-step + Physical variable				
$\mathcal{X}$ = Graph representaion	3.13M	0.0917	0.2256	0.0067
$\mathcal{X}$ = Conv representation	7.63M	0.0438	0.0325	0.0042
$\mathcal{X}$ = Fourier representation	72.0M	0.0398	0.0463	0.0050
$\mathcal{X}$ = Latent representation	89.0M	0.0560	0.0392	0.0072
$\mathcal{X}$ = Self-attention representation	36.8M	0.0500	0.0430	0.0047
Self-attention representation + $\mathcal{Y}$ + Physical variable				
$\mathcal{Y}$ = Self-attention	36.4M	0.1094	0.0740	0.0104
$\mathcal{Y}$ = Temporal Bundling	36.3M	0.0998	0.0675	0.0089
$\mathcal{Y}$ = Neural ODE	37.1M	0.0913	0.0617	0.0083

Table 7: Evaluation of different modular designs on prediction performance and computational efficiency across Stochastic N-S.

G SUPPLEMENTARY RESULTS ON GENERALIZATION STUDY

G.1 LONG-HORIZON ROLLOUT FOR DIFFUSION AND DETERMINISTIC MODELS

In our generalization experiments, we primarily select the optimal loss setting, i.e., the physical variable prediction setting. In addition, we compare deterministic models trained with physical variable prediction loss and those trained with a denoising loss. The generalization ability is evaluated on both out-of-distribution (OOD) data and at longer time horizons. The network architecture employs self-attention for both spatial and temporal representations, and all experiments are conducted on the Compressible Navier–Stokes equations.

Out-of-Distribution Generalization. For the OOD generalization experiments, the model is trained only on the setting "Mach = 0.1, $\zeta = 10^{-8}$" and evaluated on unseen viscosity parameters. Table 8 summarizes the results, showing that models trained with denoising loss consistently outperform their deterministic counterparts.

Table 8: Generalization performance on out-of-distribution data. Models trained with denoising loss exhibit consistently lower RMSE and nRMSE across all conditions.

Mach	ζ	Method	RMSE ↓	nRMSE ↓
0.1	1e-8	Deterministic	0.2278	0.2033
		Denoising	**0.1664**	**0.1530**
0.1	1e-2	Deterministic	0.9129	0.3669
		Denoising	**0.7540**	**0.3336**
0.1	1e-1	Deterministic	0.6660	0.2720
		Denoising	**0.5427**	**0.2744**
1.0	1e-8	Deterministic	0.8693	0.7591
		Denoising	**0.8156**	**0.7162**
1.0	1e-2	Deterministic	0.5098	0.6160
		Denoising	**0.4924**	**0.6048**
1.0	1e-1	Deterministic	0.6801	0.2910
		Denoising	**0.5679**	**0.2923**

Long-Horizon Generalization. For long-horizon tests, the model is trained with next-step prediction only, and rollout is performed autoregressively at test time. Table 9 reports the RMSE and

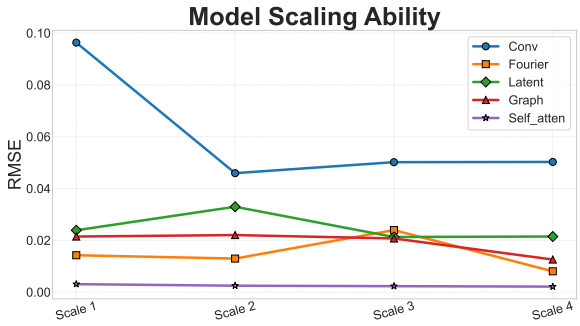


Figure 13: RMSE versus model capacity for five architectures at four scaling levels.

nRMSE for up to 16 rollout steps. The denoising-trained model consistently achieves lower errors and exhibits improved stability over longer horizons.

Table 9: Long-horizon rollout performance. Denoising-trained models maintain lower error accumulation compared to deterministic models, demonstrating enhanced stability.

Time Step	Deterministic RMSE	Denoising RMSE	Deterministic nRMSE	Denoising nRMSE
1	0.1754	**0.1601**	0.1917	**0.1611**
2	0.1753	**0.1390**	0.1940	**0.1422**
3	0.1739	**0.1313**	0.1960	**0.1355**
4	0.1755	**0.1290**	0.2012	**0.1329**
5	0.2484	**0.1742**	0.2873	**0.1869**
6	0.2470	**0.1621**	0.2915	**0.1742**
7	0.2458	**0.1593**	0.2929	**0.1734**
8	0.2530	**0.1600**	0.3043	**0.1729**
9	0.3086	**0.1936**	0.3822	**0.2164**
10	0.3135	**0.1842**	0.3956	**0.2059**
11	0.3023	**0.1812**	0.3877	**0.2065**
12	0.3280	**0.1838**	0.4160	**0.2071**
13	0.3688	**0.2116**	0.4800	**0.2414**
14	0.3754	**0.2078**	0.5083	**0.2366**
15	0.3659	**0.2071**	0.4926	**0.2407**
16	0.3979	**0.2167**	0.5273	**0.2476**

Overall, these results confirm that training with a denoising loss improves both OOD generalization and long-horizon stability.

G.2 Scaling Analysis

We systematically assessed the impact of model capacity on predictive accuracy across five representative architectures by defining four scaling levels for each:

- **Convolutional modules:** number of layers $\in \{2, 3, 4, 5\}$
- **Fourier-based learning:** hidden dimension $\in \{64, 128, 256, 512\}$
- **Latent learning:** attention dimension $\in \{64, 128, 256, 512\}$
- **Self-attention learning:** hidden size $\in \{96, 192, 384, 768\}$
- **Graph learning:** hidden feature size $\in \{64, 128, 256, 512\}$

As shown in Figure 13, RMSE decreases monotonically from Scale 1 to Scale 4 for all five methods. The self-attention model achieves the largest absolute reduction (over 0.08), demonstrating its superior ability to exploit increased representational capacity. The Fourier- and graph-based models follow, with improvements of approximately 0.06 and 0.05, respectively, indicating diminishing returns at higher dimensions. The convolutional baseline exhibits the smallest gain (about 0.03), suggesting that depth alone yields limited expressivity for this task, while the latent flow model attains an intermediate reduction (about 0.04). These findings underscore that transformer-style self-attention mechanisms benefit most from scaling, making them particularly well-suited for high-fidelity modeling in our PDE-driven benchmarks.

H More Implementation Details

Modular Comparison. All of our experiments are implemented on PyTorch on 8 × NVIDIA A6000 GPUs. For different experiments, we utilize different optimizers with different strategies, including weight decay, learning rates, and scheduler obtained from grid search for all experiments.

Traditional solvers. FNO uses width=12 and width=32. We train the model for 10 epochs using a cosine annealing learning rate with a starting learning rate of $1e-4$ and an end learning rate of $1e-6$. We use the Adam optimizer.

Discretization comparison. FNO uses width=32 and width=128. MeshGraphnets uses latent size=128 and message passing steps=10. GNS uses latent size=128 and message passing steps=10. All models are trained for 100 epochs using a cosine annealing learning rate with a start learning rate of $1e-4$ and an end learning rate of $1e-5$. We use Adam optimizer for all model training.

Hyperparameter Selection. We first normalize the computational budgets by adjusting the GFLOPs of all methods to comparable levels. Subsequently, we conduct a systematic grid search over both model-level and training-level hyperparameters, exploring architecture choices, learning rates, regularization strengths, and optimization strategies. For each method, we select the configuration that achieves the best validation performance under the matched compute budget. While we strive to showcase the strongest performance of each baseline, we acknowledge that absolute fairness across all possible hyperparameter combinations is inherently challenging. Our procedure nonetheless provides a principled and reproducible way to tune each method to near-optimal performance.

Protocol Template Availability. To further facilitate reproducibility and fair comparison, we also include the full protocol templates in the released codebase. These templates specify the optimizer type, search ranges for learning rate and weight decay, scheduler choices, and training length for each module family. Users can directly load these configurations to reproduce our reported results or extend them to new architectures with minimal modification. Please refer to the `config` folder in our anonymous GitHub codebase https://anonymous.4open.science/r/FD-Bench-15BC for more details.

I Easy Integration of Existing Leaderboard

Our benchmark is designed to be seamlessly extensible and easily integrated with public leaderboards and community-driven competitions. For example, users can directly build from widely used public libraries such as `neuraloperator` by adding them as a submodule, installing their dependencies, and defining a simple configuration file under `config/public`. The model can then be initialized in `src/train.py` and executed using our unified training scripts (*e.g.*, `bash src/train_public.sh`). This plug-and-play design significantly lowers the barrier to participation, enabling researchers to quickly evaluate their methods under standardized settings. This will encourage community participation, provide a living benchmark for tracking state-of-the-art progress, and foster reproducible and comparable research in data-driven fluid simulation.

J More Datasets Details

J.1 Datasets for Modular Comparison

Compressible N-S Equations. The compressible N-S (CNS) equations govern the dynamics of fluid flows with variable density and internal energy. The compressible model accounts for variations in density ρ and pressure p due to thermodynamic effects, and captures phenomena such as shock waves, sound propagation, and high-speed turbulent mixing.

We utilize the 2D data from PDEBench (Takamoto et al., 2022). The general form of the CNS equations in conservative form is given by:

$$\text{Mass conservation (Continuity equation):} \quad \partial_t \rho + \nabla \cdot (\rho \mathbf{v}) = 0, \tag{1a}$$

$$\text{Momentum conservation:} \quad \partial_t (\rho \mathbf{v}) + \nabla \cdot (\rho \mathbf{v} \otimes \mathbf{v}) = -\nabla p + \nabla \cdot \sigma, \tag{1b}$$

$$\text{Total energy conservation:} \quad \partial_t E + \nabla \cdot [(E+p)\mathbf{v}] = \nabla \cdot (\sigma \cdot \mathbf{v}) + \nabla \cdot (k \nabla T), \tag{1c}$$

where ρ is the mass density, $\mathbf{v} \in \mathbb{R}^d$ is the velocity field, p is the pressure, $E = \epsilon + \frac{1}{2}\rho\|\mathbf{v}\|^2$ is the total energy per unit volume, ϵ is the internal energy, T is the temperature, k is the thermal conductivity, and σ is the viscous stress tensor given by:

$$\sigma = \eta\left[\nabla\mathbf{v} + (\nabla\mathbf{v})^T\right] + \left(\zeta - \frac{2}{3}\eta\right)(\nabla\cdot\mathbf{v})\mathbf{I}, \tag{2}$$

with η and ζ denoting shear and bulk viscosities, respectively. To close the system, an equation of state (EOS) is required. A common choice for an ideal gas is:

$$p = (\gamma - 1)\epsilon, \quad \text{or} \quad p = \rho RT, \tag{3}$$

where γ is the adiabatic index and R is the specific gas constant.

The CNS equations describe a wide range of fluid phenomena where compressibility effects are non-negligible. In particular, they can capture shock and rarefaction waves, supersonic and hypersonic flows, acoustic propagation, atmospheric and weather systems, combustion and detonation and astrophysical flows.

Diffusion-Reaction Flow. We consider a 2D diffusion-reaction system involving two nonlinearly coupled variables: the activator $u = u(t, x, y)$ and the inhibitor $v = v(t, x, y)$. We utilize the data from PDEBench (Takamoto et al., 2022). The governing equations are:

$$\partial_t u = D_u(\partial_{xx}u + \partial_{yy}u) + R_u, \tag{4}$$
$$\partial_t v = D_v(\partial_{xx}v + \partial_{yy}v) + R_v\,, \tag{5}$$

where D_u and D_v denote the diffusion coefficients, and $R_u(u, v)$ and $R_v(u, v)$ represent the nonlinear reaction terms for the activator and inhibitor, respectively. The simulation is defined over the spatial domain $x, y \in (-1, 1)$ and temporal range $t \in (0, 5]$. This system is widely used to model biological pattern formation. The reaction terms follow the FitzHugh–Nagumo model (Klaasen & Troy, 1984):

$$R_u(u, v) = u - u^3 - k - v, \tag{6}$$
$$R_v(u, v) = u - v, \tag{7}$$

with $k = 5\times10^{-3}$, and diffusion coefficients $D_u = 1\times10^{-3}$, $D_v = 5\times10^{-3}$. The initial condition is sampled from a standard normal distribution: $u(0, x, y) \sim \mathcal{N}(0, 1.0)$, for $x, y \in (-1, 1)$. We provide simulation data discretized at high resolution ($N_x = N_y = 512, N_t = 501$) and a downsampled version ($N_x = N_y = 128, N_t = 101$) for training. Spatial discretization uses the finite volume method (Moukalled et al., 2016), and time integration is performed via a fourth-order Runge-Kutta scheme from the `scipy` library.

Kolmogorov Flow. It is a novel physical scenario and a 2D variant of the classical Kolmogorov flow (Majda et al., 2002). We follow Poseidon (Herde et al., 2024) for data settings. This flow is governed by the incompressible N-S equations with an external forcing term. The governing equations take the form:

$$u_t + (u\cdot\nabla)u + \nabla p - \nu\Delta u = f, \quad \text{div } u = 0, \tag{8}$$

defined over the spatial domain $[0, 1]^2$ with periodic boundary conditions imposed in both spatial directions. Here, $u = (u_x, u_y)$ denotes the velocity field, p is the pressure, ν is the viscosity, and f is the external forcing. The forcing term f is smooth, spatially varying, and held constant over time. It is defined as:

$$f(x, y) = 0.1\sin(2\pi(x + y)). \tag{9}$$

This configuration introduces a diagonal sinusoidal excitation that drives the flow in a nontrivial manner, resulting in rich dynamics ideal for evaluating neural operator performance. The initial conditions are based on the vorticity, which is assumed to be constant along a uniform (square) partition of the underlying domain. Numerical simulations are conducted using the same discretization and time-stepping schemes adopted for other N-S-based flows.

This problem can be recast into the unified PDE framework by defining an augmented state variable $U = [u_x, u_y, f]$, and introducing the trivial evolution equation $\partial_t f = 0$ to reflect the time-invariance of the forcing. The initial data is augmented accordingly using Equation equation 9. The corresponding solution operator $\mathcal{S}(t, \cdot)$ maps the initial state $U^0_{x,y}$ to the solution at time t as:

$$\mathcal{S}(t, U^0_{x,y}) = [u_x(t), u_y(t), f] \tag{10}$$

where (u_x, u_y) evolve according to the forced N-S Equations equation 8, and f remains fixed over time.

Stochastic N-S Equation. We extend the existing 2D incompressible N-S equation by adding a stochastic forcing term, resulting in the 2D stochastic N-S equation, an emerging modeling class of fluid dynamics that aims to better handle uncertainty and study the mix-in behavior of fluid flows (Mikulevicius & Rozovskii, 2004). In particular, we present semilinear form over the space-time interval $(t, x) \in [0, T] \times [0, 1]^2$, similar to the dataset in (Salvi et al., 2022):

$$\frac{\partial \mathbf{u}}{\partial t} = \nu \Delta \mathbf{u} + \mathbf{f} + \sigma \xi, \quad \mathbf{u}(0, x) = \mathbf{u}_0(x), \tag{11}$$

where $\xi = \dot{W}$ for W a Q-Wiener process, colored in space and with scale $\sigma = 0.05$ and viscosity $\nu = 10^{-4}$. For our data generation, the initial condition is sampled according to $\mathbf{u}_0 \sim \mathcal{N}(0, 3^{3/2}(-\Delta + 49I)^{-3})$ with periodic boundary conditions.

J.2 Datasets for comparison with numerical solvers

2D Incompressible N-S Equation in Vorticity Form. This pseudo-spectral N-S solver uses a semi-implicit, two-stage (Heun) time integrator and adaptive time stepping governed by CFL conditions. At each step, the vorticity field $\omega(x, y)$ is transformed to Fourier space, the stream-function ψ is obtained via multiplication by the precomputed inverse Laplacian, and the velocity $(u, v) = (\partial_y \psi, , -\partial_x \psi)$ is recovered by spectral differentiation. Nonlinear advection is computed in physical space and re-transformed, while viscous diffusion is handled implicitly through Crank–Nicolson scaling. To suppress aliasing, high-wavenumber modes are zeroed via a $2/3$-rule mask.

We use a physical domain of 1×1 ($L_1 = L_2 = 1$) and a discretization grid size of 256×256 ($s_1 = s_2 = 256$). The total time $T = 20$, and we store vorticity data at each $\frac{1}{50}$ intervals. This means that the neural solvers make predictions with $\Delta t = \frac{1}{50}$. We generate 1200 sequences each with a sequence length of 1000. We use 1000 sequences for training, 100 sequences for validation, and 100 sequences for testing. The external forcing term is $f(x, y) = 0.1(\sin(2\pi(x + y)) + \cos(2\pi(x + y)))$. The initial vorticity is generated from a 2D gaussian random field.

2D Burgers' Equation. This solver advances the 2D Burgers equations as follows:

$$\begin{aligned} u_t + u\, u_x + v\, u_y &= \nu(u_{xx} + u_{yy})\,, \\ v_t + u\, v_x + v\, v_y &= \nu(v_{xx} + v_{yy})\,. \end{aligned} \tag{12}$$

using a pseudo-spectral RK4 integrator. Spatial derivatives are computed in Fourier space, while nonlinear advection is formed in physical space and then dealiased with a 2/3-rule mask. Viscous diffusion appears as a spectral multiplier $-\nu, K^2$ and is treated in the same explicit stages. Crucially, the time step Δt is chosen adaptively via a CFL constraint.

We use a physical domain of 1×1 ($L_1 = L_2 = 1$) and a discretization grid size of 256×256 ($s_1 = s_2 = 256$). The total time $T = 20$, and we store velocity data at each $\frac{1}{50}$ intervals. We generate 1200 sequences each with a sequence length of 1000. We use 1000 sequences for training, 100 sequences for validation, and 100 sequences for testing. The initial velocity is generated from a 2D gaussian random field.

ComputeRHS$(\hat{u}, \hat{v})$: Compute nonlinear advection and diffusion in spectral space, i.e.

$$R_u = -\,\mathcal{F}\{u\, u_x + v\, u_y\}\, M \;-\; \nu\, K^2\, \hat{u}, \quad R_v = -\,\mathcal{F}\{u\, v_x + v\, v_y\}\, M \;-\; \nu\, K^2\, \hat{v}.$$

2D Advection Equation. This solver advances the linear advection equation

$$u_t + v_x(x, y)\, u_x + v_y(x, y)\, u_y = 0 \tag{13}$$

via a pseudo-spectral Runge–Kutta 4 scheme. At each step, the field is FFT-transformed, the spatial-dependent velocity is sampled, and an adaptive CFL-based time step is chosen to ensure stability. Nonlinear advection fluxes are computed in physical space, re-transformed, differentiated spectrally, and then combined in the RK4 stages; dealiasing enforces the 2/3-rule before the next step.

Algorithm 1 Pseudo-Spectral Solver for 2D Incompressible N-S

Require: initial vorticity $\omega^0(x, y)$, forcing $f(x, y)$ (optional), final time T, Reynolds number Re, grid sizes (s_1, s_2), domain (L_1, L_2)

Build 2/3-rule dealiasing mask $M = \big(|K_X| < \frac{2}{3}\max|k_x|\big) \wedge \big(|K_Y| < \frac{2}{3}\max|k_y|\big)$
$t \leftarrow 0$
while $t < T$ **do**
  Compute velocity $\mathbf{u} = (u, v)$ from $\hat{\omega}$ via Poisson-solve + spectral derivatives
  Determine time step Δt from CFL:

$$\Delta t = \min\big(\mathrm{cfl}\, h/\|\mathbf{u}\|_\infty,\ \mathrm{cfl}\, h^2/\nu\big), \quad \nu = 1/\mathrm{Re}$$

  $\mathcal{N}^{(1)} \leftarrow -\mathcal{F}\{\mathbf{u}\cdot\nabla\omega\} + \hat{f}$
  Predictor:

$$\hat{\omega}^* \leftarrow \frac{\hat{\omega} + \Delta t\big(\mathcal{N}^{(1)} - \frac{\nu}{2} G\,\hat{\omega}\big)}{1 + \frac{\nu}{2} G\,\Delta t}$$

  Compute $\mathcal{N}^{(2)}$ at $\hat{\omega}^*$ similarly
  Corrector:

$$\hat{\omega} \leftarrow \frac{\hat{\omega} + \Delta t\big(\frac{\mathcal{N}^{(1)}+\mathcal{N}^{(2)}}{2} - \frac{\nu}{2} G\,\hat{\omega}\big)}{1 + \frac{\nu}{2} G\,\Delta t}$$

  Dealias: $\hat{\omega} \leftarrow \hat{\omega}\ \cdot\ M$
  $t \leftarrow t + \Delta t$
end while
$\omega(x, y) \leftarrow \mathcal{F}^{-1}\{\hat{\omega}\}$
Recover ψ and then $\mathbf{u} = (\partial_y\psi,\ -\partial_x\psi)$

Algorithm 2 Pseudo-Spectral Solver for 2D Burgers Equation

Require: initial velocity fields (u^0, v^0), final time T, viscosity ν, grid sizes (s_1, s_2), domain (L_1, L_2)

Compute $K^2 = K_X^2 + K_Y^2$, replace zero-mode by ε to avoid division by zero
Build 2/3-rule dealiasing mask $M = \big(|K_X| < \frac{2}{3}\max|k_x|\big) \wedge \big(|K_Y| < \frac{2}{3}\max|k_y|\big)$
$t \leftarrow 0$
while $t < T$ **do**
  Determine time step Δt from CFL:

$$\Delta t = \min\big(\mathrm{cfl}\, h/U_{\max},\ \mathrm{cfl}\, h^2/\nu\big)$$

  // RK4 stages
  Compute $(k_1^u, k_1^v) \leftarrow \textsc{ComputeRHS}(\hat{u}, \hat{v})$
  Compute $(k_2^u, k_2^v) \leftarrow \textsc{ComputeRHS}(\hat{u} + \frac{\Delta t}{2}k_1^u,\ \hat{v} + \frac{\Delta t}{2}k_1^v)$
  Compute $(k_3^u, k_3^v) \leftarrow \textsc{ComputeRHS}(\hat{u} + \frac{\Delta t}{2}k_2^u,\ \hat{v} + \frac{\Delta t}{2}k_2^v)$
  Compute $(k_4^u, k_4^v) \leftarrow \textsc{ComputeRHS}(\hat{u} + \Delta t\,k_3^u,\ \hat{v} + \Delta t\,k_3^v)$
  $\hat{u} \leftarrow \hat{u} + \frac{\Delta t}{6}(k_1^u + 2k_2^u + 2k_3^u + k_4^u)$
  $\hat{v} \leftarrow \hat{v} + \frac{\Delta t}{6}(k_1^v + 2k_2^v + 2k_3^v + k_4^v)$
  Apply dealiasing: $\hat{u} \leftarrow \hat{u}\cdot M,\ \hat{v} \leftarrow \hat{v}\cdot M$
  $t \leftarrow t + \Delta t$
end while
$(u, v) \leftarrow \mathcal{F}^{-1}\{\hat{u}, \hat{v}\}$

We use a physical domain of 1×1 ($L_1 = L_2 = 1$) and a discretization grid size of 256×256 ($s_1 = s_2 = 256$). The total time $T = 20$, and we store field data at each $\frac{1}{50}$ intervals. We generate 1200 sequences each with a sequence length of 1000. We use 1000 sequences for training, 100

Algorithm 3 Pseudo-Spectral RK4 Solver for 2D Linear Advection

Require: initial scalar field u^0, , final time T, advection speed function VELOCITY(x, y), grid sizes (s_1, s_2), domain (L_1, L_2)

Precompute wavenumbers K_X, K_Y and 2/3-rule dealias mask M
$\hat{u} \leftarrow \mathcal{F}\{u\}$
$t \leftarrow 0$
while $t < T$ **do**
  Obtain velocity field $(v_x, v_y) \leftarrow$ VELOCITY(x, y, t)
  Determine time step Δt from CFL: $\Delta t \leftarrow 0.5 \min\big(L_x/(N_x\|v_x\|_\infty),\ L_y/(N_y\|v_y\|_\infty)\big)$
  // RK4 stages
  $k_1 \leftarrow \text{RHS}(\hat{u}, v_x, v_y)$
  $k_2 \leftarrow \text{RHS}(\hat{u} + \frac{\Delta t}{2}k_1,\ v_x(t + \frac{\Delta t}{2}),\ v_y(t + \frac{\Delta t}{2}))$
  $k_3 \leftarrow \text{RHS}(\hat{u} + \frac{\Delta t}{2}k_2,\ v_x(t + \frac{\Delta t}{2}),\ v_y(t + \frac{\Delta t}{2}))$
  $k_4 \leftarrow \text{RHS}(\hat{u} + \Delta t\, k_3,\ v_x(t + \Delta t),\ v_y(t + \Delta t))$
  Update spectral solution:

$$\hat{u} \leftarrow \hat{u} + \tfrac{\Delta t}{6}\,(k_1 + 2k_2 + 2k_3 + k_4)$$

  Apply dealiasing: $\hat{u} \leftarrow \hat{u} \cdot M$
  $t \leftarrow t + \Delta t$
end while
$u \leftarrow \mathcal{F}^{-1}\{\hat{u}\}$

sequences for validation, and 100 sequences for testing. The field velocity is generated from a 2D gaussian random field.

In the following, we specify the spatial dependent velocity.

Advection0:

$$\begin{aligned} v_x(x, y) &= \sin\big(2\pi y + t\big), \\ v_y(x, y) &= \cos\big(2\pi x + t\big). \end{aligned} \tag{14}$$

Advection1:

$$\begin{aligned} v_x(x, y) &= U_0 \frac{y}{L_y}, \\ v_y(x, y) &= 0. \end{aligned} \tag{15}$$

Advection2:

$$\begin{aligned} v_x(x, y) &= \sin\Big(\frac{\pi x}{L_1}\Big)\,\cos\Big(\frac{\pi y}{L_2}\Big), \\ v_y(x, y) &= -\cos\Big(\frac{\pi x}{L_1}\Big)\,\sin\Big(\frac{\pi y}{L_2}\Big). \end{aligned} \tag{16}$$

Advection3:

$$\begin{aligned} v_x(x, y) &= -\big(y - \tfrac{L_2}{2}\big), \\ v_y(x, y) &= x - \tfrac{L_1}{2}. \end{aligned} \tag{17}$$

Advection4:

$$\begin{aligned} v_x(x, y, t) &= a\, x, \\ v_y(x, y, t) &= -a\, y. \end{aligned} \tag{18}$$

ComputeRHS$(\hat{u}, v_x, v_y)$: Compute the spectral right-hand side of $u_t + \nabla\cdot(\mathbf{v}\, u) = 0$ by 1. transforming $\hat{u}$ to u, 2. forming fluxes $u\, v_x$ and $u\, v_y$ and re-transforming, 3. differentiating in x, y via iK_X, iK_Y, 4. summing $-(\partial_x + \partial_y)$ in spectral space.

J.3 DATASETS FOR COMPARISON WITH DISCRETIZATION STRATEGY

We employ Smoothed Particle Hydrodynamics (SPH) (Gingold & Monaghan, 1977) to generate three Lagrangian particle subsets: the Taylor–Green vortex (TGV), the lid-driven cavity flow (LDC), and the reverse Poiseuille flow (RPF). Each subset is governed by the compressible N-S equations. For Taylor–Green vortex, we set the Reynolds number $\mathrm{Re} = 100$ and simulate on a 1×1 periodic domain with spatial resolution $\Delta x = 0.01$ and time step $\Delta t = 4 \times 10^{-4}$. We generate 100 sequences for training and 50 sequences for testing, each comprising 126 time steps, using $N = 10,000$ particles. For Lid-driven cavity, we also use the same domain and discretization parameters ($\Delta x = 0.01$, $\Delta t = 4 \times 10^{-4}$) at $\mathrm{Re} = 100$, producing 10k frames for training and 5k frames for testing with $N = 11,236$ particles. For Reverse Poiseuille flow, Re is set as 10 on a 1.12×1.12 periodic domain with $\Delta x = 1.25 \times 10^{-2}$ and $\Delta t = 5 \times 10^{-4}$. We generate 20k training frames and 5k testing frames with $N = 12,800$ particles. For all three datasets, particle positions are recorded every 100 time steps, so that the neural models are trained to predict the flow evolution over intervals of $100\Delta t$.

K VISUALIZATION STUDY

In this section, we provide more visualizations for the simulated fluid and showcases as a supplement. We choose some specific methods and fluid types for demonstration. For example, we show the visualization results of Fourier-based methods on Compressible N-S in Figure 14, 15 and 16 for prediction, ground truth and residual gap respectively. Besides, we also provide results of the Fourier-based method on Diffusion-Reaction in Figure 17. And visualizations of three different methods on Stochastic N-S are shown in Figure 18, 19, and 20.

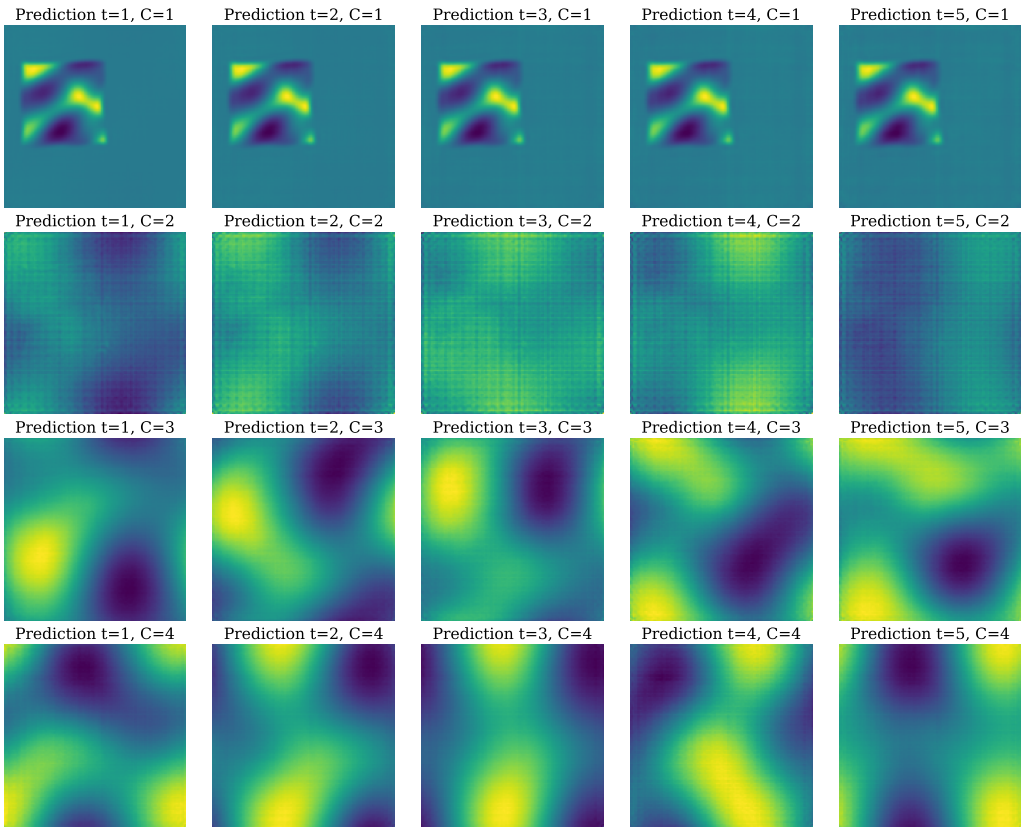


Figure 14: Visualization of Fourier+Next-step+Variable on Compressible N-S.

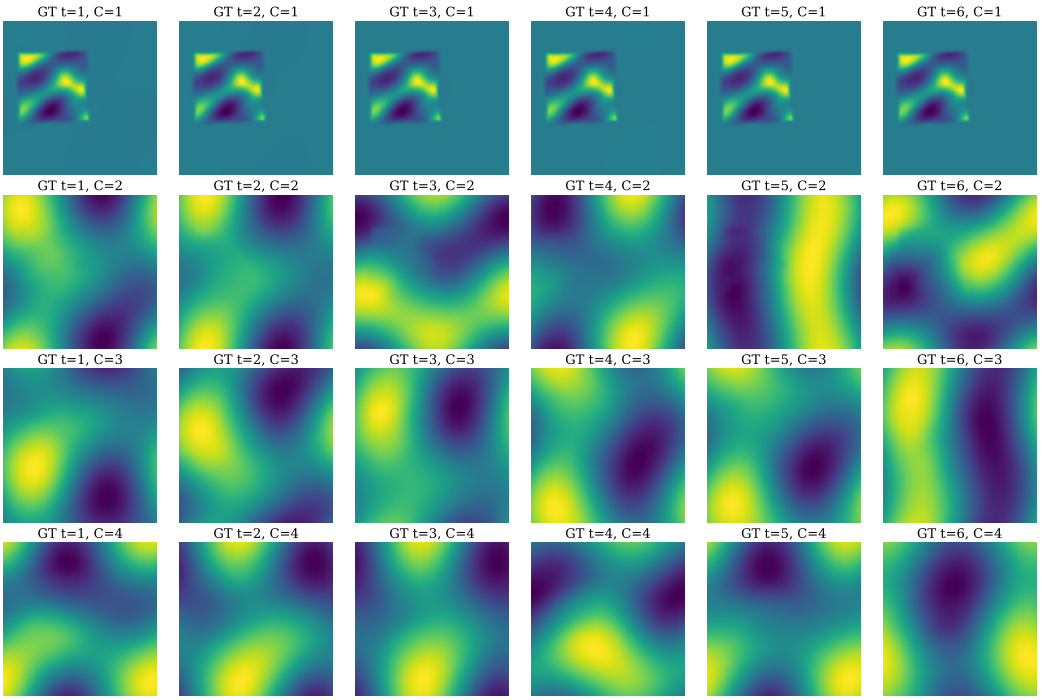


Figure 15: Visualization of Fourier+Next-step+Variable on Compressible N-S.

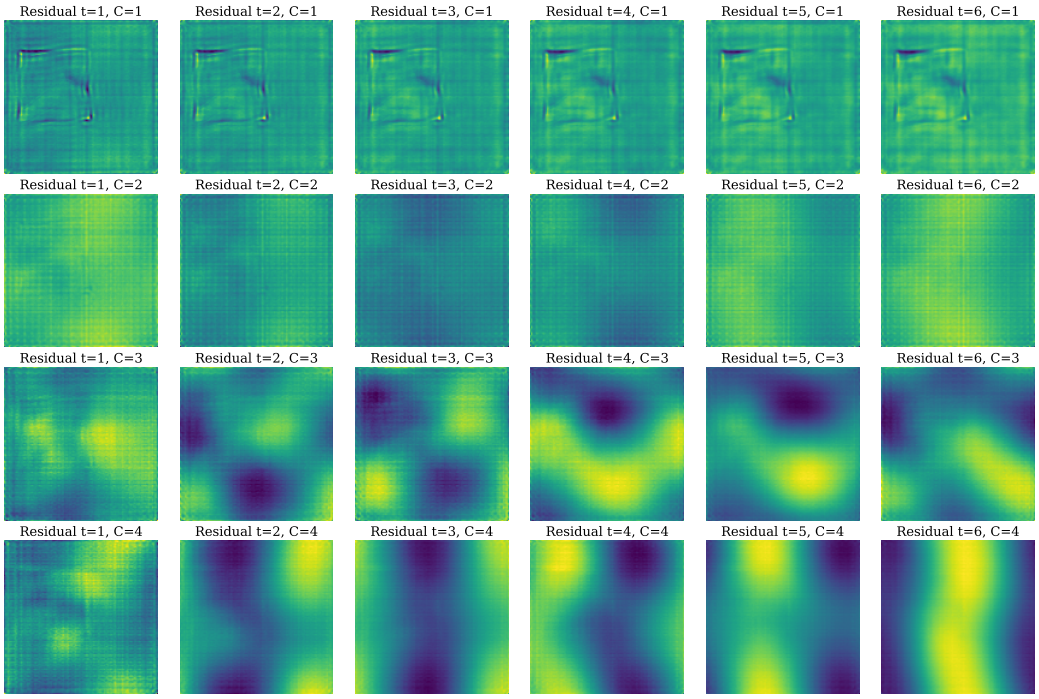


Figure 16: Visualization of Fourier+Next-step+Variable on Compressible N-S.

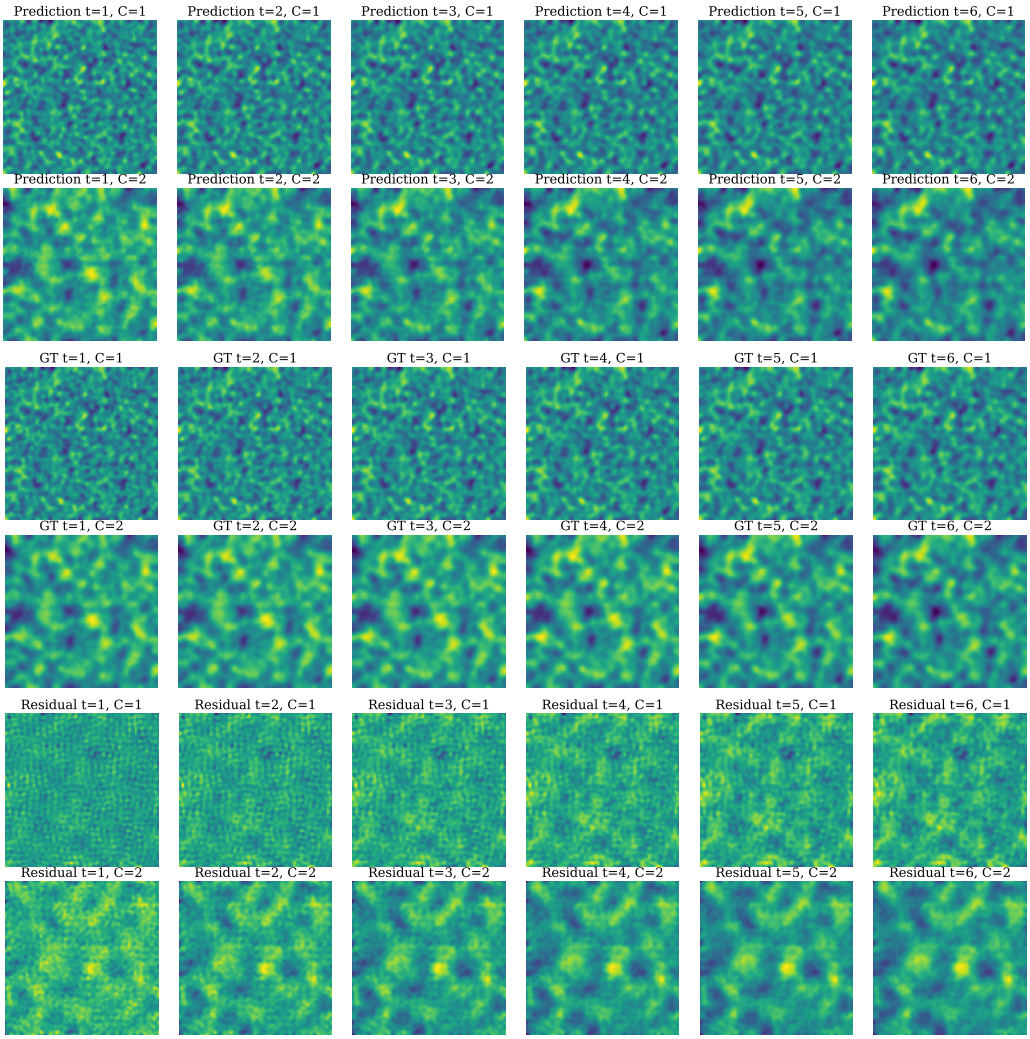


Figure 17: Visualization of Fourier+Next-step+Variable on Diffusion-Reaction.

L Limitations and Broad Impact

Despite our efforts to cover a diverse set of architectures, our benchmark is limited by time constraints and thus does not yet include all possible representation paradigms (*e.g.*, higher-order spectral methods, graph-wavelet embeddings, or learned Lagrangian bases). Expanding to these and other emerging modalities remains an important direction for future work.

Another limitation is that our experiments are mainly based on 2D PDEs. Prior works (Li et al.; 2023a) have shown that the same operator-based architectures scale to 3D turbulence, but at significantly higher costs. Our focus here is to first establish a fair, reproducible 2D benchmark, which we view as a necessary prerequisite for systematic 3D evaluation. Although our current experiments focus on 2D systems, the modular decomposition and evaluation pipeline are dimension-agnostic and can be directly extended to 3D settings, subject to available computational resources.

Nonetheless, we have released our full evaluation pipeline as a modular, well-documented codebase that makes it straightforward to incorporate new model classes, data modalities, or loss functions. This design ensures that researchers can readily extend the benchmark, plugging in novel representations and comparators with minimal effort. In this way, our framework lays the groundwork for a fair,

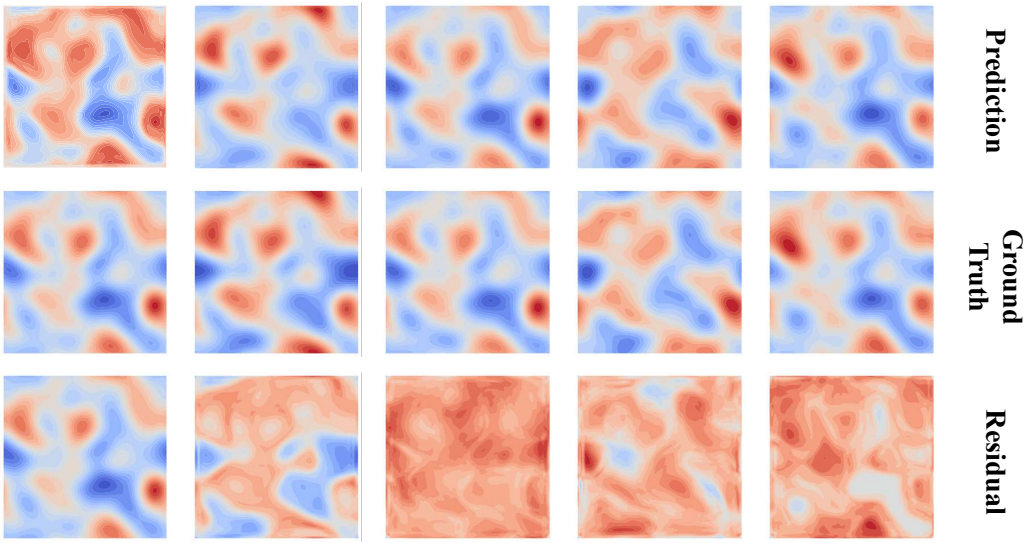


Figure 18: Visualization of Conv+Next-step+Variable on Stochastic N-S .

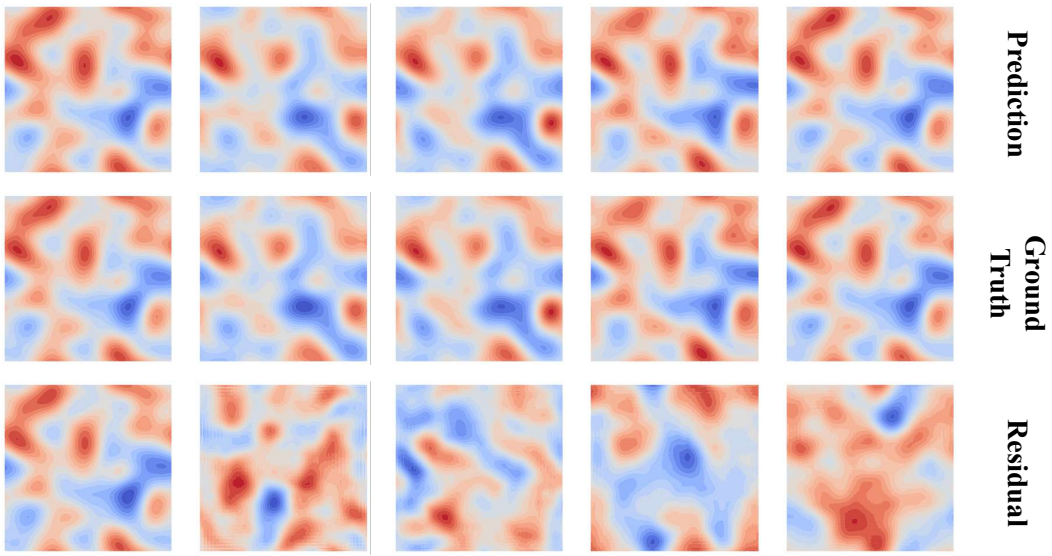


Figure 19: Visualization of Fourier+Next-step+Variable on Stochastic N-S .

reproducible, and easy-to-use codebase and standard for comparing data-driven fluid simulation methods going forward.

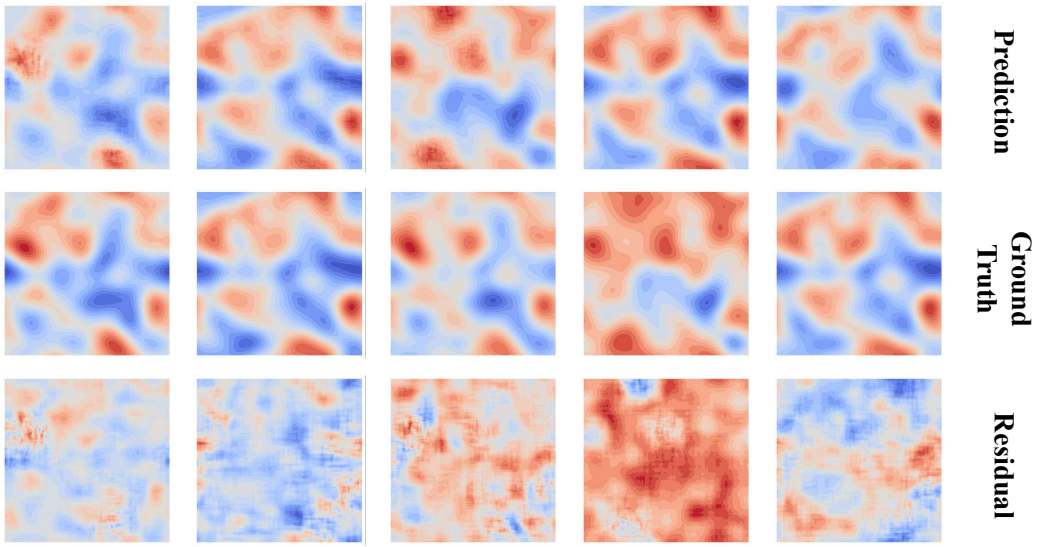


Figure 20: Visualization of Latent+Next-step+Variable on Stochastic N-S .