Physics-Informed Neural PDE Solvers via Spatio-Temporal MeanFlow

Hanru Bai [1] **Yuncheng Zhou** [1] **Difan Zou** [2]

Abstract

Deep learning paradigms, such as PINNs and neural operators, have significantly advanced the solving of PDEs. However, they often struggle to capture the continuous integral nature of physical systems, relying either on pointwise residuals that ignore the integral perspective or on pre-discretized temporal grids. Drawing inspiration from MeanFlow, a continuous-time integrator recently developed to efficiently solve generative ODEs, we introduce Spatio-Temporal MeanFlow, which functions as a novel PDE solver learning the finite-interval evolution of physical states. By substituting the generative velocity field with the physical PDE operator, we transform multi-step numerical integration into an efficient prediction with a freely controllable integration length. Crucially, we extend the original MeanFlow constraint from the temporal to the spatio-temporal domain, coupling time evolution with spatial consistency. This yields a unified framework naturally accommodating both time-dependent and stationary PDEs. Comprehensive experiments on benchmarks demonstrate that our approach achieves superior accuracy and inference efficiency over representative baselines. Furthermore, the proposed integral constraint enables excellent generalization to out-of-distribution initial conditions and varying spatial resolutions.

1. Introduction

Solving partial differential equations (PDEs) efficiently and accurately is fundamental to modeling complex physical phenomena across science and engineering, such as shock wave propagation and fluid turbulence (Bandrauk et al., 2007; Madhavi, 2025; Glimm, 1991). Fundamentally, this mathematical process is a continuous integration process. For example, for a time-dependent PDE governed by $\frac{\partial \boldsymbol{u}}{\partial t} = \boldsymbol{f}[\boldsymbol{u}, \nabla, \boldsymbol{x}, t]$, the evolution of the state field $\boldsymbol{u}(\boldsymbol{x}, t)$ across the continuous spatial domain from time τ to t is the integral:

$$\boldsymbol{u}(\boldsymbol{x}, t) = \boldsymbol{u}(\boldsymbol{x}, \tau) + \int_{\tau}^{t} \boldsymbol{f}[\boldsymbol{u}(\boldsymbol{x}, s), \nabla, \boldsymbol{x}, s] \, \mathrm{d}s. \quad (1)$$

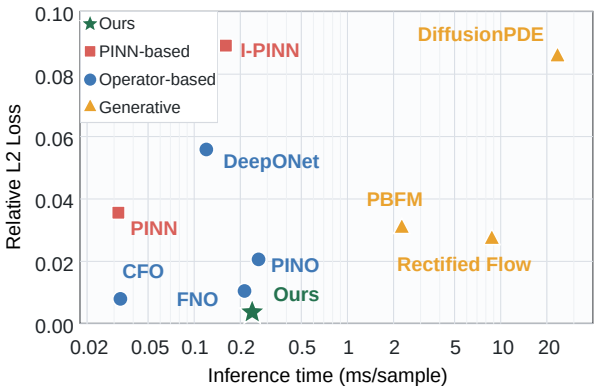

Figure 1. Accuracy-efficiency trade-off on the Burgers benchmark. Our Spatio-Temporal MeanFlow achieves an optimal Pareto frontier (bottom-left), delivering the lowest relative L_2 error with highly competitive inference speeds across all baselines.

While the two dominant deep learning paradigms for PDE solving, namely Physics-Informed Neural Networks (PINNs) (Cuomo et al., 2022), which embed governing equations as pointwise loss penalties, and neural operators like FNO (Li et al., 2020a), which learn data-driven mappings between function spaces, have achieved remarkable success, they struggle to fully capture the continuous integral process defined in Eq. (1). Specifically, FNO relies heavily on pre-discretized time slices and treats the integration interval as a fixed hyperparameter, essentially functioning as a discrete, step-by-step integrator. Conversely, PINNs rely solely on the pointwise PDE formula itself, bypassing the integral perspective entirely. Grounded in the formulation of Eq. (1), we argue that a more straightforward approach is to directly train a neural integrator capable of operating flexibly in continuous domain.

A similar challenge has been extensively studied in generative modeling, such as diffusion models (Croitoru et al., 2023; Yang et al., 2023; Luo, 2022) and flow matching (Lip-

[1]Fudan University [2]The University of Hong Kong. Correspondence to: Difan Zou <dzou@cs.hku.hk>.

Proceedings of the AI4Physics Workshop at the 43rd International Conference on Machine Learning (AI4Physics@ICML 2026), Seoul, South Korea. 2026. Copyright 2026 by the author(s).

man et al., 2022), where the inference process can also be formulated as an ordinary differential equation (ODE). Consequently, several works strive to develop efficient methods to solve this generative ODE (Liu et al., 2022; Song et al., 2023). One notable model is MeanFlow (Geng et al., 2025), which directly learns the average velocity over a finite interval using neural networks. By doing so, it essentially functions as a continuous-time integrator whose core advantage lies in bypassing repeated velocity evaluations, transforming multi-step integration into an efficient, single-step prediction. This provides a powerful intuition: can we adapt this analogous mechanism to construct a continuous-time neural integrator dedicated to serving PDE solving?

Inspired by this, we adapt the continuous integral formulation of MeanFlow to PDE solving. Specifically, we replace the generative velocity field $\boldsymbol{v}$ in the MeanFlow identity with the PDE right-hand side $\boldsymbol{f}$, and use the resulting identity to parameterize the finite-interval evolution of PDE solutions. The original MeanFlow constraint characterizes temporal evolution alone. For PDEs, however, generalization also depends critically on spatial consistency. We therefore extend the MeanFlow loss from the temporal domain to the spatio-temporal domain, introducing a spatio-temporal MeanFlow loss tailored to PDE solving that strengthens the model's ability to generalize across spatial locations and resolutions. Furthermore, this formulation naturally accommodates static PDEs by specializing to a purely spatial MeanFlow loss, providing a unified framework for both time-dependent and stationary PDEs. By transcending purely data-driven learning, the model achieves superior generalization to out-of-distribution initial conditions and varied spatial resolutions. Our contributions are:

- **A MeanFlow-inspired PDE solver framework**: We develop a novel PDE solver that enables efficient inference by transforming conventional multi-step numerical integration into finite-interval predictions, with the integration length freely controllable at inference time.
- **A novel spatio-temporal MeanFlow loss**: We introduce a physics-informed constraint that spatio-temporal significantly enhances solving accuracy and generalization under out-of-distribution initial conditions and different spatial resolutions.
- **Comprehensive experimental validation**: We conduct extensive evaluations on canonical PDE benchmarks and a real-world dataset. As shown in Fig. 1, our method achieves an optimal accuracy-efficiency trade-off across synthetic and real-world PDEs. Furthermore, extensive ablations validate our core components, demonstrating the integration-step flexibility of our continuous formulation and showing how the spatio-temporal MeanFlow losses drive robust generalization to higher resolutions and out-of-distribution (OOD) conditions.

2. Preliminaries

MeanFlow. For states $\boldsymbol{u}_t$ ($\boldsymbol{z}_t$ in (Geng et al., 2025)), MeanFlow (Geng et al., 2025) achieves one-step prediction by modeling the average velocity $\boldsymbol{m}$ ($\boldsymbol{u}$ in (Geng et al., 2025)) along ODE $\frac{\mathrm{d}\boldsymbol{u}_t}{\mathrm{d}t} = \boldsymbol{v}(\boldsymbol{u}_t, t)$: $\boldsymbol{m}(\boldsymbol{u}_t, \tau, t) \triangleq \frac{1}{t-\tau}\int_\tau^t \boldsymbol{v}(\boldsymbol{u}_s, s)\,\mathrm{d}s$ between time τ (r in (Geng et al., 2025)) and t, rather than multi-step integration in Flow Matching. Then, by taking the total derivative with respect to time t on both sides, we obtain the **MeanFlow Identity** in Eq. 2, whose right-hand side serves as the training objective for network $\boldsymbol{m}_\theta(\boldsymbol{u}_t, \tau, t)$. (derived in App. A)

$$\boldsymbol{m}(\boldsymbol{u}_t, \tau, t) = \boldsymbol{v}(\boldsymbol{u}_t, t) - (t-\tau)\tfrac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{m}(\boldsymbol{u}_t, \tau, t). \quad (2)$$

Problem Formulation. We formulate the PDE system over a spatial domain $D \subset \mathbb{R}^n$ and time $t \in [0, T]$ as follows:

$$\gamma\tfrac{\partial \boldsymbol{u}(\boldsymbol{x},t)}{\partial t} = \boldsymbol{f}[\boldsymbol{u}(\boldsymbol{x},t), \nabla, \boldsymbol{x}, t, \boldsymbol{a}], \boldsymbol{x} \in D,\ t \in (0, T], \quad (3)$$

$$\boldsymbol{u}(\boldsymbol{x}, 0) = \boldsymbol{u}_0(\boldsymbol{x}), \boldsymbol{x} \in D, \quad (4)$$

$$\mathcal{B}[\boldsymbol{u}(\boldsymbol{x},t)] = \boldsymbol{g}(\boldsymbol{x},t), \boldsymbol{x} \in \partial D,\ t \in [0, T]. \quad (5)$$

Here, $\boldsymbol{u} \in \mathbb{R}^d$ is the state variable, $\boldsymbol{f}$ is the differential operator with parameters $\boldsymbol{a}$, while $\boldsymbol{u}_0$ and $\mathcal{B}$ denote the initial condition and boundary operator, respectively. The coefficient $\gamma \in \mathbb{R}$ unifies time-variant ($\gamma \neq 0$) and time-invariant ($\gamma = 0$) PDEs. Notably, for static systems ($\gamma = 0$), the time dimension t acts as a dummy variable, yielding a constant state $\boldsymbol{u}(\boldsymbol{x}, t) \equiv \boldsymbol{u}_{\text{static}}(\boldsymbol{x})$ along the temporal axis, which naturally accommodates our unified spatio-temporal modeling.

3. Methodology

This section presents the Spatio-Temporal MeanFlow framework. Sec. 3.1 and 3.2 formulate the problem via integration paths and derive physics-informed governing constraints. To mitigate temporal-spatial scale mismatches, Sec. 3.3 introduces a decoupling strategy with a theoretical decoupling bound. Sec. 3.4 integrates these into a unified objective function and details the specific algorithms. Finally, Sec. 3.5 establishes theoretical foundations, providing rigorous justifications for consistency, a-posteriori error bounds, and improved optimization dynamics.

3.1. Spatio-Temporal MeanFlow Formulation

We consider the general partial differential equation (PDE) system defined by Eq. 3. To relate the states between a

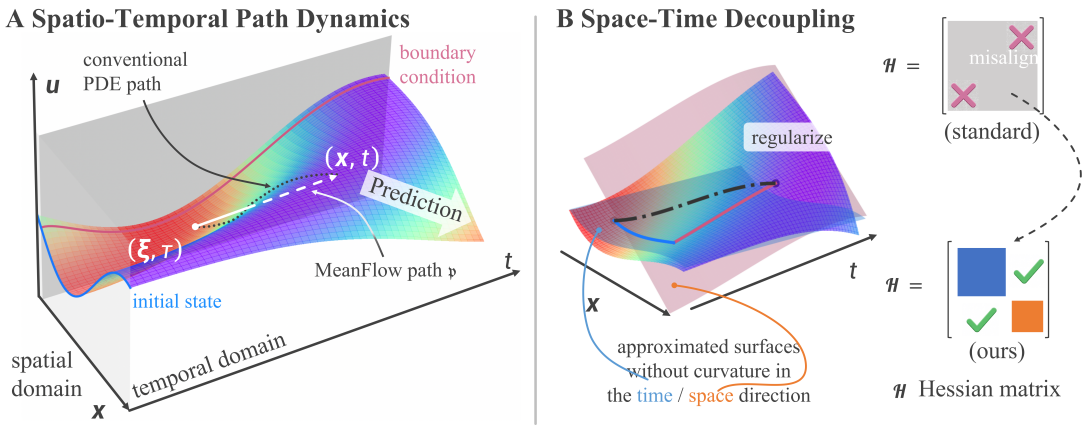

Figure 2. Overview of our framework. (**A**) Spatio-Temporal Path Dynamics, formulating PDE solutions via straight paths. (**B**) Space-Time Decoupling, constraining the surface in space or time direction only, designed to mitigate scale mismatches within the Hessian matrix.

starting point $(\boldsymbol{\xi}, \tau)$ and a target point $(\boldsymbol{x}, t)$, we define a spatio-temporal flow path $\mathfrak{p} : [0,1] \mapsto D \times [0,T]$ such that $\mathfrak{p}(0) = (\boldsymbol{\xi}, \tau)$ and $\mathfrak{p}(1) = (\boldsymbol{x}, t)$ (Fig. 2 A). We then define the spatio-temporal MeanFlow function $\boldsymbol{m}$ as the integral average of the state evolution along path $\mathfrak{p}$:

$$\boldsymbol{m}[\boldsymbol{u}(\boldsymbol{\xi}, \tau), \mathfrak{p}, \boldsymbol{a}] \triangleq \int_{\mathfrak{p}} \mathrm{d}\boldsymbol{u}(\mathfrak{p}) \;/\; \int_{\mathfrak{p}} \|\,\mathrm{d}\mathfrak{p}\|_2. \tag{6}$$

For a convex domain D, we consider a straight path $\mathfrak{p}(s) \triangleq [\boldsymbol{\xi} + s(\boldsymbol{x} - \boldsymbol{\xi}), \tau + s(t - \tau)]$ and define path length $l(\boldsymbol{\xi}, \tau, \boldsymbol{x}, t) \triangleq \int_{\mathfrak{p}} \|\mathrm{d}\mathfrak{p}\|_2 = \sqrt{\|\boldsymbol{x} - \boldsymbol{\xi}\|_2^2 + (t - \tau)^2}$ which is the total spatio-temporal Euclidean distance. Following the Fundamental Theorem of Line Integrals, Eq. 6 satisfies

$$l(\boldsymbol{\xi}, \tau, \boldsymbol{x}, t) \cdot \boldsymbol{m}[\boldsymbol{u}(\boldsymbol{\xi}, \tau), \boldsymbol{\xi}, \tau, \boldsymbol{x}, t; \boldsymbol{a}] = \boldsymbol{u}(\boldsymbol{x}, t) - \boldsymbol{u}(\boldsymbol{\xi}, \tau). \tag{7}$$

Note that using a straight path requires D to be convex in classical analysis, as it might temporarily exit the physical boundary of non-convex domains (e.g., fluid flow around obstacles). Fortunately, parameterized neural networks act as global continuous functions, hence implicitly perform a *smooth analytic continuation* of the physical state field into the convex hull of D.

In our formulation, the proposed path integration and differentiation operators rely on two fundamental mathematical prerequisites: 1) The state variable $\boldsymbol{u}$ is sufficiently smooth. Specifically, we require $\boldsymbol{u} \in \mathcal{C}^1(D \times [0,T])$ for first-order PDEs, and $\boldsymbol{u} \in \mathcal{C}^2$ for systems involving second-order spatial derivatives. This guarantees that the Jacobian-based operators (e.g., $\boldsymbol{K}$ in Sec. 3.2) are well-defined; 2) The domain is topologically connected to ensure the physical meaningfulness of bridging any two states.

3.2. Physics-informed MeanFlow Constraints

By differentiating the fundamental identity Eq. (7) with respect to the initial coordinates $(\boldsymbol{\xi}, \tau)$, we derive the physics-informed constraints for our framework, i.e.,

$$l_t \cdot \boldsymbol{m} = l\boldsymbol{K}\boldsymbol{f} + \gamma l^2 \cdot \tfrac{\partial \boldsymbol{m}}{\partial \tau}, \tag{8}$$

$$\begin{cases} l \cdot \boldsymbol{K} \cdot \nabla^\top \boldsymbol{u}(\boldsymbol{\xi}, \tau) = \boldsymbol{m}(\boldsymbol{x} - \boldsymbol{\xi})^\top - l^2 \cdot \dfrac{\partial \boldsymbol{m}}{\partial \boldsymbol{\xi}}, \\ l \cdot \boldsymbol{K} \cdot \Delta \boldsymbol{u}(\boldsymbol{\xi}, \tau) = -(\boldsymbol{\alpha} + 2\boldsymbol{K}^{-1}\boldsymbol{\beta}), \quad \cdots . \end{cases} \tag{9}$$

Here, $\boldsymbol{\alpha} \triangleq n\boldsymbol{m} - (l_s^2 \,/\, l^2) \cdot \boldsymbol{m} + 2l^2 \Delta \boldsymbol{m}(\boldsymbol{\xi}, \tau)$ and $\boldsymbol{\beta} \triangleq -\frac{\partial \boldsymbol{m}}{\partial \boldsymbol{\xi}}(\boldsymbol{x} - \boldsymbol{\xi}) - (l_s^2 / l) \cdot \frac{\partial \boldsymbol{m}}{\partial \boldsymbol{u}} \boldsymbol{m}$, with $l_t = t - \tau$ being the time step, $l_s = \|\boldsymbol{x} - \boldsymbol{\xi}\|_2$ the spatial distance, and $\boldsymbol{K} = \boldsymbol{I} + l \frac{\partial \boldsymbol{m}}{\partial \boldsymbol{u}}$ the Jacobian-based correction operator. Specifically, Eq. 8 is derived by differentiating Eq. (7) with respect to time τ, while Eq. 9 are obtained by taking spatial derivatives (first-order, second-order, and so forth) with respect to $\boldsymbol{\xi}$. Prop. D.1, Appendix D provides the detailed derivation. Consequently, the spatio-temporal MeanFlow loss is defined as

$$\mathcal{L}_{\text{ST-MeanFlow}} \triangleq \|l_t \cdot \boldsymbol{m} - l\boldsymbol{K}\boldsymbol{f} - \gamma l^2 \cdot \tfrac{\partial \boldsymbol{m}}{\partial \tau}\|_2^2, \tag{10}$$

with the gradient terms in $\boldsymbol{f}$ substituted by Eqs. (9).

3.3. Space-Time Decoupling Strategy

In many physical systems, temporal evolution and spatial variations occur at vastly different magnitudes (e.g., fast chemical reactions in large-scale reactors) (Wang et al., 2021; Krishnapriyan et al., 2021). When these dimensions are tightly coupled in a single loss function, the optimization landscape often exhibits pathological curvature, characterized by a highly ill-conditioned Hessian matrix (Wang et al., 2022) with large off-diagonal cross-coupling terms (Fig. 2 B). To solve the challenge of inherent *spatio-temporal scale mismatch*, we decompose the spatio-temporal objective into two independent first-order components:

Temporal MeanFlow Loss. $\mathcal{L}_{\text{T-MeanFlow}} \triangleq \left\|\boldsymbol{m} - \boldsymbol{K}\boldsymbol{f} - \gamma l_t \frac{\partial \boldsymbol{m}}{\partial \tau}\right\|_2^2$ constrains the network to satisfy

Algorithm 1 Training

Require: PDE params $\boldsymbol{a}$, temporal indicator γ, network $\boldsymbol{m}_\theta$, learning rate η
for each training iteration **do**
  if $\gamma \neq 0$ **then** ▷ Time-dependent PDEs
    Sample time pairs $\tau < t$ and grid $\boldsymbol{x}$;
    Get input: $\boldsymbol{I} \leftarrow [\boldsymbol{u}(\boldsymbol{x}, \tau), \tau, \boldsymbol{x}, t, \boldsymbol{a}]$;
    Predict MeanFlow: $\boldsymbol{m} \leftarrow \boldsymbol{m}_\theta(\boldsymbol{I})$;
    Compute $\boldsymbol{u}(\boldsymbol{x}, t) \leftarrow \boldsymbol{u}(\boldsymbol{x}, \tau) + (t - \tau)\boldsymbol{m}$;
    Compute Loss: $\mathcal{L}_{\text{Total}}$;
  else ▷ Static PDEs ($\gamma = 0$)
    Analytically extend boundary conditions to the entire domain as $\boldsymbol{u}_0(\boldsymbol{x})$;
    Get input: $\boldsymbol{I} \leftarrow [\boldsymbol{u}_0(\boldsymbol{x}), \boldsymbol{a}]$;
    Predict MeanFlow: $\boldsymbol{m} \leftarrow \boldsymbol{m}_\theta(\boldsymbol{I})$;
    Construct state: $\boldsymbol{u}(\boldsymbol{x}) \leftarrow \boldsymbol{u}_0(\boldsymbol{x}) + \boldsymbol{m}$;
    Compute Loss: $\mathcal{L}_{\text{Total}} \leftarrow \mathcal{L}_{\text{Data}} + \lambda_s \mathcal{L}_{\text{S-MF}}$;
  end if
  Update parameters: $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}_{\text{Total}}$;
end for
return $\theta^* = \theta$;

the temporal evolution predicted by the Spatio-Temporal MeanFlow identity along the time-line.

Spatial MeanFlow Loss. $\mathcal{L}_{\text{S-MeanFlow}} \triangleq \left\| l_s \boldsymbol{K} \nabla^\top \boldsymbol{u} - \boldsymbol{m}(\boldsymbol{x} - \boldsymbol{\xi})^\top + l_s^2 \frac{\partial \boldsymbol{m}}{\partial \boldsymbol{\xi}} \right\|_2^2$ enforces consistency between predicted Spatio-Temporal MeanFlow's spatial derivatives and numerical spatial gradients.

Thm. 3.2, Sec. 3.5 ensures that the decoupling provide a theoretical reference for the overall Spatio-Temporal MeanFlow loss. The core advantage of this decoupling is the removal of explicit spatio-temporal cross-derivatives in the residual definitions. Mathematically, this transformation reshapes the Hessian into a nearly block-diagonal structure, reducing its condition number and enabling faster, more robust convergence (Thm. D.4, App. D.4). This decoupling is theoretically supported by the *Space-Time Decoupling Bound* presented in the following section, which ensures that minimizing these independent losses effectively constrains the global spatio-temporal error.

3.4. Detailed Implemention

Training. We parameterize the MeanFlow predictor $\boldsymbol{m}$ using a neural network, denoted as $\boldsymbol{m}_\theta$, to approximate the true $\boldsymbol{m}$. For time-dependent PDEs, the network is designed to map the concatenated input $[\boldsymbol{u}(\boldsymbol{x}, \tau), \tau, \boldsymbol{x}, t, \boldsymbol{a}]$, where τ and t are randomly sampled time pairs during training. The model outputs the average integral from time τ to t. According to the fundamental identity (Eq. (7)), the target state at time t is explicitly constructed as $\boldsymbol{u}(\boldsymbol{x}, t) = \boldsymbol{u}(\boldsymbol{x}, \tau) + (t - \tau) \cdot \boldsymbol{m}_\theta$. The total objective function is defined as the weighted sum of the data loss (measuring the discrepancy between the predicted solution at time t and observational data) and the physics-informed decoupled

Algorithm 2 Multi-step Inference

Require: Pretrained network $\boldsymbol{m}_{\theta^*}$, initial state $\boldsymbol{u}_0 = \boldsymbol{u}(\boldsymbol{x}, 0)$, PDE params $\boldsymbol{a}$, target time T, steps N
Calculate time step: $\Delta t \leftarrow T/N$;
Initialize trajectory list with $\boldsymbol{u}_0$;
Initialize current time: $t_0 \leftarrow 0$;
for $k = 0$ to $N - 1$ **do**
  Target time: $t_{k+1} \leftarrow t_k + \Delta t$;
  Get input: $\boldsymbol{I}_k \leftarrow [\boldsymbol{u}_k, t_k, \boldsymbol{x}, t_{k+1}, \boldsymbol{a}]$;
  Predict average integral (MeanFlow): $\boldsymbol{m}_k \leftarrow \boldsymbol{m}_{\theta^*}(\boldsymbol{I}_k)$;
  Update state via continuous integral: $\boldsymbol{u}_{k+1} \leftarrow \boldsymbol{u}_k + \Delta t \cdot \boldsymbol{m}_k$;
  Append $\boldsymbol{u}_{k+1}$ to trajectory list;
end for
return $\boldsymbol{u}_T$;

spatio-temporal MeanFlow losses:

$$\mathcal{L}_{\text{Total}} = \mathcal{L}_{\text{Data}} + \lambda_t \mathcal{L}_{\text{T-MeanFlow}} + \lambda_s \mathcal{L}_{\text{S-MeanFlow}}, \quad (11)$$

where λ_t and λ_s are the weighting coefficients.

Conversely, when $\gamma = 0$, the system degenerates into a static PDE. In this scenario, temporal variables are omitted. The model instead takes the analytically extended boundary conditions $\boldsymbol{u}_0(\boldsymbol{x})$ (where $\boldsymbol{u}_0|_{\partial D} = \boldsymbol{u}|_{\partial D}$) and the coefficient field $\boldsymbol{a}$ as inputs. The solution over the interior domain $\boldsymbol{x} \in D \setminus \partial D$ is constructed as $\boldsymbol{u}(\boldsymbol{x}) = \boldsymbol{u}_0(\boldsymbol{x}) + \boldsymbol{m}_\theta$, where $\boldsymbol{m}_\theta$ represents the network output. The temporal loss naturally vanishes, and the training objective simplifies to: $\mathcal{L}_{\text{Total}} = \mathcal{L}_{\text{Data}} + \lambda_s \mathcal{L}_{\text{S-MeanFlow}}$. Alg. 1 detail the training optimization procedures of the proposed framework.

Note that in our practical implementation, considering that the benchmark PDEs evaluated in this work are defined on regular grids, we maintain fixed spatial coordinates ($\boldsymbol{x}$) across inputs and outputs. Consequently, our current network design predominantly emphasizes continuous-time evolution. However, our fundamental mathematical formulation inherently supports continuous spatial evolution as well, where the anchor point $\boldsymbol{\xi}$ and the target point $\boldsymbol{x}$ can vary continuously. The detailed algorithms for the fully generalized spatio-temporal framework are provided in App. C.

Inference. We first focus on the one-step inference scenario. For time-dependent PDEs, standard deep learning-based solvers typically take the initial condition and physical coefficients as inputs to predict the solution over the spatial domain at a target time t. Following this paradigm, our algorithm takes $[\boldsymbol{u}(\boldsymbol{x}, 0), 0, \boldsymbol{x}, t, \boldsymbol{a}]$ as input during inference to predict the average integral from time 0 to t, denoted as $\boldsymbol{m}_{\boldsymbol{\theta}}(\boldsymbol{x}, t)$. The final solution at time t is then recovered via

$$\boldsymbol{u}(\boldsymbol{x}, t) = \boldsymbol{u}(\boldsymbol{x}, 0) + t \cdot \boldsymbol{m}_{\boldsymbol{\theta}}(\boldsymbol{x}, t). \quad (12)$$

For static systems, the general objective is to derive the global solution given specific boundary conditions and co-

Table 1. Results for forward PDE solving on four synthetic benchmarks and one real-world dataset, alongside the inverse problem on Darcy flow. We report the relative L_2 error (↓) as the mean ± standard deviation across three different random seeds. All reported metrics are multiplied by 10^{-2} for readability. Results for CFO on steady-state PDEs are omitted, as it is designed for time-dependent system. The best results are highlighted in **bold**, while the second-best results are underlined.

Method ($\times 10^{-2}$)	Synthetic Forward PDEs				Real-world	Inverse
	Burgers	Navier–Stokes	Darcy	Poisson	HYCOM	Darcy
Physics-informed coordinate-based solvers						
PINN	3.56±0.42	40.21±0.20	15.98±1.55	32.01±4.69	28.12±0.24	27.41±0.01
I-PINN	8.92±3.19	35.65±0.15	8.41±0.12	54.64±4.09	33.05±0.45	8.95±0.01
Neural operator solvers						
DeepONet	5.59±0.07	19.68±1.14	2.92±0.02	5.46±0.33	34.23±0.54	8.73±0.26
FNO	1.05±0.07	0.38±0.00	0.46±0.00	0.23±0.01	24.08±0.79	1.40±0.00
PINO	2.06±0.18	0.32±0.00	16.30±0.03	1.58±0.00	23.79±0.11	12.19±0.32
CFO	0.80±0.15	0.88±0.00	–	–	32.75±4.26	17.07±1.01
Generative PDE solvers						
DiffusionPDE	8.61±0.31	118.74±0.95	5.48±1.75	0.76±0.03	180.89±0.72	4.49±0.94
Rectified Flow	2.76±0.06	1.40±0.28	0.71±0.07	1.58±0.00	31.80±0.21	36.95±1.88
PBFM	3.11±0.60	137.45±15.35	0.62±0.01	28.56±0.07	83.86±5.43	**1.02±0.03**
Ours	**0.26±0.00**	**0.19±0.02**	**0.25±0.00**	**0.18±0.02**	**23.43±0.12**	2.72±0.00

efficient fields. Maintaining consistency with our training phase, the model simply takes the analytically extended boundary conditions and the coefficient field as inputs and then get the solution across the spatial domain through $\boldsymbol{u}(\boldsymbol{x}) = \boldsymbol{u}_0(\boldsymbol{x}) + \boldsymbol{m}_\theta$. Furthermore, for time-dependent PDEs, since our framework models the continuous integral over arbitrary time intervals, it naturally extends to multi-step inference. The detailed procedure for multi-step prediction is provided in Alg. 2.

3.5. Theoretical Results

The proposed framework is supported by two core theoretical pillars: a-posteriori error bounds for the predicted state (Thm. 3.1), and effectiveness of the decoupled loss terms (Thm. 3.2). App. D.2-D.4 provide proofs for them, with the improved optimization landscapes through decoupling (Thm. D.4).

For simplicity, we define $\boldsymbol{r}_{\text{MeanFlow}} \triangleq l_t\boldsymbol{m} - l\boldsymbol{K}\boldsymbol{f} - \gamma l^2 \frac{\partial \boldsymbol{m}}{\partial \tau}$ with higher-order spatial gradients in $\boldsymbol{h}$ substituted by spatial MeanFlow Eqs. (9), $\boldsymbol{r}_{\text{Temp}} \triangleq \boldsymbol{m} - \boldsymbol{K}\boldsymbol{f} - \gamma l_t \frac{\partial \boldsymbol{m}}{\partial \tau}$, and $\boldsymbol{r}_{\text{Spac}} \triangleq l_s \boldsymbol{K}\nabla^\top \boldsymbol{u} - \boldsymbol{m}(\boldsymbol{x} - \boldsymbol{\xi})^\top + l_s^2 \frac{\partial \boldsymbol{m}}{\partial \boldsymbol{\xi}}$ as the residual vectors for different MeanFlow losses.

Theorem 3.1 (A-posteriori Error Estimate)**.** *Assume the MeanFlow predictor $\boldsymbol{m}_\theta$ is $\mathcal{C}^1$-continuous. If the total training loss bounds the squared path residuals such that $\mathcal{L}_{MeanFlow} \leq \epsilon$, then the pointwise error between reconstructed state $\boldsymbol{u}_\theta$ and true solution $\boldsymbol{u}_{True}$ at any target point $(\boldsymbol{x}, t)$ is bounded by:*

$$\|\boldsymbol{u}_\theta(\boldsymbol{x}, t) - \boldsymbol{u}_{True}(\boldsymbol{x}, t)\|_2 \leq C\sqrt{\epsilon} + \mathcal{O}(l^2), \tag{13}$$

where C is a constant depending on the path length l and the Lipschitz constant of the PDE operator.

Thm. 3.1 provides a bridge between the optimization target and the physical accuracy, justifying the use of MeanFlow residuals as a proxy for solution quality.

Theorem 3.2 (Space-Time Decoupling Bound)**.** *Assume that the physical system satisfies regular linear gradient structure, whose differential function $\boldsymbol{f}$ can be split into $\nabla^\top \boldsymbol{u} \cdot \boldsymbol{g} + \boldsymbol{h}$ where $\boldsymbol{h}$ is irrelevant to first-order gradient $\nabla \boldsymbol{u}$. The global MeanFlow loss is bounded by space-time decoupled losses:*

$$\mathcal{L}_{MeanFlow} \leq c_1 \mathcal{L}_{Temp} + c_2 \mathcal{L}_{Spac} + \varepsilon \tag{14}$$

where the coefficients are defined as $c_1 = 3l_t^2$, and $c_2 = 3\left(\frac{l - l_t}{l_s}\|\boldsymbol{g}\|_{op}\right)^2$. Here, ε represents the structural coupling error and $\varepsilon = \mathcal{O}(l_s^2)$.

The theoretical validity of decoupled training is established in Thm. 3.2, which proves that the global spatio-temporal consistency is bounded by the individual temporal and spatial errors. This bound ensures that minimizing decoupled losses effectively controls the total physics-informed error. In addition, the gap between decoupled losses and spatio-temporal consistency is controlled by l_s, which can theoretically be infinitesimally small due to continuous modeling.

4. Experiments

In this section, we comprehensively evaluate our Spatio-Temporal MeanFlow framework. First, in Sec. 4.1, we

Table 2. Performance and computational cost on the Burgers benchmark. We report the relative L_2 error ($\times 10^{-2}$), inference time (milliseconds per sample), and GFLOPs per sample. Lower is better. The best results are highlighted with bold text and a gray background.

	Coord-based		*Neural operators*				*Generative*			
Metric	PINN	I-PINN	DeepONet	FNO	PINO	CFO	DiffPDE	RectFlow	PBFM	Ours
Error ($\times 10^{-2}$) ↓	3.56	8.92	5.59	1.05	2.06	0.80	8.61	2.76	3.11	**0.26**
Time (ms) ↓	**0.032**	0.161	0.120	0.213	0.263	0.033	23.487	8.753	2.263	0.239
GFLOPs ↓	0.0715	0.0764	**0.0004**	0.0012	0.2742	0.0007	662.6990	155.3370	12.9219	0.0118

validate its PDE solving accuracy and computational efficiency against representative baselines on four PDE benchmarks. Next, Sec. 4.2 presents extensive ablations to verify the effectiveness of our core components. Specifically, we demonstrate the integration-step flexibility enabled by our framework functioning as a continuous integrator over arbitrary time intervals, and show how the temporal and spatial MeanFlow losses drive generalization to higher resolutions and OOD initial conditions. Finally, Sec. 4.3 highlights the framework's broader applicability beyond synthetic forward prediction, showcasing its potential on real-world datasets and inverse problems.

4.1. Forward PDE Solving on Synthetic Benchmarks

Benchmarks. We first evaluate the effectiveness of the proposed method for PDE solving using four standard synthetic benchmarks. The benchmarks include two time-dependent PDEs, 1D Burgers' equation and the 2D Navier–Stokes equation, and two steady-state PDEs, 2D Darcy flow and the 2D Poisson equation. All datasets are generated using numerical solvers and are divided into training, validation, and test sets with a ratio of 7:2:1. Further details about the governing equations, numerical discretizations, and data generation configurations are provided in App. F.1.

Baseline Methods and Metrics. We compare our method with three categories of PDE solvers: (1) physics-informed coordinate-based methods: PINN (Cuomo et al., 2022), I-PINN (Niu et al., 2025); (2) neural operators: FNO (Li et al., 2020a), DeepONet (Lu et al., 2019), PINO (Li et al., 2024), CFO (Hou et al., 2026)); and (3) generative solvers: DiffusionPDE (Huang et al., 2024), Rectified Flow (Armegioiu et al., 2025), PBFM (Baldan et al., 2026). All compared methods are reproduced using their publicly released code. Further details are provided in App. F.6. Following (Huang et al., 2024), we report the relative L_2 error on the test set. This metric enables fair cross-dataset comparisons regardless of solution magnitudes.

Implementation Details. All experiments are implemented in PyTorch and conducted on NVIDIA GeForce RTX 4080 GPUs. We employ the Adam (Kingma & Ba, 2014) optimizer for training. The learning rates are set to 5.5×10^{-4}, 2.0×10^{-4}, and 3.0×10^{-4} for the Burgers' equation, NS equation, and steady-state PDEs. For the loss functions, the MeanFlow coefficients are consistently set to $\lambda_t = \lambda_s = 0.01$, with λ_t omitted for steady-state cases. The model checkpoint achieving the lowest validation error is used for testing. Detailed hyperparameters and network architectures are provided in App. F.3.

Main Results. Table 1 reports the forward prediction results on the four benchmarks. Our method achieves the best overall performance across both time-dependent and steady-state PDEs. On Burgers' equation and NS equation, the improvement indicates that directly learning finite-interval evolution can reduce temporal error accumulation compared with methods that rely only on supervised data fitting or pointwise physical residuals. On Darcy flow and Poisson equation, our method also obtains lower errors, showing that the spatial MeanFlow constraint provides an effective regularization. Figure 4 shows qualitative results on NS test samples. Ours closely matches the ground-truth vorticity fields across different time steps, capturing the main flow structures and temporal evolution. Additional visual comparisons with baseline methods and training curves for all loss components (the total loss, the data loss, the temporal and the spatial MeanFlow loss) are detailed in App. F.7.

Efficiency and Computational Cost Tab. 2 reports the detailed inference latency and computational cost on the Burgers benchmark. Inference time is measured in milliseconds per sample, and computational cost is measured in GFLOPs per sample. Our method achieves low prediction error with sub-millisecond inference and only 0.0118 GFLOPs per sample. Compared to generative PDE solvers, it is substantially more efficient as it avoids the costly multi-step sampling or ODE evaluations. Meanwhile, when compared to similarly fast neural operator methods, our framework delivers lower prediction errors.

4.2. Ablation Study

Effect of Spatio-Temporal MeanFlow losses. We first conduct an ablation study on the Burgers' equation at a base

Table 3. Ablation study on Burgers evaluating the effects of temporal and spatial MeanFlow losses, resolution generalization and out-of-distribution initial condition generalization. All models are trained at the base resolution $s = 128$ and evaluated on test sets generated with the same random seed for each setting. We report global relative L_2 error; lower is better.

Type	Test Setting	w/o Both	w/o T-MF	w/o S-MF	Ours
Resolution	$s = 128$	0.017561	0.003075	0.010821	0.002574
	$s = 256$	0.017562	0.003051	0.010820	0.002419
	$s = 512$	0.017558	0.002913	0.010818	0.002395
OOD IC	Smooth, $\gamma = 3.5$	0.011952	0.010559	0.009174	0.003732
	Low freq., $\tau = 10$	0.020556	0.002792	0.016768	0.001223
	Smooth+Amp., $\gamma = 3.0, \sigma = 61.25$	0.007475	0.003032	0.004114	0.001092
	Smooth+Low-τ, $\gamma = 3.5, \tau = 5$	0.008023	0.006197	0.005505	0.002187

resolution of $s = 128$ to test the effect of spatial MeanFlow losses (S-MF) and temporal MeanFlow losses (T-MF). We define four model variants based on their loss weight configurations: a baseline without Spatio-Temporal MeanFlow regularization (w/o both), models omitting either the temporal (w/o T-MF) or spatial component (w/o S-MF), and our full framework (Ours). As presented in Tab. 3, omitting either regularization term leads to an increase in predictive error, showing the effectiveness of both loss components. To further analyze the model's sensitivity, a further evaluation under various combinations of T-MF and S-MF coefficients is deferred to App. F.7.

Resolution Generalization. Benefiting from the proposed spatial MeanFlow losses, a key advantage of our framework is its potential to generalize across varying spatial resolutions. We therefore directly evaluate the model, trained exclusively at $s = 128$, on higher-resolution grids ($s = 256/512$) without any retraining. As shown in Tab. 3, our method achieves stable zero-shot generalization across varying resolutions. Notably, removing the spatial MeanFlow loss (w/o S-MF) leads to severe performance degradation, confirming its critical role in maintaining physical consistency across discretizations.

OOD Initial Condition Generalization. We further test generalization to out-of-distribution initial conditions by changing the Gaussian random field parameters used to generate the Burgers initial states, while keeping the resolution fixed. We consider smooth, low-frequency, smooth amplified, and smooth low-τ IC shifts. As Tab. 3 shows, our method's superior OOD performance confirms that spatio-temporal MeanFlow losses robustly handles initial-condition shifts.

Integration steps. Since our model learns an integral operator over arbitrary time intervals, inference does not require a fixed temporal discretization. Given an initial state and a target time, the model can either predict the full interval in a single step or compose several shorter integrations. We therefore evaluate the effect of the number of integration steps on Burgers dataset. As shown in Fig. 3, increasing the

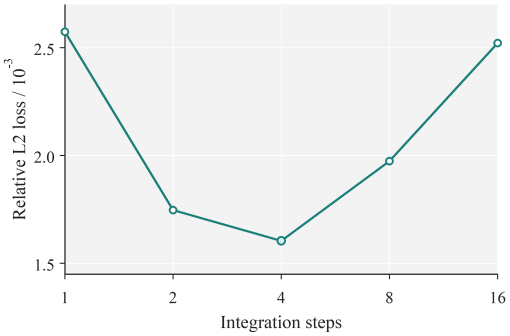

Figure 3. Relative L_2 error on Burgers with different numbers of integration steps.

number of steps from 1 to 4 reduces the relative L_2 error, showing that the learned operator can be effectively reused over sub-intervals and benefits from moderate temporal refinement. However, using too many steps degrades performance, likely due to accumulated approximation errors from repeated composition. This suggests that our framework provides a flexible efficiency–accuracy trade-off: one-step inference gives fast direct prediction, while a small number of integration steps improves accuracy without retraining.

4.3. Broader Applicability: Real-World Data and Inverse Modeling

To demonstrate the versatility of our Spatio-Temporal MeanFlow framework beyond standard synthetic forward prediction, we extend its evaluation to two challenging scenarios.

Real-World Data. We first evaluate our method on the HYCOM dataset (details in App. F.2), which provides data-assimilative 3D water temperature fields on a $32 \times 32 \times 8$ grid. A key challenge here is the absence of an explicit PDE residual due to realistic noise and assimilation artifacts. Therefore, we leverage the Spatio-Temporal MeanFlow formulation as a weak physical regularization to learn the temporally averaged flow. This mitigates the overfitting risks of unconstrained direct regression and encourages temporally

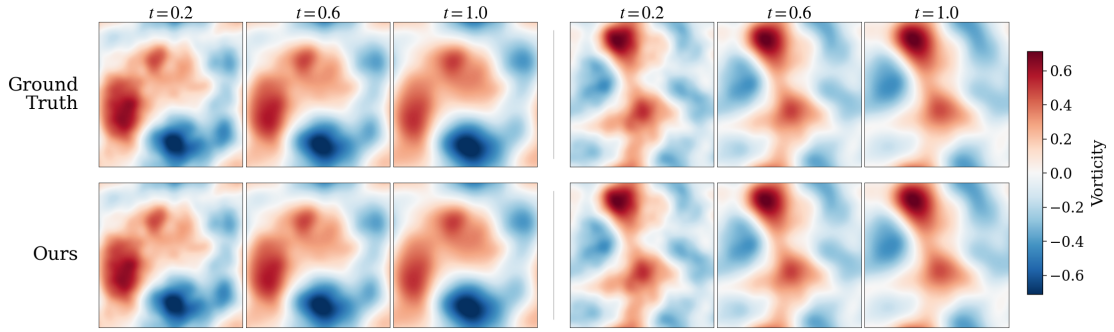

Figure 4. Qualitative visualization on the NS dataset. We compare ground-truth vorticity fields with predictions from our method at three time steps for two test samples.

coherent predictions. As shown in Tab. 1 ("Real-world"), this strategy ensures effective and stable predictions under realistic distributional shifts. **Inverse Problem on Darcy Flow.** Furthermore, we evaluate whether our framework can be applied to inverse problems on Darcy flow, where the objective is to recover the unknown permeability field $\boldsymbol{a}(x)$ from observed pressure $\boldsymbol{u}^{\text{obs}}(x)$. Via test-time optimization, we freeze the pretrained forward model and optimize the log-permeability $\log \boldsymbol{a}$ as a learnable parameter using gradient descent. As shown in Tab. 1, our approach achieves accurate coefficient reconstruction, highlighting the broader potential of extending our framework to inverse modeling.

5. Related Work

Physics-Informed Neural Networks (PINNs). PINNs (Raissi et al., 2019) represent a foundational class of PDE solvers that parameterize solutions by embedding physical laws as soft constraints. Despite their widespread success, PINNs suffer from a stiff and pathological optimization landscape due to competing physical loss terms. While numerous strategies have been proposed to stabilize training (e.g., uncertainty-based weighting (Huang et al., 2021; Kendall et al., 2018)), we argue that these methods rely solely on the PDE formulation itself, thereby overlooking the inherently continuous integral nature of the solving process. A more detailed discussion is provided in App. B.1.

Neural Operators. Neural operators like FNO (Li et al., 2020a) and DeepONet (Lu et al., 2019) enable rapid PDE inference. To enhance physical consistency, methods like PINO (Li et al., 2024) explicitly incorporate residual constraints. For time-dependent systems, the Continuous Flow Operator (CFO) (Hou et al., 2026) mitigates autoregressive error accumulation by repurposing flow matching to learn continuous-time dynamics. While CFO shares a similar continuous integration modeling philosophy with our approach, our framework additionally introduces a physics-inspired spatio-temporal MeanFlow loss, significantly enhancing its generalization capabilities. Other architectural variants targeting irregular geometries (Li et al., 2023; 2020b) and multi-scale structures (Tripura & Chakraborty, 2022; Rahman et al., 2022) are further discussed in App. B.2.

Generative Modeling for PDEs. Generative approaches simulate PDEs by treating solution fields as samples from learned distributions. While early diffusion models (Huang et al., 2024; Jacobsen et al., 2025) require numerous inference steps, recent works have improved sampling speed via rectified flows (Armegioiu et al., 2025), physical consistency via flow matching (Baldan et al., 2025), and distillation (Zhang & Zou, 2025). Nevertheless, these methods operate on a noise-to-solution generation paradigm. Rather than learning full data distributions, we simply adopt the continuous-time integration mechanism of generative models to formulate a continuous integrator over arbitrary time intervals. Further details are provided in App. B.3.

6. Conclusion

In this paper, we propose a physics-informed Neural PDE Solver via Spatio-Temporal MeanFlow. Functioning as an arbitrary-interval continuous integrator, it enables the computational efficiency of flexible integration steps. Furthermore, by enforcing physical consistency via a novel spatio-temporal loss, our method achieves enhanced generalization capabilities. Extensive evaluations demonstrate its superior accuracy, efficiency, and generalization.

Limitations and Future Work. Although our underlying mathematical formulation natively supports continuous spatial evolution, our current empirical implementation focuses on regular grids. Activating this continuous spatial modeling to tackle irregular domains represents the future step to further maximize the framework's generalization capabilities. Furthermore, extending our spatio-temporal loss to real-world datasets lacking analytical PDEs addresses a ubiquitous bottleneck in physics-informed learning. Finally, evolving our inverse modeling proof-of-concept into a unified architecture that seamlessly co-learns coefficient fields and solutions will further unleash the framework's potential.

Impact Statement

Our Spatio-Temporal MeanFlow framework accelerates complex physical simulations by transforming traditional multi-step numerical integration into efficient, finite-interval predictions. This enhanced computational efficiency has the potential to significantly reduce the energy consumption and carbon footprint associated with large-scale scientific modeling, such as fluid dynamics, meteorology, and climate forecasting. Furthermore, its demonstrated versatility in handling real-world datasets and inverse problems opens up promising downstream applications in environmental monitoring and engineering design.

Despite these positive implications, the deployment of neural PDE solvers in safety-critical domains (e.g., aerospace design or nuclear engineering) requires careful consideration. Due to the inherent black-box nature of deep learning, neural solvers may fail unpredictably on extreme anomalies, even with improved out-of-distribution generalization. Therefore, we strongly recommend utilizing our framework to accelerate iterative design and optimization phases, while continuing to rely on mathematically guaranteed numerical methods for final safety certifications.

References

Armegioiu, V., Ramic, Y., and Mishra, S. Rectified flows for fast multiscale fluid flow modeling. *arXiv preprint arXiv:2506.03111*, 2025.

Baldan, G., Liu, Q., Guardone, A., and Thuerey, N. Flow matching meets PDEs: A unified framework for physics-constrained generation. *arXiv preprint arXiv:2506.08604*, 2025.

Baldan, G., Liu, Q., Guardone, A., and Thuerey, N. Physics vs distributions: Pareto optimal flow matching with physics constraints. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=tAf1KI3d4X.

Bandrauk, A. D., Delfour, M. C., and Le Bris, C. *High-dimensional partial differential equations in science and engineering*, volume 41. American Mathematical Soc., 2007.

Cao, S. Choose a transformer: Fourier or galerkin. *Advances in neural information processing systems*, 34: 24924–24940, 2021.

Croitoru, F.-A., Hondru, V., Ionescu, R. T., and Shah, M. Diffusion models in vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45 (9):10850–10869, 2023.

Cuomo, S., Di Cola, V. S., Giampaolo, F., Rozza, G., Raissi, M., and Piccialli, F. Scientific machine learning through physics–informed neural networks: Where we are and what's next. *Journal of Scientific Computing*, 92(3):88, 2022.

Geng, Z., Deng, M., Bai, X., Kolter, J. Z., and He, K. Mean flows for one-step generative modeling. *arXiv preprint arXiv:2505.13447*, 2025.

Glimm, J. Nonlinear and stochastic phenomena: The grand challenge for partial differential equations. *SIAM review*, 33(4):626–643, 1991.

Hou, X., Huang, X., and Perdikaris, P. CFO: Learning continuous-time PDE dynamics via flow-matched neural operators. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=IQhaeSzyup.

Huang, J., Yang, G., Wang, Z., and Park, J. J. Diffusion-PDE: Generative PDE-solving under partial observation. *Advances in Neural Information Processing Systems*, 37: 130291–130323, 2024.

Huang, X., Liu, H., Shi, B., Wang, Z., Yang, K., Li, Y., Weng, B., Wang, M., Chu, H., Zhou, J., et al. Solving partial differential equations with point source based on physics-informed neural networks. *arXiv preprint arXiv:2111.01394*, 2021.

Jacobsen, C., Zhuang, Y., and Duraisamy, K. Cocogen: Physically consistent and conditioned score-based generative models for forward and inverse problems. *SIAM Journal on Scientific Computing*, 47(2):C399–C425, 2025.

Kendall, A., Gal, Y., and Cipolla, R. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7482–7491, 2018.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Krishnapriyan, A., Gholami, A., Zhe, S., Kirby, R., and Mahoney, M. W. Characterizing possible failure modes in physics-informed neural networks. *Advances in Neural Information Processing Systems*, 34:26548–26560, 2021.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020a.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Stuart, A., Bhattacharya, K., and Anandkumar, A. Multipole graph neural operator for parametric partial differential

equations. *Advances in Neural Information Processing Systems*, 33:6755–6766, 2020b.

Li, Z., Huang, D. Z., Liu, B., and Anandkumar, A. Fourier neural operator with learned deformations for PDEs on general geometries. *Journal of Machine Learning Research*, 24(388):1–26, 2023.

Li, Z., Zheng, H., Kovachki, N., Jin, D., Chen, H., Liu, B., Azizzadenesheli, K., and Anandkumar, A. Physics-informed neural operator for learning partial differential equations. *ACM/IMS Journal of Data Science*, 1(3):1–27, 2024.

Lipman, Y., Chen, R. T., Ben-Hamu, H., Nickel, M., and Le, M. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Liu, X., Gong, C., and Liu, Q. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

Lu, L., Jin, P., and Karniadakis, G. E. Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv preprint arXiv:1910.03193*, 2019.

Luo, C. Understanding diffusion models: A unified perspective. *arXiv preprint arXiv:2208.11970*, 2022.

Madhavi, M. Advanced numerical methods for solving nonlinear partial differential equations in fluid mechanics: Applications in aerospace engineering. *International Journal of Applied Mathematics*, 38(3s):140–152, 2025.

Niu, P., Guo, J., Chen, Y., Zhou, Y., Feng, M., and Shi, Y. Improved physics-informed neural network in mitigating gradient-related failures. *Neurocomputing*, 638:130167, 2025.

Rahman, M. A., Ross, Z. E., and Azizzadenesheli, K. U-NO: U-shaped neural operators. *arXiv preprint arXiv:2204.11127*, 2022.

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

Song, Y., Dhariwal, P., Chen, M., and Sutskever, I. Consistency models. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, volume 202 of *Proceedings of Machine Learning Research*. PMLR, July 2023. URL https://proceedings.mlr.press/v202/song23i.html. Poster. Submission Number 2703.

Tripura, T. and Chakraborty, S. Wavelet neural operator: A neural operator for parametric partial differential equations. *arXiv preprint arXiv:2205.02191*, 2022.

Wang, S., Teng, Y., and Perdikaris, P. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43(5):A3055–A3081, 2021.

Wang, S., Yu, X., and Perdikaris, P. When and why PINNs fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449:110768, 2022.

Wang, S., Seidman, J. H., Sankaran, S., Wang, H., Pappas, G. J., and Perdikaris, P. Cvit: Continuous vision transformer for operator learning. *arXiv preprint arXiv:2405.13998*, 2024.

Yang, L., Zhang, Z., Song, Y., Hong, S., Xu, R., Zhao, Y., Zhang, W., Cui, B., and Yang, M.-H. Diffusion models: A comprehensive survey of methods and applications. *ACM computing surveys*, 56(4):1–39, 2023.

Zhang, Y. and Zou, D. Physics-informed distillation of diffusion models for pde-constrained generation. *arXiv preprint arXiv:2505.22391*, 2025.

Appendix Overview

The following lists the structure of the appendix, with links to the respective sections.

A Detailed Derivation of the MeanFlow Identity

B Extended Related Work

- B.1 Physics-Informed Neural Networks (PINNs): Optimization Challenges in PINN
- B.2 Neural Operators: Advancements in Neural Operator Architectures
- B.3 Generative Modeling for PDE Simulation

C Generalized Spatio-Temporal MeanFlow Framework

D Theoretical Results

- D.1 The Construction of Spatio-Temporal MeanFlow Constraints
- D.2 Error Estimate
- D.3 MeanFlow Error Bounded by Decoupled Losses
- D.4 Hessian Condition Number Analysis

E Examples of Spatio-Temporal MeanFlow Losse

F Experiments

- F.1 Details of Synthetic PDE Benchmark
- F.2 Details of the Real-world HYCOM Datase
- F.3 Implementation Details of Sec. 4.1
- F.4 Implementation Details for HYCO
- F.5 Implementation Details for the Inverse Darcy Experiment
- F.6 Compared Methods
- F.7 Additional Experimental Results of Sec. 4.1

A. Detailed Derivation of the MeanFlow Identity

Conventional Flow Matching parameterizes the instantaneous velocity field $\boldsymbol{v}$, which inherently requires multi-step numerical integration during inference. MeanFlow (Geng et al., 2025) seeks to directly model the average velocity $\boldsymbol{m}$ over a finite time interval, thereby enabling one-step prediction.

Specifically, let $\boldsymbol{u}_t$ denote the state at time t along the true ODE trajectory governed by $\frac{\mathrm{d}\boldsymbol{u}_t}{\mathrm{d}t} = \boldsymbol{v}(\boldsymbol{u}_t, t)$. The average velocity field $\boldsymbol{m}(\boldsymbol{u}_t, \tau, t)$ is defined as the displacement divided by the time interval $(t - \tau)$:

$$\boldsymbol{m}(\boldsymbol{u}_t, \tau, t) \triangleq \frac{1}{t-\tau} \int_{\tau}^{t} \boldsymbol{v}(\boldsymbol{u}_s, s)\, \mathrm{d}s.$$

Directly utilizing this definition as a training objective necessitates continuous numerical integration during training. To construct an integral-free training objective, we establish a differential relationship between $\boldsymbol{u}$ and $\boldsymbol{v}$. By rewriting the definition as $(t-\tau)\boldsymbol{m}(\boldsymbol{u}_t, \tau, t) = \int_{\tau}^{t} \boldsymbol{v}(\boldsymbol{u}_s, s)\, \mathrm{d}s$ and taking the total derivative with respect to time t on both sides, we can eliminate the integral.

For the left-hand side, applying the product rule yields:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left[(t-\tau)\boldsymbol{m}(\boldsymbol{u}_t, \tau, t)\right] = \boldsymbol{m}(\boldsymbol{u}_t, \tau, t) + (t-\tau)\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{m}(\boldsymbol{u}_t, \tau, t).$$

The total derivative $\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{m}(\boldsymbol{u}_t, \tau, t)$ accounts for both the explicit dependence of $\boldsymbol{m}$ on t and its implicit dependence on t through the evolving state $\boldsymbol{u}_t$. Using the chain rule, this expands as $\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{m} = \partial_t \boldsymbol{m} + (\partial_{\boldsymbol{u}} \boldsymbol{m})\frac{\mathrm{d}\boldsymbol{u}_t}{\mathrm{d}t} = \partial_t \boldsymbol{m} + (\partial_{\boldsymbol{u}} \boldsymbol{m})\boldsymbol{v}(\boldsymbol{u}_t, t)$, assuming the reference time τ is a fixed constant independent of t (hence $\frac{\mathrm{d}\tau}{\mathrm{d}t} = 0$).

For the right-hand side, applying the Fundamental Theorem of Calculus with respect to the upper limit of the integral yields the integrand evaluated at time t:

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{\tau}^{t} \boldsymbol{v}(\boldsymbol{u}_s, s)\, \mathrm{d}s = \boldsymbol{v}(\boldsymbol{u}_t, t).$$

Equating the derivatives of both sides and rearranging the terms leads to the MeanFlow Identity:

$$\boldsymbol{m}(\boldsymbol{u}_t, \tau, t) = \boldsymbol{v}(\boldsymbol{u}_t, t) - (t-\tau)\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{m}(\boldsymbol{u}_t, \tau, t).$$

This identity provides a necessary and sufficient condition (given the natural boundary condition $\lim_{\tau \to t} \boldsymbol{m}(\boldsymbol{u}_t, \tau, t) = \boldsymbol{v}(\boldsymbol{u}_t, t)$) to characterize the true average velocity using a purely differential, integral-free form.

B. Extended Related Work

B.1. Physics-Informed Neural Networks (PINNs): Optimization Challenges in PINNs

Physics-Informed Neural Networks (PINNs) (Raissi et al., 2019) represent a foundational class of deep learning-based PDE solvers. They parameterize the solution manifold by embedding the governing physical laws into the training objective as soft constraints. Despite their widespread success across diverse physical domains (ranging from fluid dynamics to power systems), PINNs suffer from severe scalability bottlenecks due to their inherent reliance on instance-specific, iterative optimization. Furthermore, a prominent challenge in training PINNs is their notoriously pathological optimization landscape, caused by severe stiffness among competing boundary, initial, and residual loss terms.

To address this problem, subsequent works have developed sophisticated adaptive schemes. Beyond standard uncertainty weighting (Huang et al., 2021) and gradient pathology mitigation (Kendall et al., 2018), Wang et al. (Wang et al., 2021) proposed an adaptive learning rate annealing strategy along with an improved neural architecture (IA-PINN) to balance fitting and regularization. Concurrently, methods like IAW-PINN advanced uncertainty weighting by controlling the upper bounds of weights, offering more flexible balancing among distinct loss terms. Recognizing that singular approaches are often limited by network structure, recent efforts have integrated these strengths (such as I-PINN (Niu et al., 2025), which unifies both IA-PINN's architecture and IAW-PINN's weighting strategies).

B.2. Neural Operators: Advancements in Neural Operator Architectures

Neural operators learn a direct mapping between infinite-dimensional function spaces. Architectures like Fourier Neural Operators (FNO) (Li et al., 2020a) and DeepONet (Lu et al., 2019) enable rapid, zero-shot inference for new initial or boundary conditions. However, purely data-driven operators often struggle with out-of-distribution generalization and may violate physical laws. To address this, Physics-Informed Neural Operators (PINO) (Li et al., 2024) explicitly incorporate PDE residuals into the operator training objective, blending fast inference with physical constraints. However, when applied to time-dependent systems, these models typically rely on autoregressive (AR) schemes, which are prone to error accumulation over long horizons and restricted by uniform temporal discretization.

To address these limitations, the Continuous Flow Operator (CFO) (Hou et al., 2026) was introduced as a novel class of flow-based operators. CFO shares our core conceptual motivation: rather than treating flow matching as a simple generative sampler, it repurposes the framework to learn the underlying differential dynamics (the physical right-hand side of the PDE) directly. By fitting temporal splines to trajectory data, CFO constructs continuous velocity fields that grant the model inherent time-resolution invariance. However, despite this shared perspective, its formulation is purely temporal; it captures the continuous evolution over time but lacks explicit constraints to ensure physical consistency across the spatial domain, which is critical for PDE generalization. Our proposed temporal-spatial MeanFlow framework strategically overcomes this. By expanding the integral constraints to the spatio-temporal domain, we explicitly enforce physical consistency across spatial resolutions.

Furthermore, a wide array of other architectural extensions have been proposed to overcome the geometric constraints of standard neural operators. For instance, while original FNOs rely heavily on uniform grids, methods such as Geo-FNO (Li et al., 2023) and multipole graph neural operators (Li et al., 2020b) have been developed to accurately map and compute solutions across highly irregular physical geometries. Similarly, the Galerkin Transformer (Cao, 2021) and CViT (Wang et al., 2024) introduce robust mechanisms for handling non-uniform or irregularly sampled observational data.

To capture complex, overlapping physical phenomena, models like the Wavelet Neural Operator (WNO) (Tripura & Chakraborty, 2022) integrate multi-scale structural processing. Finally, architectures such as U-NO (Rahman et al., 2022) address the depth limitations of traditional operators, employing U-Net-like structures to enable deeper, more expressive feature extraction without sacrificing computational efficiency. While these methods successfully expand the expressivity and geometric flexibility of operator learning, our approach uniquely pivots toward using generative flow matching as a direct, physical equation solver to achieve rapid, single-step integration.

B.3. Generative Modeling for PDE Simulation

Generative modeling offers a novel paradigm for PDE simulation by treating solution fields as samples from learned distributions. Early advancements in this domain heavily relied on diffusion models. For example, DiffusionPDE (Huang et al., 2024) applies EDM-style diffusion combined with physics guidance during the sampling phase, while CoCoGen (Jacobsen et al., 2025) extends score-based models to ensure physically consistent generation. Although these conditional diffusion models achieve high fidelity, they typically require hundreds of stochastic steps at inference, severely limiting their practical utility for fast simulation.

To address this severe latency, researchers have increasingly explored flow matching and rectified flows. Armegioiu et al. (Armegioiu et al., 2025) introduced a rectified flow surrogate that learns a time-dependent conditional velocity field, effectively transporting input-to-output laws along nearly straight trajectories. By leveraging a curvature-aware sampler, this approach drastically accelerates inference, reducing the required ODE evaluations to as few as eight steps while matching the Wasserstein statistics of standard diffusion models. In parallel, to improve physical fidelity within these generative frameworks, methods like PBFM (Baldan et al., 2025) have been developed to explicitly integrate physical constraints directly into flow matching training through conflict-free gradient updates. Furthermore, (Zhang & Zou, 2025) proposed a post-hoc distillation method to enable more accurate and fast generation for PDE problems, achieving remarkable acceleration and improvements over many PDE-related problems.

Despite these significant advancements in inference speed and physical consistency, the core objective of these models remains unchanged: they are designed to map initial noise distributions to target solution manifolds. They fundamentally operate on a noise-to-sample generation paradigm. Our approach diverges completely from this statistical sampling formulation. Rather than generating solutions from noise, we adapt the continuous-time modeling capabilities of flow matching purely from the perspective of direct equation solving. By replacing the unconstrained generative velocity field

with the physical right-hand side of the PDE, our framework functions strictly as an efficient, deterministic neural integrator.

C. Generalized Spatio-Temporal MeanFlow Framework

In the main text, we presented a simplified algorithm tailored for regular spatial grids, which fixes the spatial coordinates and focuses on continuous-time integration. Here, we present the fully generalized Spatio-Temporal MeanFlow algorithms (Alg. 3 and Alg. 4). In this generalized formulation, both the temporal and spatial coordinates are treated as continuous variables, allowing the integration path to move simultaneously across time from τ to t and across space from an anchor point $\boldsymbol{\xi}$ to a target point $\boldsymbol{x}$.

Algorithm 3 Generalized Training

Require: PDE params $\boldsymbol{a}$, temporal indicator γ, network $\boldsymbol{m}_\theta$, max iter M, learning rate η
Ensure: Trained network parameters θ^*
for $i = 1$ to M **do**
  if $\gamma \neq 0$ **then** ▷ Time-variant PDEs
    Sample anchor time τ and anchor grid $\boldsymbol{\xi}$;
    Compute anchor state $\boldsymbol{u}_{\text{anchor}} = \boldsymbol{u}(\boldsymbol{\xi}, \tau)$;
    Sample target time $t > \tau$ and target grid $\boldsymbol{x}$;
    Predict MeanFlow:
    $\boldsymbol{m} \leftarrow \boldsymbol{m}_\theta([\boldsymbol{u}_{\text{anchor}}, \boldsymbol{\xi}, \tau, \boldsymbol{x}, t, \boldsymbol{a}])$;
    Compute total loss:
    $\mathcal{L}_{\text{Total}} \leftarrow \mathcal{L}_{\text{Data}} + \lambda_t \mathcal{L}_{\text{T-MF}} + \lambda_s \mathcal{L}_{\text{S-MF}}$;
  else ▷ Steady-state PDEs ($\gamma = 0$)
    Create initial state $\boldsymbol{u}_0$ by the harmonic extension of $\boldsymbol{u}_0(\boldsymbol{\xi})$ for $\boldsymbol{\xi} \in \partial D$;
    Sample target grid $\boldsymbol{x}$ for domain D;
    Compute $\mathcal{L}_{\text{Data}}$ and $\mathcal{L}_{\text{S-MF}}$;
    $\mathcal{L}_{\text{Total}} \leftarrow \mathcal{L}_{\text{Data}} + \lambda_s \mathcal{L}_{\text{S-MF}}$;
  end if
  Update: $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}_{\text{Total}}$;
end for
return $\theta^* = \theta$;

Algorithm 4 Generalized Multi-step Inference

Require: Pretrained net $\boldsymbol{m}_{\theta^*}$, start grid $\boldsymbol{x}_0$, start time t_0, start state $\boldsymbol{u}_0$, target grid $\boldsymbol{x}_N$, step Δt, steps N
Ensure: Spatio-temporal trajectory $\{\boldsymbol{u}_k\}_{k=0}^N$
Initialize trajectory list with $\boldsymbol{u}_0$;
for $k = 0$ to $N - 1$ **do**
  Anchor coords: $(\boldsymbol{\xi}, \tau) \leftarrow (\boldsymbol{x}_k, t_k)$;
  Anchor state: $\boldsymbol{u}_{\text{anchor}} \leftarrow \boldsymbol{u}_k$;
  Target grid: $\boldsymbol{x}_{k+1} \leftarrow \boldsymbol{x}_k + (\boldsymbol{x}_N - \boldsymbol{x}_0)/N$;
  Target time: $t_{k+1} \leftarrow t_k + \Delta t$;
  Spatio-temporal arc length:
  $l \leftarrow \sqrt{\frac{\|\boldsymbol{x}_N - \boldsymbol{x}_0\|_2^2}{N^2} + \Delta t^2}$;
  Predict MeanFlow:
  $\boldsymbol{m} \leftarrow \boldsymbol{m}_\theta([\boldsymbol{u}_{\text{anchor}}, \boldsymbol{\xi}, \tau, \boldsymbol{x}_{k+1}, t_{k+1}, \boldsymbol{a}])$;
  State increment: $\Delta \boldsymbol{u} \leftarrow l \cdot \boldsymbol{m}$;
  Update field: $\boldsymbol{u}_{k+1} \leftarrow \boldsymbol{u}_k + \Delta \boldsymbol{u}$;
  Append $\boldsymbol{u}_{k+1}$ to trajectory list;
end for
return $\{\boldsymbol{u}_0, \boldsymbol{u}_1, \ldots, \boldsymbol{u}_N\}$;

D. Theoretical Results

D.1. The Construction of Spatio-Temporal MeanFlow Constraints

In this section, Prop. D.1 conclude and show how the MeanFlow constraints being constructed.

Proposition D.1 (MeanFlow Constraints.)**.** *For MeanFlow function $\boldsymbol{m}$ satisfying*

$$l(\boldsymbol{\xi}, \tau, \boldsymbol{x}, t) \cdot \boldsymbol{m}[\boldsymbol{u}(\boldsymbol{\xi}, \tau), \boldsymbol{\xi}, \tau, \boldsymbol{x}, t, \boldsymbol{a}] = \boldsymbol{u}(\boldsymbol{x}, t) - \boldsymbol{u}(\boldsymbol{\xi}, t), \tag{15}$$

we have

$$\begin{cases} l_t \cdot \boldsymbol{m} = l\boldsymbol{K}\boldsymbol{f} + \gamma l^2 \cdot \dfrac{\partial \boldsymbol{m}}{\partial \tau}, \\ l \cdot \boldsymbol{K} \cdot \nabla^\top \boldsymbol{u}(\boldsymbol{\xi}, \tau) = \boldsymbol{m}(\boldsymbol{x} - \boldsymbol{\xi})^\top - l^2 \cdot \dfrac{\partial \boldsymbol{m}}{\partial \boldsymbol{\xi}}, \\ l \cdot \boldsymbol{K} \cdot \Delta \boldsymbol{u}(\boldsymbol{\xi}, \tau) = -(\boldsymbol{\alpha} + 2\boldsymbol{K}^{-1}\boldsymbol{\beta}), \end{cases}$$

where $\boldsymbol{\alpha} = n\boldsymbol{m} - (l_s^2 \,/\, l^2) \cdot \boldsymbol{m} + 2l^2 \Delta \boldsymbol{m}(\boldsymbol{\xi}, \tau)$ and $\boldsymbol{\beta} = -\frac{\partial \boldsymbol{m}}{\partial \boldsymbol{\xi}}(\boldsymbol{x} - \boldsymbol{\xi}) - (l_s^2 \,/\, l) \cdot \frac{\partial \boldsymbol{m}}{\partial \boldsymbol{u}} \boldsymbol{m}$.

Proof. We enforce a perturbation on γ if $\gamma = 0$ and let $\gamma \to 0$ in the end phase. This technique allows us to regard parameter γ as a non-sigular value.

We compute the derivatives of $\boldsymbol{m}$ given that $\boldsymbol{\xi}, \boldsymbol{x}, t, \boldsymbol{a}$ are constants w.r.t. τ and $\tau, \boldsymbol{x}, t, \boldsymbol{a}$ are constants w.r.t. $\boldsymbol{\xi}$, leading to

$$\begin{cases} \dfrac{\mathrm{d}\boldsymbol{m}}{\mathrm{d}\tau} = \dfrac{\partial \boldsymbol{m}}{\partial \boldsymbol{u}} \cdot \dfrac{\partial \boldsymbol{u}}{\partial \tau} + \dfrac{\partial \boldsymbol{m}}{\partial \tau} = \dfrac{\partial \boldsymbol{m}}{\partial \boldsymbol{u}} \cdot \dfrac{\boldsymbol{f}(\boldsymbol{\xi}, \tau)}{\gamma} + \dfrac{\partial \boldsymbol{m}}{\partial \tau}, \\ \dfrac{\mathrm{d}\boldsymbol{m}}{\mathrm{d}\boldsymbol{\xi}} = \dfrac{\partial \boldsymbol{m}}{\partial \boldsymbol{u}} \cdot \dfrac{\partial \boldsymbol{u}}{\partial \boldsymbol{\xi}} + \dfrac{\partial \boldsymbol{m}}{\partial \boldsymbol{\xi}} = \dfrac{\partial \boldsymbol{m}}{\partial \boldsymbol{u}} \cdot \nabla^\top \boldsymbol{u}(\boldsymbol{\xi}, \tau) + \dfrac{\partial \boldsymbol{m}}{\partial \boldsymbol{\xi}}. \end{cases} \tag{16}$$

The derivatives w.r.t. τ of both sides of Eq. (15) are

$$\begin{cases} \dfrac{\mathrm{dLHS}}{\mathrm{d}\tau} = \dfrac{\mathrm{d}l}{\mathrm{d}\tau}\boldsymbol{m} + l\dfrac{\mathrm{d}\boldsymbol{m}}{\mathrm{d}\tau} = \dfrac{-(t-\tau)\boldsymbol{m}}{l} + l\left(\dfrac{\partial \boldsymbol{m}}{\partial \boldsymbol{u}} \cdot \dfrac{\boldsymbol{f}(\boldsymbol{\xi}, \tau)}{\gamma} + \dfrac{\partial \boldsymbol{m}}{\partial \tau}\right), \\ \dfrac{\mathrm{dRHS}}{\mathrm{d}\tau} = -\dfrac{\partial \boldsymbol{u}(\boldsymbol{\xi}, \tau)}{\partial \tau} = -\dfrac{\boldsymbol{f}(\boldsymbol{\xi}, \tau)}{\gamma}, \end{cases} \tag{17}$$

where $\boldsymbol{f}(\boldsymbol{\xi}, \tau) \triangleq \boldsymbol{f}[\boldsymbol{u}(\boldsymbol{\xi}, \tau), \nabla, \boldsymbol{\xi}, \tau, \boldsymbol{a}]$. Notably, the notations $\frac{\mathrm{d}\cdot}{\mathrm{d}\cdot}$ and $\frac{\partial\cdot}{\partial\cdot}$ both represent the Jacobian matrices, whereas $\nabla \boldsymbol{f}$ represent spatial gradients, which is the transpose of Jacobian matrix.

Consequently, we have the ground-truth approximation

$$\begin{aligned} \boldsymbol{m}_{\mathrm{gt}} &= \frac{l}{t-\tau}\left[l\left(\frac{\partial \boldsymbol{m}}{\partial \boldsymbol{u}} \cdot \frac{\boldsymbol{f}(\boldsymbol{\xi}, \tau)}{\gamma} + \frac{\partial \boldsymbol{m}}{\partial \tau}\right) + \frac{\boldsymbol{f}(\boldsymbol{\xi}, \tau)}{\gamma}\right], \\ &= \frac{l}{t-\tau}\left[\boldsymbol{K} \cdot \frac{\boldsymbol{f}(\boldsymbol{\xi}, \tau)}{\gamma} + l\frac{\partial \boldsymbol{m}}{\partial \tau}\right], \end{aligned} \tag{18}$$

where $l \triangleq \sqrt{(t-\tau)^2 + \|\boldsymbol{x}-\boldsymbol{\xi}\|_2^2}$ and $\boldsymbol{K} = \boldsymbol{I} + l\frac{\partial \boldsymbol{m}}{\partial \boldsymbol{u}}$. **This coheres to the final result in conventional *temporal MeanFlow*. The *spatial* part is derived as follows.**

The derivatives of Eq. (15) w.r.t. $\boldsymbol{\xi}$, on the other hand, result in

$$\begin{cases} \dfrac{\mathrm{dLHS}}{\mathrm{d}\boldsymbol{\xi}} = \dfrac{\boldsymbol{m}(\boldsymbol{\xi}-\boldsymbol{x})^\top}{l} + l\left(\dfrac{\partial \boldsymbol{m}}{\partial \boldsymbol{u}} \cdot \nabla^\top \boldsymbol{u}(\boldsymbol{\xi}, \tau) + \dfrac{\partial \boldsymbol{m}}{\partial \boldsymbol{\xi}}\right), \\ \dfrac{\mathrm{dRHS}}{\mathrm{d}\boldsymbol{\xi}} = -\dfrac{\partial \boldsymbol{u}(\boldsymbol{\xi}, \tau)}{\partial \boldsymbol{\xi}} = -\nabla^\top \boldsymbol{u}(\boldsymbol{\xi}, \tau), \end{cases} \tag{19}$$

and thus

$$\boldsymbol{K} \cdot \nabla^\top \boldsymbol{u}(\boldsymbol{\xi}, \tau) = \frac{1}{l}\boldsymbol{m}(\boldsymbol{x}-\boldsymbol{\xi})^\top - l\frac{\partial \boldsymbol{m}}{\partial \boldsymbol{\xi}}. \tag{20}$$

Note that this result may substitute the $\nabla \boldsymbol{u}$'s of $\boldsymbol{K}\boldsymbol{f}$ in (18).

There are more PDEs containing two or more order of derivatives, e.g. the Navier-Stokes equations with $\Delta \boldsymbol{u}$ within considerations.

For PDE with higher order in space, say order-2, we have

$$\begin{cases} \dfrac{\mathrm{d}^2\mathrm{LHS}}{\mathrm{d}\boldsymbol{\xi}^2} = \dfrac{\boldsymbol{m}\boldsymbol{1}^\top}{l} - \dfrac{1}{l^3}\boldsymbol{m}(\boldsymbol{\xi}-\boldsymbol{x})^\top \mathrm{diag}\,(\boldsymbol{\xi}-\boldsymbol{x}) + l\left(\dfrac{\partial \boldsymbol{m}}{\partial \boldsymbol{u}} \cdot \dfrac{\partial^2 \boldsymbol{u}}{\partial \boldsymbol{\xi}^2} + 2\dfrac{\partial^2 \boldsymbol{m}}{\partial \boldsymbol{\xi}^2}\right) \\ \qquad\qquad + \dfrac{2}{l}\left(\dfrac{\partial \boldsymbol{m}}{\partial \boldsymbol{u}} \cdot \nabla^\top \boldsymbol{u}(\boldsymbol{\xi}, \tau) + \dfrac{\partial \boldsymbol{m}}{\partial \boldsymbol{\xi}}\right)\mathrm{diag}\,(\boldsymbol{\xi}-\boldsymbol{x}), \\ \dfrac{\mathrm{d}^2\mathrm{RHS}}{\mathrm{d}\boldsymbol{\xi}^2} = -\dfrac{\partial^2 \boldsymbol{z}(\boldsymbol{\xi}, \tau)}{\partial \boldsymbol{\xi}^2}, \end{cases} \tag{21}$$

where the matrices $\left[\frac{\mathrm{d}^2\boldsymbol{z}}{\mathrm{d}\boldsymbol{x}^2}\right]_{ij} \triangleq \frac{\mathrm{d}^2\boldsymbol{z}_i}{\mathrm{d}\boldsymbol{x}_j^2}$ and $\left[\frac{\partial^2\boldsymbol{z}}{\partial\boldsymbol{x}^2}\right]_{ij} \triangleq \frac{\partial^2\boldsymbol{z}_i}{\partial\boldsymbol{x}_j^2}$, then,

$$
\begin{aligned}
&-\boldsymbol{K}\Delta\boldsymbol{u}(\boldsymbol{\xi},\tau),\\
=&-\boldsymbol{K}\frac{\partial^2\boldsymbol{u}(\boldsymbol{\xi},\tau)}{\partial^2\boldsymbol{\xi}^2}\mathbf{1},\\
=&-\left(\boldsymbol{I}+l\frac{\partial\boldsymbol{m}}{\partial\boldsymbol{u}}\right)\cdot\frac{\partial^2\boldsymbol{u}(\boldsymbol{\xi},\tau)}{\partial^2\boldsymbol{\xi}^2}\mathbf{1},\\
=&\frac{n}{l}\boldsymbol{m}-\frac{\|\boldsymbol{\xi}-\boldsymbol{x}\|_2^2}{l^3}\boldsymbol{m}+2l\Delta\boldsymbol{m}(\boldsymbol{\xi},\tau)+\frac{2}{l}\frac{\partial\boldsymbol{m}}{\partial\boldsymbol{\xi}}(\boldsymbol{\xi}-\boldsymbol{x})\\
&-\frac{2}{l}\frac{\partial\boldsymbol{m}}{\partial\boldsymbol{u}}\boldsymbol{K}^{-1}\left(\frac{1}{l}\boldsymbol{m}(\boldsymbol{\xi}-\boldsymbol{x})^\top+l\frac{\partial\boldsymbol{m}}{\partial\boldsymbol{\xi}}\right)(\boldsymbol{\xi}-\boldsymbol{x}),\\
=&\frac{\boldsymbol{\alpha}}{l}-\frac{2}{l^2}\frac{\partial\boldsymbol{m}}{\partial\boldsymbol{u}}\boldsymbol{K}^{-1}\boldsymbol{m}\|\boldsymbol{\xi}-\boldsymbol{x}\|_2^2+2\left[\frac{\boldsymbol{K}}{l}-\frac{\partial\boldsymbol{m}}{\partial\boldsymbol{u}}\right]\boldsymbol{K}^{-1}\frac{\partial\boldsymbol{m}}{\partial\boldsymbol{\xi}}(\boldsymbol{\xi}-\boldsymbol{x}),
\end{aligned}
\tag{22}
$$

hence

$$
\boldsymbol{K}\Delta\boldsymbol{u}(\boldsymbol{\xi},\tau)=-\Big(\frac{\boldsymbol{\alpha}}{l}+\frac{2}{l}\boldsymbol{K}^{-1}\boldsymbol{\beta}\Big),
\tag{23}
$$

where $\boldsymbol{\alpha}\triangleq n\boldsymbol{m}-\frac{\|\boldsymbol{x}-\boldsymbol{\xi}\|_2^2}{l^2}\boldsymbol{m}+2l^2\Delta\boldsymbol{m}(\boldsymbol{\xi},\tau)$ and $\boldsymbol{\beta}\triangleq-\frac{\partial\boldsymbol{m}}{\partial\boldsymbol{\xi}}(\boldsymbol{x}-\boldsymbol{\xi})-\frac{\|\boldsymbol{x}-\boldsymbol{\xi}\|_2^2}{l}\frac{\partial\boldsymbol{m}}{\partial\boldsymbol{u}}\boldsymbol{m}$, in which n represents the number of spatial dimensions. Consequently, $-l\boldsymbol{K}^2\Delta\boldsymbol{u}(\boldsymbol{\xi},\tau)=\boldsymbol{K}\boldsymbol{\alpha}+2\boldsymbol{\beta}$. □

D.2. Error Estimate

Theorem D.2 (A-posteriori Error Bound, manuscript)**.** *Assume the PDE operator $\boldsymbol{f}$ is uniformly Lipschitz continuous with respect to the state variable with constant L, and continuously differentiable ($\mathcal{C}^1$) such that its Jacobian $\nabla_{\boldsymbol{u}}\boldsymbol{f}$ is bounded locally. Further, assume the MeanFlow predictor $\boldsymbol{m}_\theta$ and the true physical state $\boldsymbol{u}_{True}$ are $\mathcal{C}^1$-continuous. If the total training loss bounds the squared path residuals such that $\mathcal{L}_{MeanFlow}\le\epsilon$, then the pointwise error between the reconstructed state $\boldsymbol{u}_\theta$ and the true solution $\boldsymbol{u}_{True}$ at any target point $(\boldsymbol{x},t)$ is bounded by:*

$$
\|\boldsymbol{u}_\theta(\boldsymbol{x},t)-\boldsymbol{u}_{True}(\boldsymbol{x},t)\|_2\le C\sqrt{\epsilon}+\mathcal{O}(l^2),
\tag{24}
$$

where C is a constant depending on the spatio-temporal path length l and the Lipschitz constant L.

Proof. Let the straight spatio-temporal path $\mathfrak{p}$ connecting the starting point $(\boldsymbol{\xi},\tau)$ and the target point $(\boldsymbol{x},t)$ be parameterized by a scalar $s\in[0,1]$. The predicted state along the path is linearly modeled as $\boldsymbol{u}_\theta(s)=\boldsymbol{u}(\boldsymbol{\xi},\tau)+s\cdot l\cdot\boldsymbol{m}_\theta$, implying its path derivative is precisely $\frac{\mathrm{d}\boldsymbol{u}_\theta}{\mathrm{d}s}=l\cdot\boldsymbol{m}_\theta$.

We define the pointwise state error as $\boldsymbol{v}(s)=\boldsymbol{u}_\theta(s)-\boldsymbol{u}_{\text{True}}(s)$. Assuming the starting state is exact (i.e., anchored by data constraints such that $\boldsymbol{v}(0)=\mathbf{0}$), the evolution of the error along the path is governed by the difference in their derivatives:

$$
\frac{\mathrm{d}\boldsymbol{v}}{\mathrm{d}s}=\frac{\mathrm{d}\boldsymbol{u}_\theta}{\mathrm{d}s}-\frac{\mathrm{d}\boldsymbol{u}_{\text{True}}}{\mathrm{d}s}=l\cdot\boldsymbol{m}_\theta-l\cdot\boldsymbol{f}(\boldsymbol{u}_{\text{True}}(s)).
\tag{25}
$$

To bridge the predicted MeanFlow and the true physical operator, we explicitly add and subtract the term $l\cdot\boldsymbol{f}(\boldsymbol{u}_\theta(s))$:

$$
\frac{\mathrm{d}\boldsymbol{v}}{\mathrm{d}s}=l\cdot\left[\boldsymbol{f}(\boldsymbol{u}_\theta(s))-\boldsymbol{f}(\boldsymbol{u}_{\text{True}}(s))\right]+l\cdot\left[\boldsymbol{m}_\theta-\boldsymbol{f}(\boldsymbol{u}_\theta(s))\right].
\tag{26}
$$

The first bracketed term is governed by the Lipschitz continuity of the PDE operator $\boldsymbol{f}$ with constant L, bounded by $L\|\boldsymbol{v}(s)\|_2$. The second bracketed term measures the deviation between the constant MeanFlow vector and the continuous local PDE evaluation. By Taylor expanding $\boldsymbol{f}(\boldsymbol{u}_\theta(s))$ around the starting point $s=0$, we relate it to the global MeanFlow residual $\boldsymbol{r}_{\text{MeanFlow}}$ evaluated by the network:

$$
l\cdot[\boldsymbol{m}_\theta-\boldsymbol{f}(\boldsymbol{u}_\theta(s))]=\boldsymbol{r}_{\text{MeanFlow}}+\boldsymbol{r}(s),
\tag{27}
$$

where $\boldsymbol{r}(s) \sim s \cdot l^2 \nabla_{\boldsymbol{u}} \boldsymbol{f} \cdot \boldsymbol{m}_\theta$ is the first-order Taylor remainder along the path. Because the state derivatives and the Jacobian $\nabla_{\boldsymbol{u}} \boldsymbol{f}$ are assumed bounded, the remainder satisfies $\|\boldsymbol{r}(s)\|_2 \leq M s l^2 = \mathcal{O}(l^2)$, with M being a local bounding constant.

Taking the L_2 norm on both sides and applying the triangle inequality yields a rigorous differential inequality:

$$\left\| \frac{\mathrm{d}\boldsymbol{v}}{\mathrm{d}s} \right\|_2 \leq l \cdot L \|\boldsymbol{v}(s)\|_2 + \|\boldsymbol{r}_{\text{MeanFlow}}\|_2 + \mathcal{O}(l^2). \tag{28}$$

Applying Grönwall's inequality to integrate this bound from $s = 0$ to $s = 1$ yields:

$$\begin{aligned} &\frac{\mathrm{d}}{\mathrm{d}s} \|\boldsymbol{v}(s)\|_2 \leq \left\| \frac{\mathrm{d}\boldsymbol{v}}{\mathrm{d}s} \right\|_2 \leq l \cdot L \|\boldsymbol{v}(s)\|_2 + \|\boldsymbol{r}_{\text{MeanFlow}}\|_2 + \mathcal{O}(l^2), \\ \Longrightarrow &\frac{\mathrm{d}}{\mathrm{d}s} \left[\|\boldsymbol{v}(s)\|_2 \, \mathrm{e}^{-lLs} \right] \leq \left[\|\boldsymbol{r}_{\text{MeanFlow}}\|_2 + \mathcal{O}(l^2) \right] \mathrm{e}^{-lLs}, \\ \Longrightarrow &\|\boldsymbol{v}(1)\|_2 \leq \int_0^1 \left[\|\boldsymbol{r}_{\text{MeanFlow}}\|_2 + \mathcal{O}(l^2) \right] \exp\left[lL(1-s) \right] \mathrm{d}s. \end{aligned} \tag{29}$$

The integral linearly separates into two components. For the first component involving the trained residual, we apply the Cauchy-Schwarz inequality to relate it to the squared loss $\mathcal{L}_{\text{MeanFlow}} \leq \epsilon$:

$$\begin{aligned} \int_0^1 \|\boldsymbol{r}_{\text{MeanFlow}}\|_2 \exp\left[lL(1-s) \right] \mathrm{d}s &\leq \sqrt{\mathcal{L}_{\text{MeanFlow}}} \cdot \left(\int_0^1 \exp\left[2lL(1-s) \right] \mathrm{d}s \right)^{\frac{1}{2}} \\ &\leq C\sqrt{\epsilon}, \end{aligned} \tag{30}$$

where $C = \sqrt{\frac{\exp(2lL)-1}{2lL}}$. For the second component, the integration of the $\mathcal{O}(l^2)$ term over $s \in [0, 1]$ simply scales the exponential factor, strictly preserving the $\mathcal{O}(l^2)$ magnitude order.

Superimposing these integrated bounds directly yields the final pointwise error bound:

$$\|\boldsymbol{u}_\theta(\boldsymbol{x}, t) - \boldsymbol{u}_{\text{True}}(\boldsymbol{x}, t)\|_2 \leq C\sqrt{\epsilon} + \mathcal{O}(l^2). \tag{31}$$

This establishes that the global error is bounded by the optimization loss and a strict structural truncation error, concluding the proof. □

D.3. MeanFlow Error Bounded by Decoupled Losses

Theorem D.3 (Thm. 3.2, manuscript)**.** *Assume the MeanFlow predictor $\boldsymbol{m}_\theta$ is $\mathcal{C}^1$-continuous. If the total training loss bounds the squared path residuals such that $\mathcal{L}_{MeanFlow} \leq \epsilon$, then the pointwise error between the reconstructed state $\boldsymbol{u}_\theta$ and the true solution $\boldsymbol{u}_{True}$ at any target point $(\boldsymbol{x}, t)$ is bounded by:*

$$\mathcal{L}_{MeanFlow} \leq c_1 \mathcal{L}_{Temp} + c_2 \mathcal{L}_{Spac} + \varepsilon, \tag{32}$$

where the coefficients are defined as $c_1 = 3l_t^2$ and $c_2 = 3\left(\frac{l-l_t}{l_s}\|\boldsymbol{g}\|_{op}\right)^2$. Here, ε represents the structural coupling error and $\varepsilon = \mathcal{O}(l_s^2)$.

Proof. We begin with the spatio-temporal residual vector $\boldsymbol{r}_{\text{MeanFlow}}$:

$$\boldsymbol{r}_{\text{MeanFlow}} = l_t \boldsymbol{m} - l\boldsymbol{K}\boldsymbol{f} - \gamma l^2 \frac{\partial \boldsymbol{m}}{\partial \tau}. \tag{S1}$$

From the definition of the temporal residual $\boldsymbol{r}_{\text{Temp}}$, we isolate the ground-truth term:

$$\boldsymbol{m} = \boldsymbol{r}_{\text{Temp}} + \boldsymbol{K}\boldsymbol{f} + l_t \frac{\partial \boldsymbol{m}}{\partial \tau}. \tag{33}$$

Substituting this into Eq. (S1) yields:

$$\begin{aligned} \boldsymbol{r}_{\text{MeanFlow}} =& l_t(\boldsymbol{r}_{\text{Temp}} + \boldsymbol{K}\boldsymbol{f} + l_t\frac{\partial \boldsymbol{m}}{\partial \tau}) - l\boldsymbol{K}\boldsymbol{f} - \gamma l^2\frac{\partial \boldsymbol{m}}{\partial \tau}, \\ =& l_t\boldsymbol{r}_{\text{Temp}} + (l_t - l)\boldsymbol{K}\boldsymbol{f} + (l_t^2 - \gamma l^2)\frac{\partial \boldsymbol{m}}{\partial \tau}. \end{aligned} \tag{S2}$$

Given the linear gradient PDE structure $\boldsymbol{f} = \nabla^\top \boldsymbol{u} \cdot \boldsymbol{g} + \boldsymbol{h}$, and the spatial residual $\boldsymbol{r}_{\text{Spac}}$, we substitute the spatial gradient $\boldsymbol{K}\nabla^\top \boldsymbol{u}$ into the expression for $\boldsymbol{K}\boldsymbol{f}$, i.e.,

$$\begin{aligned} \boldsymbol{r}_{\text{MeanFlow}} =& l_t\boldsymbol{r}_{\text{Temp}} + (l_t - l)\boldsymbol{K}(\nabla^\top \boldsymbol{u} \cdot \boldsymbol{g} + \boldsymbol{h}) + (l_t^2 - \gamma l^2)\frac{\partial \boldsymbol{m}}{\partial \tau}, \\ =& l_t\boldsymbol{r}_{\text{Temp}} + \frac{l_t - l}{l_s}(\boldsymbol{r}_{\text{Spac}} + \boldsymbol{m}(\boldsymbol{x} - \boldsymbol{\xi})^\top - l_s^2\frac{\partial \boldsymbol{m}}{\partial \boldsymbol{\xi}}) \cdot \boldsymbol{g} + (l_t - l)\boldsymbol{K}\boldsymbol{h} + (l_t^2 - \gamma l^2)\frac{\partial \boldsymbol{m}}{\partial \tau}, \end{aligned} \tag{S3}$$

Applying the generalized triangle inequality for squared L_2 norms (or Jensen's inequality for quadratic function, i.e., $\|\sum_{i=1}^3 \boldsymbol{a}_i\|_2^2 \le 3\sum_{i=1}^3 \|\boldsymbol{a}_i\|_2^2$) and the operator norm bound $\|\boldsymbol{v} \cdot \boldsymbol{g}\|_2 \le \|\boldsymbol{g}\|_{\text{op}}\|\boldsymbol{v}\|_2$, we obtain:

$$\|\boldsymbol{r}_{\text{MeanFlow}}\|_2^2 \le 3l_t^2\|\boldsymbol{r}_{\text{Temp}}\|_2^2 + 3\left(\frac{l - l_t}{l_s}\|\boldsymbol{g}\|_{\text{op}}\right)^2\|\boldsymbol{r}_{\text{Spac}}\|_2^2 + 3\|\boldsymbol{r}\|_2^2 \tag{34}$$

where

$$\boldsymbol{r} = \underbrace{\frac{l_t - l}{l_s}\left[\boldsymbol{m}(\boldsymbol{x} - \boldsymbol{\xi})^\top - l_s^2\frac{\partial \boldsymbol{m}}{\partial \boldsymbol{\xi}}\right] \cdot \boldsymbol{g}}_{\text{Boundary or Higher-Order Discretization Errors}} + \underbrace{(l_t - l)\boldsymbol{K}\boldsymbol{h}}_{\text{Source Term Error}} + \underbrace{(l_t^2 - \gamma l^2)\frac{\partial \boldsymbol{m}}{\partial \tau}}_{\text{Temporal Scaling Mismatch}}, \tag{35}$$

collects the remaining structural terms. Therefore, as $\|\boldsymbol{K}\|_{\text{op}} \le \rho$ and the state variables are sufficiently smooth such that their norms are bounded, we have, applying generalized triangle inequality for squared L_2 norms again,

$$\begin{aligned} \varepsilon = 3\|\boldsymbol{r}\|_2^2 \le& 9(l_t - l)^2 \cdot \left[\frac{1}{l_s^2} \cdot \|\boldsymbol{r}_{\boldsymbol{g}}\|_2^2 + \rho^2\|\boldsymbol{h}\|_2^2 + \left(\frac{l_t^2 - \gamma l^2}{l_t - l}\right)^2 \left\|\frac{\partial \boldsymbol{m}}{\partial \tau}\right\|_2^2\right], \\ =& \begin{cases} 9\,\|\boldsymbol{r}_{\boldsymbol{g}}\|_2^2 + 9l_s^2 \cdot \rho^2\|\boldsymbol{h}\|_2^2, & \text{if } \gamma = 0, \\ 9(l_t - l)^2 \cdot \left[\frac{1}{l_s^2} \cdot \|\boldsymbol{r}_{\boldsymbol{g}}\|_2^2 + \rho^2\|\boldsymbol{h}\|_2^2 + (l_t + l)^2\left\|\frac{\partial \boldsymbol{m}}{\partial \tau}\right\|_2^2\right], & \text{if } \gamma = 1, \end{cases} \end{aligned} \tag{36}$$

where $\boldsymbol{r}_{\boldsymbol{g}} \triangleq \left[\boldsymbol{m}(\boldsymbol{x} - \boldsymbol{\xi})^\top - l_s^2\frac{\partial \boldsymbol{m}}{\partial \boldsymbol{\xi}}\right] \cdot \boldsymbol{g}$. Here, the result for $\gamma = 0$ follows from the fact that $l_t = 0$ and $l = l_s$ for a steady-state system implied by $\gamma = 0$.

Note that the higher-order discretization error $\boldsymbol{r}_{\boldsymbol{g}}$ satisfies

$$\begin{aligned} \|\boldsymbol{r}_{\boldsymbol{g}}\|_2 \le& \|\boldsymbol{m}(\boldsymbol{x} - \boldsymbol{\xi})^\top\boldsymbol{g}\|_2 + \left\|l_s^2\frac{\partial \boldsymbol{m}}{\partial \boldsymbol{\xi}}\boldsymbol{g}\right\|_2 \le \|\boldsymbol{m}\|_2 \cdot \|\boldsymbol{x} - \boldsymbol{\xi}\|_2 \cdot \|\boldsymbol{g}\|_{\text{op}} + \mathcal{O}(l_s^2), \\ =& \|\boldsymbol{m}\|_2 \cdot l_s \cdot \|\boldsymbol{g}\|_{\text{op}} = \mathcal{O}(l_s). \end{aligned}$$

Consequently, since the bracketed term in the $\gamma = 1$ case is bounded by $\mathcal{O}(1)$ and the temporal scaling factor satisfies $l_t - l = \frac{l_t^2 - l^2}{l_t + l} = -\frac{l_s^2}{l_t + l} = \mathcal{O}(l_s^2)$, the asymptotic bound for the structural coupling error yields:

$$\varepsilon = \begin{cases} \mathcal{O}(l_s^2), & \text{if } \gamma = 0, \\ \mathcal{O}(l_s^4), & \text{if } \gamma = 1. \end{cases}$$

This concludes the proof. □

D.4. Hessian Condition Number Analysis

Theorem D.4 (Hessian Condition Number Analysis)**.** *Let $\mathcal{H}_{Coupled}$ and $\mathcal{H}_{Decoupled}$ be the Hessian matrices of the coupled spatio-temporal loss and the disentangled loss with respect to the network parameters $\boldsymbol{\theta} \in \mathbb{R}^p$. Under inherent physical scale-mismatches between spatial and temporal dimensions, the condition numbers satisfy:*

$$\kappa(\mathcal{H}_{Decoupled}) \ll \kappa(\mathcal{H}_{Coupled}). \tag{37}$$

Proof. Consider the coupled loss optimized over the neural network parameters $\boldsymbol{\theta}$. Based on the linear decomposition in Eq. (S3), Thm. D.3, the coupled residual can be abstractly expressed as $\boldsymbol{r}_{\text{MeanFlow}}(\boldsymbol{\theta}) \approx \alpha \boldsymbol{r}_{\text{Temp}}(\boldsymbol{\theta}) + \beta \boldsymbol{r}_{\text{Spac}}(\boldsymbol{\theta})$, where $\alpha \sim \mathcal{O}(l_t)$ and $\beta \sim \mathcal{O}(l_s^{-1})$ denote the characteristic temporal and spatial scales. The coupled loss is given by:

$$\begin{aligned}\mathcal{L}_{\text{Coupled}}(\boldsymbol{\theta}) &= \frac{1}{2}\|\alpha \boldsymbol{r}_{\text{Temp}} + \beta \boldsymbol{r}_{\text{Spac}}\|_2^2 \\ &= \underbrace{\frac{1}{2}\alpha^2\|\boldsymbol{r}_{\text{Temp}}\|_2^2 + \frac{1}{2}\beta^2\|\boldsymbol{r}_{\text{Spac}}\|_2^2}_{\mathcal{L}_{\text{Decoupled}}(\boldsymbol{\theta})} + \underbrace{\alpha\beta\langle \boldsymbol{r}_{\text{Temp}}, \boldsymbol{r}_{\text{Spac}}\rangle}_{\mathcal{L}_{\text{Cross}}(\boldsymbol{\theta})}.\end{aligned} \tag{38}$$

Let $\boldsymbol{J}_T = \nabla_{\boldsymbol{\theta}} \boldsymbol{r}_{\text{Temp}} \in \mathbb{R}^{n\times p}$ and $\boldsymbol{J}_S = \nabla_{\boldsymbol{\theta}} \boldsymbol{r}_{\text{Spac}} \in \mathbb{R}^{n\times p}$ be the Jacobian matrices of the temporal and spatial residuals, respectively. Applying the Gauss-Newton approximation (neglecting second-order residual terms), the Hessian of the decoupled loss is a well-conditioned sum of two positive semi-definite (PSD) matrices:

$$\mathcal{H}_{\text{Decoupled}} \approx \alpha^2 \boldsymbol{J}_T^\top \boldsymbol{J}_T + \beta^2 \boldsymbol{J}_S^\top \boldsymbol{J}_S. \tag{39}$$

Conversely, the Hessian of the coupled loss introduces explicit cross-Jacobian interactions:

$$\mathcal{H}_{\text{Coupled}} \approx \mathcal{H}_{\text{Decoupled}} + \underbrace{\alpha\beta\left(\boldsymbol{J}_T^\top \boldsymbol{J}_S + \boldsymbol{J}_S^\top \boldsymbol{J}_T\right)}_{\mathcal{H}_{\text{Cross}}}. \tag{40}$$

In stiff physical systems, temporal variations and spatial gradients evolve at vastly mismatched scales, leading to highly skewed singular values between $\boldsymbol{J}_T$ and $\boldsymbol{J}_S$. The matrix $\mathcal{H}_{\text{Cross}}$ is highly indefinite and dense. According to Weyl's inequality for Hermitian matrices, the minimum eigenvalue of the coupled Hessian is bounded by:

$$\lambda_{\min}(\mathcal{H}_{\text{Coupled}}) \leq \lambda_{\min}(\mathcal{H}_{\text{Decoupled}}) + \lambda_{\max}(\mathcal{H}_{\text{Cross}}). \tag{41}$$

Because $\mathcal{H}_{\text{Cross}}$ is indefinite and dominated by the scale mismatch $\mathcal{O}(\alpha\beta)$, its extreme eigenvalues perturb the spectrum of $\mathcal{H}_{\text{Coupled}}$ violently. This interference dramatically suppresses $\lambda_{\min}(\mathcal{H}_{\text{Coupled}})$ toward zero (or creates pathological ravines in the optimization landscape) while inflating $\lambda_{\max}$, thereby severely escalating the condition number:

$$\kappa(\mathcal{H}_{\text{Coupled}}) = \frac{\lambda_{\max}(\mathcal{H}_{\text{Coupled}})}{\lambda_{\min}(\mathcal{H}_{\text{Coupled}})} \gg \frac{\lambda_{\max}(\mathcal{H}_{\text{Decoupled}})}{\lambda_{\min}(\mathcal{H}_{\text{Decoupled}})} = \kappa(\mathcal{H}_{\text{Decoupled}}). \tag{42}$$

By optimizing $\mathcal{L}_{\text{Temp}}$ and $\mathcal{L}_{\text{Spac}}$ independently, the decoupled strategy explicitly annihilates the pathological cross-term $\mathcal{H}_{\text{Cross}}$, ensuring a geometrically stable and well-conditioned gradient flow. □

E. Examples of Spatio-Temporal MeanFlow Losses

Example 1: d'Alembert's equation The proposed method applys to the equations

$$\begin{cases}\dfrac{\partial \boldsymbol{u}}{\partial t} + \dfrac{\partial \boldsymbol{u}}{\partial \boldsymbol{x}}\boldsymbol{c}(\boldsymbol{u}) = \boldsymbol{0}, \\ \boldsymbol{u}(\boldsymbol{x}, 0) = \boldsymbol{u}_0(\boldsymbol{x}),\end{cases}$$

the variables in our method are $\nabla^\top \boldsymbol{u} = \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{x}}, \boldsymbol{f} = -\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{x}}\boldsymbol{c}(\boldsymbol{u})$, hereby Eq. (18) obtains

$$\begin{aligned}&(t-\tau)\boldsymbol{m} - \left(-\boldsymbol{m}(\boldsymbol{x}-\boldsymbol{\xi})^\top + l^2\frac{\partial \boldsymbol{m}}{\partial \boldsymbol{\xi}}\right)\boldsymbol{c}(\boldsymbol{u}) - l^2\frac{\partial \boldsymbol{m}}{\partial \tau} = \boldsymbol{0}, \\ \iff &\left[t-\tau+(\boldsymbol{x}-\boldsymbol{\xi})^\top\boldsymbol{c}(\boldsymbol{u})\right]\boldsymbol{m} - l^2\left(\frac{\partial \boldsymbol{m}}{\partial \boldsymbol{\xi}}\boldsymbol{c}(\boldsymbol{u}) + \frac{\partial \boldsymbol{m}}{\partial \tau}\right) = \boldsymbol{0},\end{aligned}$$

where $l^2 = (t-\tau)^2 + \|\boldsymbol{x}-\boldsymbol{\xi}\|_2^2$ and the equation allows the construction of a MeanFlow loss just as the original MeanFlow framework did.

Example 2(a): Navier-Stokes equations The Navier-Stokes equations are,

$$\begin{cases} \dfrac{\partial \boldsymbol{w}(\boldsymbol{x},t)}{\partial t} = -\dfrac{\partial \boldsymbol{w}(\boldsymbol{x},t)}{\partial \boldsymbol{x}} \cdot \boldsymbol{u}(\boldsymbol{x},t) + v\Delta \boldsymbol{w}(\boldsymbol{x},t) + \boldsymbol{f}(\boldsymbol{x}), \\ \boldsymbol{w}(\boldsymbol{x},t) = \nabla \times \boldsymbol{u}(\boldsymbol{x},t), \\ \nabla \cdot \boldsymbol{u}(\boldsymbol{x},t) = 0, \\ \boldsymbol{w}(\boldsymbol{x},0) = \boldsymbol{w}_0(\boldsymbol{x}), \forall x \in D. \end{cases} \tag{43}$$

We can eliminate $\boldsymbol{u}(\boldsymbol{x},t)$ by solving

$$\boldsymbol{u}(\boldsymbol{x},t) = \frac{1}{4\pi} \int_{\mathbb{R}^3} \frac{\boldsymbol{w}(\boldsymbol{y},t) \times (\boldsymbol{x}-\boldsymbol{y})}{\|\boldsymbol{x}-\boldsymbol{y}\|_2^3} \, \mathrm{d}\boldsymbol{y},$$

using the Biot–Savart integration under the condition that $\nabla \cdot \boldsymbol{u} = 0$.

F. Experiments

F.1. Details of Synthetic PDE Benchmarks

We follow the data generation protocol of DiffusionPDE (Huang et al., 2024) and use four synthetic PDE benchmarks: 1D Burgers' equation, 2D non-bounded Navier–Stokes equation, 2D Darcy flow, and 2D Poisson equation. For each benchmark, we use 10,000 samples and split them into training, validation, and test sets with a ratio of 7:2:1, resulting in 7,000 training samples, 2,000 validation samples, and 1,000 test samples.

F.1.1. 1D Burgers' Equation

Governing equation. We consider the one-dimensional viscous Burgers' equation on $\Omega = (0,1)$ with periodic boundary conditions:

$$\partial_t u(c,t) + \partial_c \left(\frac{u^2(c,t)}{2} \right) = \nu \partial_{cc} u(c,t), \quad c \in \Omega, \quad t \in (0,T].$$

The initial condition is

$$u(c,0) = u_0(c),$$

and the viscosity is set to $\nu = 0.01$.

Numerical discretization and data generation. Following DiffusionPDE, the spatial resolution is 128, and each trajectory contains 128 temporal states, including the initial state and 127 future time steps. The time interval is $T = 1$. Initial conditions are sampled from a one-dimensional Gaussian random field with parameters $\gamma = 2.5$, $\tau = 7$, and $\sigma = 7^2$. The equation is solved using the spectral time-stepping solver used in the DiffusionPDE/FNO-style Burgers data generation code. The resulting data are stored as trajectories of shape $N \times X \times T$, where $N = 10{,}000$, $X = 128$, and $T = 128$.

Use in our method. For Burgers' equation, our model takes the initial condition $u_0(c)$ as input and predicts the full spatio-temporal solution trajectory $u(c,t)$.

F.1.2. 2D Non-bounded Navier–Stokes Equation

Governing equation. We consider the two-dimensional incompressible Navier–Stokes equation in vorticity form:

$$\partial_t \omega(c,t) + \boldsymbol{v}(c,t) \cdot \nabla \omega(c,t) = \nu \Delta \omega(c,t) + q(c), \quad c \in \Omega, \quad t \in (0,T],$$

with the incompressibility constraint

$$\nabla \cdot \boldsymbol{v}(c,t) = 0.$$

Here, $\omega = \nabla \times \boldsymbol{v}$ is the vorticity, $\boldsymbol{v}$ is the velocity field, and $q(c)$ is the forcing term. We set $\nu = 10^{-3}$, corresponding to Reynolds number $\mathrm{Re} = 1000$.

Numerical discretization and data generation. The spatial domain is discretized on a 128×128 grid. Following DiffusionPDE, the initial vorticity field is sampled from a two-dimensional Gaussian random field with $\alpha = 2.5$ and $\tau = 7$. The forcing term is fixed as

$$q(c) = 0.1\left(\sin(2\pi(x+y)) + \cos(2\pi(x+y))\right).$$

The equation is solved in Fourier space with a pseudo-spectral method, de-aliasing, and a Crank–Nicolson update. The internal time step is 10^{-4}, the final time is $T = 1$, and 10 snapshots are recorded for each trajectory. We use ten files, each containing 1,000 samples, giving 10,000 samples in total. Each file stores the initial vorticity field with shape $1000 \times 128 \times 128$ and the solution trajectory with shape $1000 \times 128 \times 128 \times 10$.

Use in our method. For Navier–Stokes, our model takes the initial vorticity field $\omega(c, 0)$ as input and predicts the future vorticity fields over the recorded time steps. The forcing term and boundary/domain setting are fixed across the dataset.

F.1.3. 2D Darcy Flow

Governing equation. We consider the steady-state Darcy flow equation on $\Omega = (0,1)^2$:

$$-\nabla \cdot (a(\mathbf{c})\nabla u(\mathbf{c})) = q(\mathbf{c}), \quad \mathbf{c} \in \Omega,$$

with homogeneous Dirichlet boundary condition

$$u(\mathbf{c}) = 0, \quad \mathbf{c} \in \partial\Omega.$$

Here, $a(\mathbf{c})$ is the permeability coefficient, $u(\mathbf{c})$ is the pressure field, and the forcing term is fixed as $q(\mathbf{c}) = 1$.

Numerical discretization and data generation. The domain is discretized on a 128×128 grid. Following DiffusionPDE, a Gaussian random field $g(\mathbf{c})$ is sampled with parameters $\alpha = 2$ and $\tau = 3$, and the log-normal permeability field is constructed as

$$a(\mathbf{c}) = \exp(g(\mathbf{c})).$$

The PDE is solved by assembling a sparse finite-difference linear system for the elliptic operator with zero boundary values. We use 10,000 Darcy samples. The coefficient field has shape $10000 \times 128 \times 128$, and the corresponding pressure solution has the same spatial resolution.

Use in our method. For Darcy flow, our model is conditioned on the permeability field $a(\mathbf{c})$ and the prescribed boundary condition, and predicts the pressure solution $u(\mathbf{c})$. Since the boundary condition is homogeneous and shared by all samples, it is treated as fixed conditioning in the benchmark.

F.1.4. 2D Poisson Equation

Governing equation. We consider the two-dimensional Poisson equation on $\Omega = (0,1)^2$:

$$\Delta u(c) = q(c), \quad c \in \Omega,$$

with homogeneous Dirichlet boundary condition

$$u(c) = 0, \quad c \in \partial\Omega.$$

Here, $q(c)$ is the source field and $u(c)$ is the solution field.

Numerical discretization and data generation. The domain is discretized on a 128×128 grid. Following DiffusionPDE, the source field $q(c)$ is sampled from a Gaussian random field with parameters $\alpha = 2$ and $\tau = 3$. The Poisson equation is solved using a finite-difference discretization with a standard five-point Laplacian stencil and homogeneous Dirichlet boundary values. We use 10,000 samples. The source field has shape $10000 \times 128 \times 128$, and the corresponding solution field has the same spatial resolution.

Use in our method. For the Poisson equation, our model is conditioned on the source field $q(c)$ and the prescribed boundary condition, and predicts the solution field $u(c)$. As in Darcy flow, the boundary condition is fixed for all samples.

Table 4. Hyperparameter settings for synthetic PDE benchmarks.

Dataset	Backbone	Batch size	Learning rate	Epochs	Loss weights
Burgers	Conditioned FNO1d	8	5.5×10^{-4}	200	$\lambda_t = \lambda_x = 10^{-2}$
Navier–Stokes	Fourier U-Net	5	2.0×10^{-4}	100	$\lambda_t = \lambda_x = 10^{-2}$
Darcy	Fourier U-Net	4	3.0×10^{-4}	300	$\lambda_x = 10^{-2}$
Poisson	Fourier U-Net	4	3.0×10^{-4}	300	$\lambda_x = 10^{-2}$

F.2. Details of the Real-world HYCOM Dataset

We use a real-world ocean dynamics dataset derived from HYCOM for evaluating the scalability and robustness of the proposed method. HYCOM, the Hybrid Coordinate Ocean Model, is a global ocean analysis and forecasting system. Different from purely synthetic PDE benchmarks, HYCOM products are generated by combining numerical ocean model dynamics with real observations through data assimilation. The assimilated observations include satellite measurements and in-situ ocean observations. Therefore, the resulting fields reflect realistic ocean variability, while also containing practical complications such as observation noise, assimilation artifacts, imperfect boundaries, and possible mismatch between the observed dynamics and idealized PDE assumptions.

Data source. The data are obtained from the public HYCOM GLBy0.08 product through the THREDDS/OPeNDAP service:

`https://tds.hycom.org/thredds/dodsC/GLBy0.08/latest`.

We use the variable `water_temp`, which represents three-dimensional ocean water temperature. The original HYCOM product provides global ocean fields over longitude, latitude, depth, and time. In our experiment, we select a local ocean region and use the first 8 depth levels to construct a three-dimensional spatio-temporal prediction task.

Selected region and variables. We extract a local spatial block from the HYCOM grid using latitude indices $[1100, 1260)$ and longitude indices $[2900, 3060)$. This corresponds approximately to latitudes from -36.0° to -29.64° and longitudes from 232.0° to 244.72° in the HYCOM convention, i.e., approximately 128.0°W to 115.28°W. The selected depth levels are $0,\ 2,\ 4,\ 6,\ 8,\ 10,\ 12,\ 15$ meters.

The raw extracted block has shape $32 \times 8 \times 160 \times 160$, corresponding to time, depth, latitude, and longitude dimensions, respectively.

Preprocessing. We normalize the extracted temperature field using the mean and standard deviation computed over the selected raw block. The statistics used in our processed dataset are $\mu = 23.01293796527723, \quad \sigma = 1.3310423861925285$. The raw temperature values in the selected block range approximately from 18.8700 to 26.1490. After normalization, we construct local spatio-temporal samples using a sliding-window procedure. Each sample contains 9 time steps on a spatial grid of size $32 \times 32 \times 8$. The spatial stride is set to 4 along the two horizontal dimensions. This produces a total of 26136 samples, each with shape $9 \times 32 \times 32 \times 8$.

Dataset split. We split the processed samples into training, validation, and test sets using an 80/10/10 split: 20908 training samples, 2614 validation samples, 2614 test samples. All methods are trained and evaluated on the same split.

Prediction task. Given the initial three-dimensional temperature field and the spatio-temporal coordinates, the model predicts the future temperature evolution over the 9-step temporal window. Unlike idealized synthetic PDE data, the HYCOM data are not generated from a single clean closed-form PDE with perfectly specified boundary conditions. Instead, they are data-assimilative ocean fields influenced by numerical model dynamics, observations, and assimilation procedures. This makes the benchmark more representative of real deployment settings, where data may be noisy, partially observed, and only approximately consistent with simplified PDE assumptions.

F.3. Implementation Details of Sec. 4.1

For all PDEs, four finite-interval pairs (τ, t) are randomly sampled at each iteration to construct the MeanFlow regularization. For steady-state PDEs, the model takes the coefficient/source field together with the prescribed boundary condition as input. Models are trained with a supervised relative L_2 loss and the proposed MeanFlow regularization. All results are reported using the checkpoint with the lowest validation relative L_2 error.

For Burgers' equation, we use a conditioned FNO1d. The model takes the current solution state and spatial coordinate as input, and is conditioned on the interval information. It uses Fourier modes $[12, 12, 12, 12, 12]$, channel width 96, fully connected hidden dimension 192, GELU activation, and a conditioning MLP with hidden dimension 224.

For Navier–Stokes, Darcy flow, and Poisson equation, we use a Fourier U-Net backbone. The Fourier U-Net follows an encoder–decoder architecture with skip connections. The initial feature projection is a 3×3 convolution with 48 hidden channels. The encoder has four resolution levels with channel multipliers $(1, 2, 2, 4)$, and each level contains two residual blocks. In the first two encoder levels, the standard convolutional residual blocks are replaced by Fourier residual blocks, where each block combines spectral convolution layers with local 1×1 convolution branches. The remaining encoder blocks, the middle block, and the decoder use convolutional residual blocks. The decoder upsamples features back to the original resolution and concatenates skip features from the encoder. We use GELU activation, no attention blocks, and no normalization.

For Navier–Stokes, the Fourier U-Net input contains the current vorticity field and the two spatial coordinate channels. For Darcy flow and Poisson equation, the Fourier U-Net uses two scalar input channels: the harmonic extension of the boundary condition and the coefficient/source field. Darcy flow and Poisson equation use Fourier modes $(8, 8)$ in the first Fourier level, while Navier–Stokes uses the default Fourier modes $(12, 12)$. The number of retained modes is reduced at lower resolutions according to the U-Net scale. Table 4 summarizes the main hyperparameters used in our experiments.

F.4. Implementation Details for HYCOM

This section provides implementation details for applying our MeanFlow-based solver to the real-world HYCOM ocean dataset. The goal of this experiment is to evaluate whether the proposed framework can scale to realistic three-dimensional spatio-temporal data and remain effective when the observed dynamics are not generated from an ideal synthetic PDE.

Input and output format. Each processed HYCOM sample has shape $9 \times 32 \times 32 \times 8$, where the first dimension denotes time and the remaining dimensions denote the three-dimensional spatial grid. We denote the normalized temperature field as $u \in \mathbb{R}^{T \times X \times Y \times Z}$, with $T = 9$, $X = 32$, $Y = 32$, and $Z = 8$. In implementation, the data are rearranged to $u \in \mathbb{R}^{X \times Y \times Z \times T}$. The initial state u_0 is used as the conditioning field, and the model predicts the future temperature evolution over the full temporal window.

Model input. For each query time step, we construct a five-channel input tensor $\mathbf{x} = (x, y, z, t, u_0)$, where (x, y, z) are normalized spatial coordinates, t is the normalized temporal coordinate, and u_0 is the initial three-dimensional temperature field broadcast to all query times. Thus, for each time step, the model input has shape $32 \times 32 \times 8 \times 5$. The model outputs a scalar temperature-related quantity at each spatial location.

Network architecture. We use a three-dimensional Fourier neural operator backbone. The model applies spectral convolution over the three spatial dimensions and predicts the temperature evolution conditioned on the initial state and coordinates. Unless otherwise specified, the HYCOM experiment uses the following architecture: $\text{modes}_x = [8, 8, 8, 8]$, $\text{modes}_y = [8, 8, 8, 8]$, $\text{modes}_z = [4, 4, 4, 4]$, with channel width 20, fully connected dimension 64, and hidden layers $[20, 20, 20, 20, 20]$. The activation function is GELU. No additional padding is used.

Training protocol. We train the model for 101 epochs using Adam with learning rate 10^{-3}. The batch size is 2. A multi-step learning rate schedule is used with milestones at epochs 60 and 90 and decay factor 0.5. Validation is performed every 10 epochs. The checkpoint with the best validation loss is selected, and the corresponding test loss is reported.

F.5. Implementation Details for the Inverse Darcy Experiment

Problem setup. We consider the inverse Darcy flow problem: given a fully observed pressure field $u^{\text{obs}} \in \mathbb{R}^{S \times S}$, recover the permeability field $a(x)$ satisfying $-\nabla \cdot (a \nabla u) = 1$ with homogeneous Dirichlet boundary conditions.

Forward model. The forward MeanFlow model is trained on the forward Darcy task as described in Section F.1.3. Its weights are fully frozen during inverse optimization; no fine-tuning is performed.

Initial field construction. Rather than initializing the input state z_0 to the zero field, we construct a *harmonic extension* of the boundary values of u^{obs}. Concretely, we fix the boundary pixels of z_0 to match $u^{\text{obs}}|_{\partial\Omega}$ and iteratively average interior values over 100 Jacobi iterations:

$$z_0^{(k+1)}(x) = \frac{1}{4} \sum_{x' \sim x} z_0^{(k)}(x'), \quad z_0|_{\partial\Omega} = u^{\text{obs}}|_{\partial\Omega}. \tag{44}$$

This initialization satisfies the boundary condition and provides a spatially smooth starting point, which we find stabilizes early optimization steps.

Hyperparameters. We run Adam for 2000 steps with initial learning rate 5×10^{-3}, decayed to 10^{-4} via cosine annealing. Gradient norms are clipped to 1.0. All experiments use a single NVIDIA GPU.

F.6. Compared Methods

We compare our method with baselines from three representative paradigms: physics-informed solvers, neural operator solvers, and generative PDE solvers.

PINN. Physics-informed neural networks directly parameterize the solution function with a neural network and train it by minimizing initial-condition, boundary-condition, and PDE residual losses. PINNs do not require paired simulation data, but their optimization can become unstable when different loss terms have conflicting scales or gradients.

I-PINN. I-PINN improves the vanilla PINN framework by combining an enhanced feed-forward architecture with adaptive loss weighting under upper-bound constraints. This design aims to mitigate gradient-flow stiffness and reduce gradient-related training failures. Compared with PINN, I-PINN improves convergence and accuracy without introducing substantial extra computational complexity.

FNO. Fourier Neural Operator is a data-driven neural operator that learns mappings between function spaces. It uses Fourier-domain integral operator layers to model global dependencies and can be evaluated on different spatial discretizations. In our experiments, FNO serves as a standard data-driven operator-learning baseline.

DeepONet. DeepONet is a neural operator architecture designed to learn nonlinear operators. It uses a branch network to encode the input function sampled at sensor locations and a trunk network to encode output coordinates. The two representations are combined to predict the output function value. Therefore, DeepONet is categorized as an operator-learning method rather than a standard pointwise neural network.

PINO. Physics-informed Neural Operator combines supervised operator learning with PDE residual constraints. PINO is built on the FNO framework and imposes physics losses, potentially at higher resolutions than the supervised data. It is a hybrid baseline between data-driven operator learning and physics-informed training.

CFO. Continuous Flow Operator learns continuous-time PDE dynamics using a flow-matching objective. Instead of autoregressively predicting future states on a fixed time grid, CFO fits temporal splines to trajectory data and trains a neural operator to predict analytic velocity fields. During inference, the learned continuous vector field is integrated with an ODE solver. We classify CFO as a continuous-time neural operator baseline.

DiffusionPDE. DiffusionPDE formulates PDE solving as generative modeling under partial observation. It models the joint distribution of solution and coefficient fields, allowing it to infer missing information and solve forward or inverse PDE problems. This makes it a representative diffusion-based generative PDE solver.

Rectified Flow. Rectified Flow learns a conditional velocity field that transports input laws to output solution laws along nearly straight trajectories. Compared with diffusion models that often require many stochastic denoising steps, rectified-flow surrogates use deterministic ODE sampling and can reduce the number of inference steps substantially.

PBFM. Physics-Based Flow Matching is a physics-constrained generative flow-matching method. It addresses the conflict between distributional fidelity and physical consistency by enforcing physical constraints during training with conflict-free gradient updates. Unlike methods that rely on inference-time correction, PBFM incorporates physics constraints into the generative training process.

F.7. Additional Experimental Results of Sec. 4.1

Additional Visual Comparisons. In this section, we provide further qualitative comparisons between our proposed framework and baselines on NS dataset. Fig. 5 visualizes the ground truth and the predicted solutions. As illustrated, our method consistently yields predictions that are most aligned with the ground truth, especially in capturing fine-grained structures and high-frequency features. For instance, in the Navier–Stokes equation, while some baseline models struggle with the turbulent regions over long-term integration, our model maintains high physical fidelity.

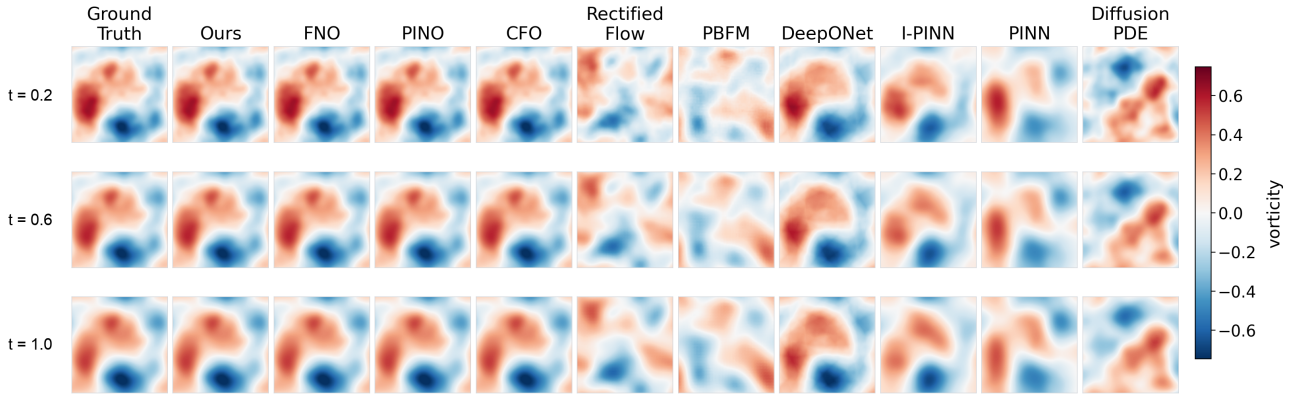

Figure 5. Visual comparison of predicted fields and error maps on ns dataset.

Training Loss Trajectories. We present the training trajectories in Fig. 6. The plots depict the evolution of the total loss, the data-driven loss ($\mathcal{L}_{\text{data}}$), and the proposed temporal and spatial MeanFlow losses ($\mathcal{L}_{t\text{-MF}}, \mathcal{L}_{s\text{-MF}}$). The results indicate that the data loss decreases rapidly during the initial epochs, allowing the model to capture the primary mapping between input conditions and solution fields. Subsequently, the temporal and spatial MeanFlow losses stabilize at low magnitudes, providing crucial physical regularizations that ensure the predicted fields satisfy the underlying PDE constraints.

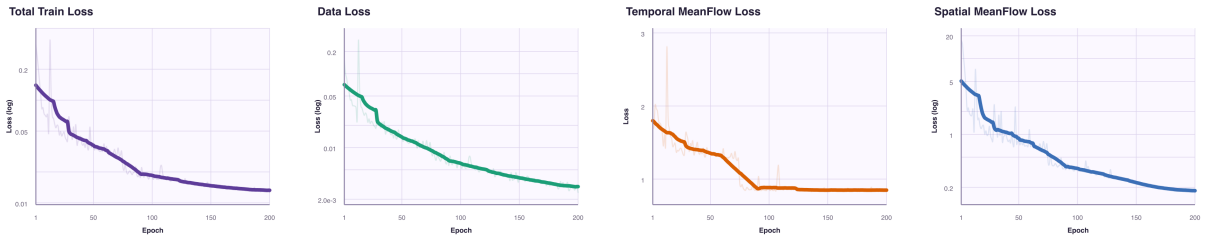

Figure 6. Training trajectories for all loss components, including the total loss, the data loss, the temporal MeanFlow loss, and the spatial MeanFlow loss.

Additional Coefficient Analysis. We further analyze the sensitivity of Burgers' equation to different temporal and spatial MeanFlow loss coefficients. The coefficient pair $(\lambda_{\text{T-MF}}, \lambda_{\text{S-MF}})$ denotes the weights of the temporal MeanFlow loss and the spatial MeanFlow loss, respectively. All models are evaluated on the same base-resolution test set at $s = 128$, and we report the global relative ℓ_2 error.

As shown in Fig. 7, introducing the spatial MeanFlow loss substantially improves performance when the temporal MeanFlow coefficient is fixed at 0.01. In particular, increasing $\lambda_{\text{S-MF}}$ from 0 to a moderate value significantly reduces the error, while overly large coefficients do not further improve the result. Similarly, when fixing $\lambda_{\text{S-MF}} = 0.01$, using a moderate temporal

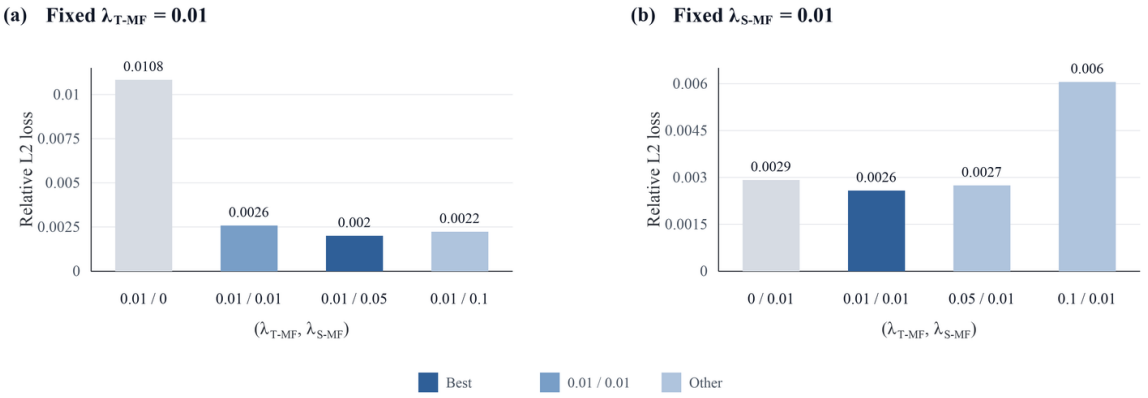

Figure 7. Additional coefficient analysis on Burgers' equation. Left: spatial MeanFlow coefficient sweep with fixed temporal MeanFlow coefficient $\lambda_{\text{T-MF}} = 0.01$. Right: temporal MeanFlow coefficient sweep with fixed spatial MeanFlow coefficient $\lambda_{\text{S-MF}} = 0.01$. The reported metric is global relative ℓ_2 error at the base resolution $s = 128$; lower is better.

MeanFlow coefficient gives the best performance, whereas a large temporal coefficient degrades accuracy. These results indicate that the temporal and spatial MeanFlow losses are complementary, but their coefficients should be balanced to avoid over-regularization.