Exploring Conditioning Strategies for Fourier Neural Operators

Wenxi Cai, Yimin Wang
caiwenxi@umich.edu,wyimin@umich.edu
University of Michigan
USA

ABSTRACT

Learning neural operators for parameter-dependent PDEs requires integrating physical parameters into the operator architecture. We systematically compare three conditioning strategies for Fourier Neural Operators (FNOs): local feature-wise modulation, global channel-wise scaling, and input-level concatenation. Using identical backbones, we evaluate all mechanisms on four PDE systems—advection, Burgers, elasticity, and Navier–Stokes—and show that conditioning consistently improves parameter generalization. Local modulation performs best on most benchmarks, while global scaling is most effective for linear elasticity. In a second stage, we apply the PDE-Refiner training scheme to the strongest method on each system and find that refinement improves some PDEs but has limited effect on Navier–Stokes due to its strongly advective dynamics. Our results provide a clear comparison of conditioning designs and practical guidance for building parameter-aware neural operators.

KEYWORDS

Neural Operators, PDE Forecasting, Parameter Generalization, Long-horizon Stability

1 INTRODUCTION

Learning efficient surrogate models for partial differential equations (PDEs) is an important goal in scientific machine learning. Many tasks in fluid dynamics, elasticity, and transport require repeated PDE evaluations under varying physical conditions, which makes traditional numerical solvers expensive for large ensembles or real-time deployment. Neural operators (NOs)[1–3, 7] provide a data-driven alternative by learning mappings between function spaces, avoiding the cost of running iterative numerical schemes.

Among existing operator-learning models, the Fourier Neural Operator (FNO)[6] is one of the most widely used due to its ability to capture long-range spatial interactions through spectral convolution. FNO-based methods have achieved strong performance in diverse applications, including weather forecasting[10] and seismic modeling[4]. However, standard FNO architectures have no clear mechanism for integrating physical parameters such as advection speeds, viscosity, forcing coefficients, or wave speeds. These parameters significantly influence solution structure.

This motivates a central question:

How should physical parameters be incorporated into FNO to improve accuracy on parameter-varying PDEs?

We propose architectural modifications to the FNO that define how physical parameters interact with its spectral blocks. Our design is instantiated through three conditioning mechanisms that inject parameter information at different stages of the operator:

- **Local Condition.** The parameter vector is encoded by an MLP to produce channel-wise scale and shift values. These values are broadcast over spatial coordinates and applied to intermediate spectral features through pointwise linear modulation, allowing the parameters to adjust each Fourier block directly.
- **Global Condition.** The parameters are mapped to a latent vector of size C_h, broadcast spatially, and used as channel-wise multiplicative factors on the learned feature map. This introduces a global adjustment per channel without modifying spatial patterns.
- **Input Concatenation.** The parameters are broadcast across the grid and appended as extra channels before the first Fourier layer. This gives the model access to parameter information without altering the internal spectral transformations.

These mechanisms represent the primary strategies through which parameter information can enter an operator-learning model: local feature modulation, global channel scaling, and direct input augmentation. To compare their behavior across diverse physical regimes, we evaluate all three mechanisms on four PDE benchmarks with different dynamics and parameter dependencies:

- **2D Advection–Diffusion** with random velocities (c_x, c_y);
- **2D Burgers** equation with viscosity ν;
- **2D Elastic Wave** propagation with wave speeds (c_p, c_s);
- **2D Navier–Stokes** in vorticity form with forcing coefficients $(\text{coef}_x, \text{coef}_y)$.

We train sixteen models in total: three conditioned variants and one unconditioned baseline on each benchmark, to assess how each conditioning strategy interacts with different PDE dynamics, including transport, diffusion, oscillations, and nonlinear fluid flow. For the best-performing strategy on each benchmark, we apply the refinement technique introduced in [8], adjusting only hyperparameters while keeping the model structure unchanged.

This study examines how different conditioning mechanisms influence the behavior of FNO. By treating conditioning as the central design variable and evaluating three representative strategies across several PDE systems, we aim to provide a clear comparison that can inform the construction of parameter-aware operator models.

2 PROBLEM DEFINITION

We study parameter-dependent partial differential equations (PDEs) of the form

$$\mathcal{F}(u; \mathbf{p}) = 0,$$

where $u : \Omega \to \mathbb{R}^C$ is the solution field defined on a spatial domain $\Omega \subset \mathbb{R}^2$, and $\mathbf{p} \in \mathbb{R}^d$ denotes a vector of physical parameters such as advection velocities, viscosity, wave speeds, or forcing coefficients.

In many scientific and engineering settings, the parameters $\mathbf{p}$ vary across instances, and the goal is to construct a surrogate model that predicts the corresponding solution fields efficiently.

Let u_0 be an initial condition and let u denote the resulting solution field at a fixed target time $t^\star$ after numerical integration of the PDE. We consider datasets of the form

$$\mathcal{D} = \{(u_0^{(i)}, \mathbf{p}^{(i)}, u^{(i)})\}_{i=1}^N,$$

where each sample is generated by solving the PDE under a specific set of parameters $\mathbf{p}^{(i)}$. Our original objective is to learn an operator

$$\mathcal{G}_\theta : (u_0, \mathbf{p}) \mapsto \hat{u},$$

where $\hat{u}$ is the model prediction of the PDE solution at time $t^\star$.

We now extend this setting to time-dependent data. Instead of observing only the solution at a single target time $t^\star$, we consider trajectories sampled on a uniform temporal grid. For each PDE instance, we observe the solution at times

$$t_k = k\Delta t, \quad k = 0, 1, \ldots, K, \quad t_K = t^\star,$$

so that each sample provides a sequence $\{u^{(i)}(t_k)\}_{k=0}^K$ together with the corresponding parameter vector $\mathbf{p}^{(i)}$. The learning problem is to construct a surrogate that uses the observed state and parameters to predict the future states along this grid.

To make use of the temporal structure, we introduce a one-step operator

$$\mathcal{G}_\theta : (u(t_k), \mathbf{p}) \mapsto \hat{u}(t_{k+1}),$$

and train it on pairs $\left(u^{(i)}(t_k), \mathbf{p}^{(i)}\right) \mapsto u^{(i)}(t_{k+1})$ across all trajectories and time indices. At inference time, we generate a rollout by applying $\mathcal{G}_\theta$ autoregressively. Starting from the initial condition $u_0 = u(0)$, we form

$$\hat{u}(t_1) = \mathcal{G}_\theta(u_0, \mathbf{p}), \hat{u}(t_2) = \mathcal{G}_\theta(\hat{u}(t_1), \mathbf{p}), \ldots, \hat{u}(t_K) = \mathcal{G}_\theta(\hat{u}(t_{K-1}), \mathbf{p}).$$

Rollout introduces an additional source of error growth beyond single-step prediction. Suppose that, on average, the relative error of one step is of order $a\%$, in the sense that

$$\frac{\|\mathcal{G}_\theta(u(t_k), \mathbf{p}) - u(t_{k+1})\|}{\|u(t_{k+1})\|} \approx a\%.$$

If local errors are amplified multiplicatively through repeated application of $\mathcal{G}_\theta$, then after n rollout steps the deviation factor scales approximately like $(1 + a\%)^n$. Such exponential amplification is common in time-dependent PDEs whose dynamics amplify perturbations. This makes it necessary to reduce the one-step prediction error as much as possible.

3 RELATED WORK

Neural operators for PDEs. Neural operators learn mappings between function spaces and have become a versatile framework for PDE surrogate modeling. DeepONet[9] represents nonlinear operators through a branch–trunk decomposition, while graph-based neural operators[12] extend operator learning to irregular meshes. Kernel-based and attention-guided variants[1–3] further enhance expressiveness for fluid dynamics, elasticity, and multiscale transport. These models focus primarily on architectural flexibility, whereas the question of how physical parameters should be integrated into operator layers remains relatively unstudied.

Fourier Neural Operators. The FNO [6] performs spectral convolution through Fourier-space weights, enabling global spatial coupling and strong resolution transfer. Extensions include multi-resolution variants[5], geometry-aware formulations[3], and domain-specific adaptations used in forecasting and seismic inversion[4, 10]. These works improve spatial modeling but generally treat physical parameters as auxiliary inputs appended to the initial condition. As a result, the way parameters interact with spectral representations is rarely analyzed. Our work focuses on this aspect by examining how parameter signals can be injected at different points within the FNO computation.

Conditioning mechanisms. Conditioning strategies such as FiLM [11], channel-wise modulation, and input augmentation are widely used in generative modeling and supervised prediction. In PDE surrogate models, parameters are often introduced through input concatenation or global embeddings [13]. Recent studies have applied conditioning ideas to FNO-like architectures in specific applications. Zeng et al.[16] propose a conditional FNO for optical fiber propagation, and Zhu et al.[17] introduce FiLM and spectral gating inside FNO blocks as part of an equation-aware emulator. These approaches demonstrate the potential value of conditioning but integrate it with additional architectural changes tailored to particular domains. In contrast, our work isolates conditioning itself as the design variable and evaluates three representative mechanisms—local modulation, global channel scaling, and input concatenation—within an otherwise standard FNO architecture.

4 METHOD

We build on the FNO[6] and study how physical parameters should be integrated. This section first recalls the baseline FNO architecture and then introduces three conditioning mechanisms that inject parameters at different stages of the architecture. In what follows, we focus on the two-dimensional case for clarity, while the same structure extends to any spatial dimension.

4.1 Baseline Fourier Neural Operator

Let $x \in \mathbb{R}^{B\times\Omega}$ denote the input field on a regular grid, where Ω is the shape of each sample, and let $\mathcal{F}$ be the 2D discrete Fourier transform applied channel-wise. A standard FNO layer computes

$$h = \sigma\left(\mathcal{F}^{-1}\left(R_\theta(\mathcal{F}(x))\right) + Wx\right), \tag{1}$$

where R_θ multiplies a fixed set of low-frequency modes by learnable weights and leaves the remaining modes unchanged, W is a pointwise 1×1 convolution, and σ is a nonlinear activation. Stacking several layers yields an operator

$$\mathcal{G}_\theta : u_0 \mapsto \hat{u}.$$

The next two conditioning mechanisms are introduced by modifying Eq. 1.

4.2 Local Conditioning: Feature-Wise Modulation

We now introduce a parameter vector $\mathbf{p} \in \mathbb{R}^d$. We begin with the local conditioning mechanism, which modifies the intermediate feature representation of each Fourier block through feature-wise affine modulation. This design introduces parameter dependence

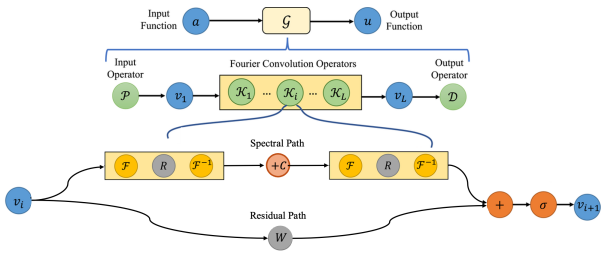

Figure 1: Illustration of the local conditioning mechanism. The physical parameters p are embedded into c, which controls feature-wise affine modulation inserted between two spectral transformations. The resulting operator is equivalent to a parameter-dependent convolution.

directly into the spectral path, allowing the operator to adjust its response for different physical regimes.

Parameter embedding. The physical parameters $\mathbf{p}$ are first expanded using a sinusoidal feature map [14, 15] with a fixed bank of frequencies and then transformed by a linear projection:

$$\mathbf{c}_{\text{loc}} = \eta_{\text{loc}}(\mathbf{p}) \in \mathbb{R}^{C_h},$$

where C_h is the hidden channel width.

Two-step spectral path with modulation. To allow conditioning to influence spectral computation at a deeper level, we adopt a two-step spectral path. The first spectral transformation produces an intermediate state:

$$\mathbf{h} = \mathcal{F}^{-1}\big(R_\theta^{(1)}(\mathcal{F}(x))\big) + W^{(1)}x,$$

where $R_\theta^{(1)}$ multiplies a set of Fourier modes by learned weights.

We then apply FiLM-style modulation based on $\mathbf{c}_{\text{loc}}$:

$$\tilde{\mathbf{h}} = \alpha(\mathbf{c}_{\text{loc}}) \odot \mathbf{h} + \beta(\mathbf{c}_{\text{loc}}),$$

with $\alpha, \beta : \mathbb{R}^{C_h} \to \mathbb{R}^{C_h}$ as learned affine projections.

The modulated representation flows into a second spectral block:

$$y = \mathcal{F}^{-1}\big(R_\theta^{(2)}(\mathcal{F}(\tilde{\mathbf{h}}))\big) + W^{(2)}\tilde{\mathbf{h}}.$$

Operator interpretation. Since FiLM modulation and the spectral transforms are linear in the feature space, the entire mapping forms a convolution whose kernel depends on $\mathbf{p}$. Formally:

Proposition 4.1. *Consider a spectral path*

$$x \xrightarrow{\mathcal{F}} R_\theta^{(1)} \xrightarrow{\mathcal{F}^{-1}} \mathbf{h} \xrightarrow{FiLM} \tilde{\mathbf{h}} \xrightarrow{\mathcal{F}} R_\theta^{(2)} \xrightarrow{\mathcal{F}^{-1}} y.$$

Then the overall mapping can be written as a parameter-dependent convolution

$$y = x * f_{|\mathbf{p}} + b_{|\mathbf{p}},$$

where $f_{|\mathbf{p}}$ and $b_{|\mathbf{p}}$ are defined by the Fourier coefficients of $R_\theta^{(1)}$, $R_\theta^{(2)}$, and the FiLM parameters $\alpha(\mathbf{c}_{\text{loc}})$ and $\beta(\mathbf{c}_{\text{loc}})$.

A full derivation of Proposition 4.1 is provided in Appendix A. This mechanism yields the most expressive form of parameter injection: each hidden channel can independently reweight and shift its spectral activation based on $\mathbf{p}$.

4.3 Global Conditioning: Channel-Wise Spectral Scaling

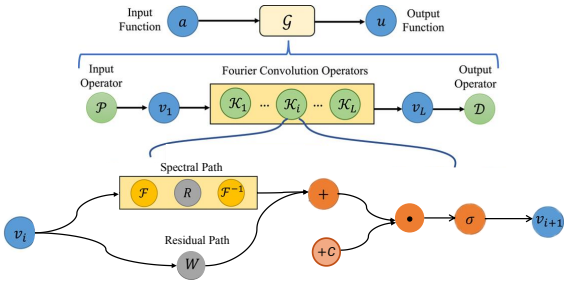

Figure 2: Illustration of the global conditioning mechanism. The physical parameters p are embedded into a global condition vector that modulates the operator through a shared affine transformation applied after the spectral update. The global modulation applies a single scale and shift across the entire feature map. This produces an operator whose response depends on p while preserving a uniform adjustment across all hidden channels.

We now introduce the global conditioning mechanism, where the physical parameters modulate the operator output through channel-wise scaling applied after the Fourier blocks. This mechanism follows the branch–trunk interaction used in DeepONet: the Fourier network acts as the trunk and the parameter encoder provides global coefficients that gate the trunk features.

Parameter embedding. The physical parameters $\mathbf{p}$ are processed totally the same as the ones in local conditioning, producing

$$\mathbf{c}_{\text{loc}} = \eta_{\text{loc}}(\mathbf{p}) \in \mathbb{R}^{C_h}.$$

Trunk operator and global gating. Let $\mathcal{G}_\theta$ denote the Fourier operator obtained by stacking several blocks as in Eq. 1:

$$y = \mathcal{G}_\theta(x) \in \mathbb{R}^{C_h \times \Omega}.$$

Global conditioning is applied at the output of this operator through direct channel-wise multiplication:

$$\tilde{y} = \mathbf{c}_{\text{glob}} \odot y,$$

where $\mathbf{c}_{\text{glob}}$ is broadcast over all spatial coordinates.

Operator interpretation. For fixed $\mathbf{p}$, the mapping

$$x \longmapsto \tilde{y}(x; \mathbf{p}) = \mathbf{c}_{\text{glob}} \odot \mathcal{G}_\theta(x)$$

is linear and translation-equivariant in x, hence it can be written as a parameter-dependent convolution

$$\tilde{y}(x; \mathbf{p}) = x * f_{|\mathbf{p}}.$$

The parameters do not modify the spatial structure of the kernel; instead they control the contribution of each channel produced by $\mathcal{G}_\theta$.

4.4 Input-Level Conditioning

The third conditioning strategy injects the parameter vector directly at the input layer by concatenating it with the spatial field x. This keeps the internal Fourier blocks unchanged and serves as the architecture with the smallest change.

Concatenation with the input field. The broadcasted parameter p is concatenated with the original input:

$$x_{\text{aug}} = \text{concat}(x,\ \mathbf{p}).$$

The augmented field x_{aug} is then passed to a standard FNO:

$$y = \text{FNO}(x_{\text{aug}}).$$

Operator interpretation. In this setting the parameters are treated as extra input channels that are constant in space. The network must infer how $\mathbf{p}$ affects the solution purely through the subsequent Fourier layers, without any explicit structure in the way $\mathbf{p}$ interacts with the spectral kernels.

4.5 Two-Stage Methodological Framework

Our study follows a two-stage methodological framework aimed at separating the effect of conditioning and evaluating its rollout performance under a refined training scheme.

Stage 1: comparison of conditioning mechanisms across PDE systems. In this stage, we train each conditioning mechanism. Each model uses the same backbone architecture, similar parameter counts, and the same spectral resolution. The goal is to determine, for each PDE system, which mechanism produces the strongest parameter-dependent operator.

Stage 2: refinement with enhanced training. In this stage, we take the best-performing mechanism for each benchmark and retrain it using the enhanced training scheme introduced in prior work [8], which uses iterative operator refinement with curriculum training. This refinement stage follows the original procedure without modification, with only a small number of hyperparameters adjusted to fit the characteristics of each PDE. This stage evaluates the maximum accuracy each PDE system can reach.

4.6 Explanation of the Refinement Scheme

To clarify the enhanced training scheme introduced in [8], we briefly summarize the refinement mechanism. The method replaces pure one-step training with an iterative denoising objective that forces the model to correct its own prediction across multiple noise levels.

At refinement step $k = 0$, the model performs a standard one-step update:

$$u(t) \approx \mathcal{G}_\theta\big(0,\ u(t-\Delta t),\ 0\big),$$

which recovers the classical MSE objective.

For $k > 0$, the ground-truth solution is perturbed by Gaussian noise whose variance decreases with the refinement index:

$$u_k(t) = u(t) + \varepsilon_k, \qquad \varepsilon_k \sim \mathcal{N}\big(0,\ \sigma_k^2 I\big), \qquad \sigma_k = \sigma_{\min}^{k/K}.$$

The training target becomes the noise itself:

$$\mathcal{G}_\theta\big(u_k(t),\ u(t-\Delta t),\ k\big) \approx \varepsilon_k.$$

This denoising objective forces the operator to model solution components across the whole frequency spectrum, since different noise levels highlight different spatial frequencies of the signal.

During inference, the refinement loop iteratively improves an initial prediction. Starting from

$$\hat{u}_1(t) = \mathcal{G}_\theta\big(0,\ u(t-\Delta t),\ 0\big),$$

each refinement step applies the inverse denoising update:

$$\hat{u}_{k+1}(t) = u_k(t) - \mathcal{G}_\theta\big(u_k(t),\ u(t-\Delta t),\ k\big)\,\sigma_k, \qquad k = 1, \dots, K.$$

As σ_k decreases, later iterations correct lower-amplitude frequency components, resulting in substantially improved long-time rollout stability.

5 EXPERIMENTS

5.1 Experimental Setup

5.1.1 PDE benchmarks. We evaluate the three conditioning mechanisms on four time-dependent PDE benchmarks constructed with Fourier pseudo-spectral solvers on periodic grids. All datasets are generated on regular 64×64 spatial grids with varying physical parameters, making them suitable for testing parameter-dependent operator learning. All the data are solved on $[0, 2]^2$ with periodic boundaries using a Fourier spectral discretization in space and a fourth-order Runge–Kutta integrator in time. Below we summarize the governing equations, conditioning parameters, and dataset construction for each benchmark.

Advection–Diffusion. This benchmark describes the transport and diffusion of a scalar field $u(x, y, t)$ under a spatially uniform but randomly sampled velocity field $v = (c_x, c_y)$ and a time-dependent source term:

$$\partial_t u = \nu \Delta u - v \cdot \nabla u + s(x, y, t),$$

with a fixed viscosity $\nu = 10^{-3}$. The velocity components (c_x, c_y) are sampled independently from $[-1, 1]$ and serve as the conditioning parameters. This system shows that the flow moves in different directions depending on the condition.

Burgers. The Burgers benchmark models a two-component velocity field $u(x, y, t) = (u_1, u_2)$ governed by a viscous Burgers system:

$$\partial_t u + (u \cdot \nabla)u = \nu \Delta u.$$

The viscosity ν is sampled uniformly from $[0.01, 0.1]$ and used as the conditioning parameter. This system shows how fast the flow will dissipate.

Elastic Waves. The elasticity benchmark simulates wave propagation of a displacement field $u(x, y, t) = (u_x, u_y)$ obeying a first-order formulation of the elastic wave equation:

$$\partial_t u = v, \qquad \partial_t v = (c_p^2 - c_s^2)\,\nabla(\nabla \cdot u) + c_s^2 \Delta u,$$

where v denotes velocity and (c_p, c_s) are the P-wave and S-wave speeds. We treat (c_p, c_s), sampled from $[0, 1]^2$, as the conditioning parameters. This system shows how fast the wave propagates in the x-direction and the y-direction, respectively.

Navier–Stokes. For the incompressible flow benchmark, we consider the Navier–Stokes equations in vorticity form:

$$\partial_t \omega + u \cdot \nabla \omega = \nu \Delta \omega + f(x, y),$$

where ω is the scalar vorticity and u is recovered from a stream function via $\omega = \Delta \psi$. We fix the viscosity at $\nu = 10^{-3}$ and introduce a divergence-free forcing f built from low-frequency sinusoidal

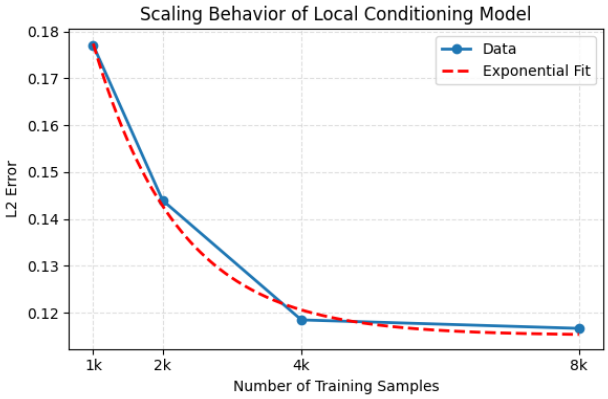

Figure 3: Scaling behavior of the local conditioning model as the number of training samples increases. Performance saturates at approximately 4k samples, indicating a data–capacity limit under fixed model size.

modes. The forcing amplitudes are controlled by two coefficients (α_x, α_y), sampled from $[0, 2]^2$, which serve as the conditioning parameters. This system will show how the flow move based on the condition.

5.1.2 Model variants. We compare four models: the FNO without conditioning and the three conditioning models. All models use the same hidden channel width of 32 and the same Fourier modes $(12, 12)$. For the local conditioning model, we use three blocks, while the other three models use six blocks so that all variants contain the same number of spectral convolutions. All conditioning inputs are embedded into a 128-dimensional vector.

5.1.3 Training procedure. Models are trained using Adam with a learning rate of 10^{-3} and cosine decay. Each model is trained until convergence. For both Stage 1 and Stage 2, we use the L^2 loss when $k = 0$, and in Stage 2 we switch to MSE loss when $k > 0$.

5.2 Scaling Law Analysis: Effect of Training Set Size

To understand how the proposed conditioning mechanisms behave as the amount of training data increases, we conduct a scaling study on the local conditioning model, which is the strongest performer in Stage 1 for most benchmarks. We fix all hyperparameters and architecture settings, and vary the number of training samples as

$$N \in \{1\text{k},\ 2\text{k},\ 4\text{k},\ 8\text{k}\}.$$

For each value of N, we train the model to convergence and evaluate on the same held-out test set.

Figure 3 summarizes the scaling behavior. The error decreases in an approximately exponential manner as the sample size increases, and by 4k samples the curve is already close to its limit. This suggests that, for this PDE family, the local conditioning mechanism reaches its data–capacity limit at moderate dataset sizes, and additional data does not significantly improve generalization unless model width or depth is increased.

Method	Advection	Burgers	Elasticity	Navier–Stokes
Local	**0.1050**	**0.0016**	0.3232	0.1185
Global	0.1262	0.0019	**0.2967**	0.1359
Input-Level	0.1343	0.0023	0.2979	**0.1175**
Vanilla FNO	0.9430	0.0369	0.5057	0.1490

Table 1: Stage 1 comparison of conditioning strategies across four PDE benchmarks. Values report L^2 error (↓).

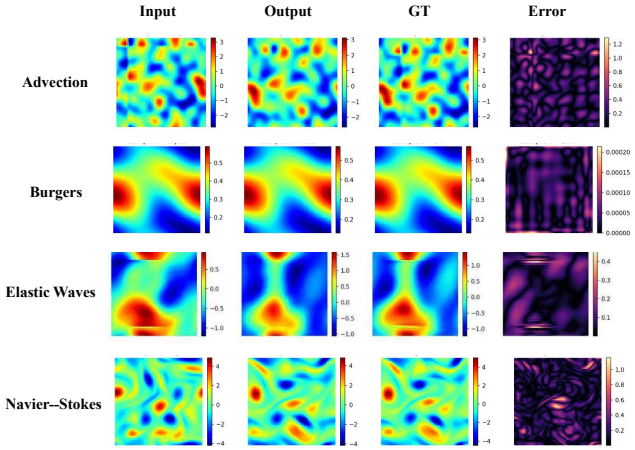

Figure 4: One-step predictions on all four PDE benchmarks using the best-performing conditioned FNO variant for each task. For every benchmark, we show the input field at the rollout start time, the predicted field, the ground-truth field at the same time, and the L^2 error.

5.3 Results Across PDE Benchmarks

Table 1 summarizes the Stage 1 performance of the three conditioning mechanisms and the vanilla FNO across the four PDE benchmarks. Across all systems, explicit conditioning yields improvements over the unconditioned baseline, showing the conditional models are effective.

Advection–Diffusion. This benchmark is the most sensitive to parameter shifts: the unconditioned FNO exhibits a large error of 0.9430, confirming that an operator trained on a limited velocity distribution generalizes poorly when the advective field changes. All three conditioning mechanisms significantly reduce the error, with local conditioning achieving the best performance at 0.1050. Global and input-level conditioning follow with moderately higher errors (0.1262 and 0.1343), indicating that the model is able to learn the direction of the flow, but the remaining errors accumulate across the uniform grid points.

Burgers. For Burgers' equation, viscosity controls the relative contribution of advection and diffusion. All conditioned models outperform the unconditioned FNO by a large margin (baseline 0.0369), and the best-performing two mechanisms—local (0.0016) and global (0.0019)—achieve nearly identical accuracy. This trend

reflects the global nature of viscosity: channel-wise spectral scaling already captures most of the required parameter dependence, so the additional affine modulation in local conditioning brings only a slight improvement. Input-level conditioning is still effective (0.0023) but remains the weakest of the three, consistent with its limited interaction with the spectral operator.

Elastic Waves. Because the elasticity system is a purely linear PDE, its parameter dependence enters the solution through linear changes in the P-wave and S-wave speeds. As a result, channel-wise scaling provides the most effective form of modulation. This is reflected in the results: global conditioning achieves the lowest error (0.2967), closely followed by input concatenation (0.2979), while local conditioning performs slightly worse (0.3232). The strong gap relative to the unconditioned baseline (0.5057) further shows that parameter awareness remains essential.

Navier–Stokes. Among the four conditioning mechanisms, input-level conditioning performs the best on the Navier–Stokes benchmark. However, as shown in Table 1, the performance differences among the four models are much smaller than in the other PDEs. This behavior reflects the structure of the equation: the dominant transport term $(u \cdot \nabla)u$ does not depend on the parameter, so the main vortex structures remain nearly unchanged across different viscosity values. The external forcing also contributes only weakly to the variation. As a result, all three conditioned models recover the large-scale flow in a similar way, while the errors are mainly concentrated in the thin filamentary regions where viscosity affects small-scale dissipation. This explains why the three conditioning mechanisms achieve very similar errors on this benchmark and why parameter conditioning only helps to a limited extent.

5.4 Stage 2 Results Across PDE Benchmarks

At this stage, we evaluated the refinement model on the Navier–Stokes dataset and observed that the results were unsatisfactory. As shown in the Fig 5, varying the refinement depth k leads to almost no change in the model's prediction. Inspecting the training process reveals that the loss remains around 0.1, indicating an error level above 10% and a clear sign of underfitting. This behavior may stem from the highly variable conditioning in this system, which makes it difficult for the model to learn a stable correction direction. Since the refinement steps do not effectively reduce the error, the model eventually becomes unstable and breaks down at about the sixth time step as shown in Fig 6.

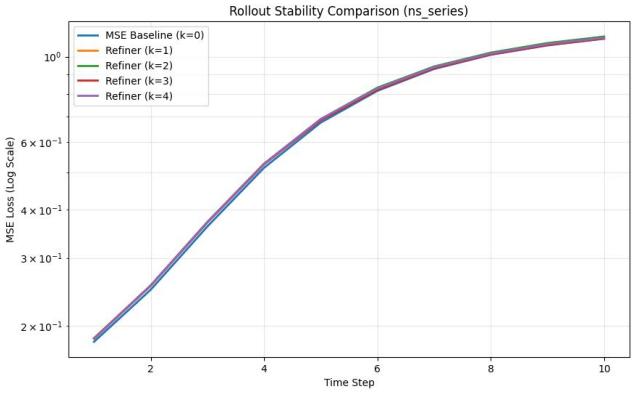

Figure 5: Across refinement depths $k = 1$ to $k = 4$, the error curves almost overlap with the baseline model ($k = 0$), indicating that refinement does not improve long-horizon stability for this system. The highly advective dynamics dominate the evolution, and the correction steps fail to reduce accumulated rollout error.

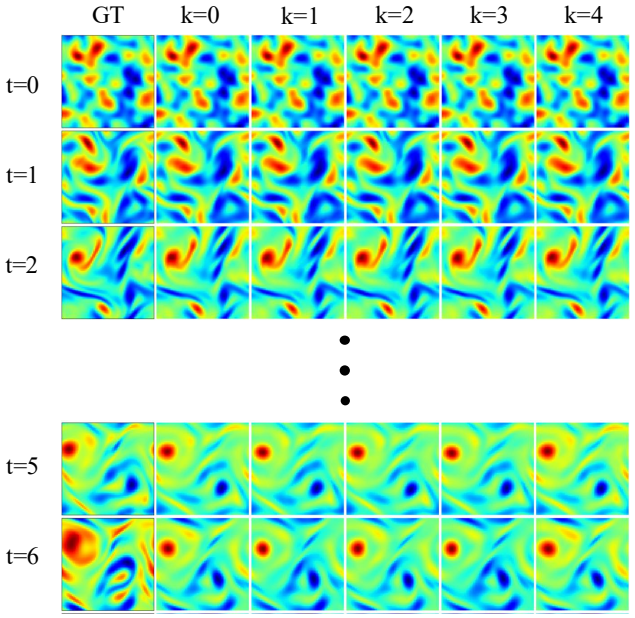

Figure 6: Across time steps $t = 0$ to $t = 6$, the predicted vorticity fields remain nearly identical for all refinement levels, showing no visible improvement over the baseline. This confirms that refinement does not alter the trajectory for this system, consistent with the dominance of the advective dynamics and the inability of the correction steps to reduce accumulated errors.

6 CONCLUSION AND DISCUSSION

This work provides a systematic comparison of how physical parameters should be incorporated into Fourier Neural Operators. By evaluating local modulation, global channel-wise scaling, and input-level concatenation across four PDE systems using identical backbones, we show that explicit conditioning is essential for parameter-dependent operator learning. Local modulation yields the strongest overall accuracy, particularly for nonlinear or parameter-sensitive systems such as advection and Navier–Stokes, while global scaling is most effective for linear elasticity, where its structure aligns naturally with the PDE. Input-level concatenation offers a simple but consistently weaker alternative. In the second stage, we applied the PDE-Refiner training scheme to the best method for each benchmark. Refinement improves some systems but has limited effect on Navier–Stokes, reflecting the dominance of advective dynamics and the difficulty of learning stable correction steps. Overall, our study

clarifies the roles of different conditioning strategies and offers practical guidance for designing parameter-aware neural operators.

REFERENCES

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

[2] Jean Kossaifi, Nikola Kovachki, Kamyar Azizzadenesheli, and Anima Anandkumar. 2023. Multi-grid tensorized fourier neural operator for high-resolution pdes. *arXiv preprint arXiv:2310.00120* (2023).

[3] Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. 2023. Neural operator: Learning maps between function spaces with applications to pdes. *Journal of Machine Learning Research* 24, 89 (2023), 1–97.

[4] Bian Li, Hanchen Wang, Shihang Feng, Xiu Yang, and Youzuo Lin. 2023. Solving seismic wave equations on variable velocity models with Fourier neural operator. *IEEE Transactions on Geoscience and Remote Sensing* 61 (2023), 1–18.

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

[6] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. 2020. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895* (2020).

[7] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. 2020. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485* (2020).

[8] Phillip Lippe, Bas Veeling, Paris Perdikaris, Richard Turner, and Johannes Brandstetter. 2023. Pde-refiner: Achieving accurate long rollouts with neural pde solvers. *Advances in Neural Information Processing Systems* 36 (2023), 67398–67433.

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

[10] Jaideep Pathak, Shashank Subramanian, Peter Harrington, Sanjeev Raja, Ashesh Chattopadhyay, Morteza Mardani, Thorsten Kurth, David Hall, Zongyi Li, Kamyar Azizzadenesheli, et al. 2022. Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators. *arXiv preprint arXiv:2202.11214* (2022).

[11] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. 2018. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32.

[12] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter Battaglia. 2020. Learning mesh-based simulation with graph networks. In *International conference on learning representations*.

[13] Makoto Takamoto, Francesco Alesiani, and Mathias Niepert. 2023. Learning neural pde solvers with parameter-guided channel attention. In *International Conference on Machine Learning*. PMLR, 33448–33467.

[14] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. 2020. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in neural information processing systems* 33 (2020), 7537–7547.

[15] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[16] Yuanhang Zeng, Xiangyu Ma, Qingzhe Cui, Zhu Guangzhi, and Xiao Zhu. 2025. Conditional Fourier neural operator for prediction of ultrafast nonlinear dynamics in optical fibers. *Journal of Lightwave Technology* PP (01 2025), 1–8. https://doi.org/10.1109/JLT.2025.3633748

[17] Qian-Ze Zhu, Paul Raccuglia, and Michael P Brenner. 2025. Generalizing PDE Emulation with Equation-Aware Neural Operators. *arXiv preprint arXiv:2511.09729* (2025).

SUPPLEMENTARY MATERIAL

A PROOF OF THEOREMS

Theorem A.1. *Let $\mathcal{F}$ and $\mathcal{F}^{-1}$ denote the Fourier transform and its inverse. Consider a spectral path that applies two consecutive spectral transformations with a FiLM modulation in between:*

$$\mathbf{h} = \mathcal{F}^{-1}\big(M_1(\xi)\,\mathcal{F}\{x\}\big), \qquad \tilde{\mathbf{h}} = \alpha(\mathbf{c})\,\mathbf{h} + \beta(\mathbf{c}), \qquad y = \mathcal{F}^{-1}\big(M_2(\xi)\,\mathcal{F}\{\tilde{\mathbf{h}}\}\big),$$

where $M_1(\xi)$ and $M_2(\xi)$ are frequency-wise multipliers, and $\alpha(\mathbf{c}), \beta(\mathbf{c})$ are per-channel scalars broadcast over space (i.e., spatially constant), with

$$\mathbf{c} = [\,\gamma(\tau),\ \eta(\mathbf{p})\,].$$

Then y can be written as a condition-dependent convolution plus (optional) bias:

$$y = x * f_{|\mathbf{c}} + b_{|\mathbf{c}},$$

with

$$f_{|\mathbf{c}} = \mathcal{F}^{-1}\big(\alpha(\mathbf{c})\,M_2(\xi)M_1(\xi)\big), \qquad b_{|\mathbf{c}} = \beta(\mathbf{c})\,\mathcal{F}^{-1}\big(M_2(\xi)\big).$$

In particular, if $\beta(\mathbf{c}) \equiv 0$, then

$$y = x * f_{|\mathbf{c}}, \qquad f_{|\mathbf{c}} = \mathcal{F}^{-1}\big(\alpha(\mathbf{c})\,M_2(\xi)M_1(\xi)\big).$$

Proof. By definition of the first spectral transformation,

$$\mathbf{h} = \mathcal{F}^{-1}\big(M_1(\xi)\,\mathcal{F}\{x\}\big).$$

Applying FiLM with spatially constant scalars $\alpha(\mathbf{c}), \beta(\mathbf{c})$ yields

$$\tilde{\mathbf{h}} = \alpha(\mathbf{c})\,\mathbf{h} + \beta(\mathbf{c}).$$

Taking the Fourier transform and using linearity,

$$\mathcal{F}\{\tilde{\mathbf{h}}\} = \alpha(\mathbf{c})\,\mathcal{F}\{\mathbf{h}\} + \beta(\mathbf{c})\,\delta(\xi) = \alpha(\mathbf{c})\,M_1(\xi)\,\mathcal{F}\{x\} + \beta(\mathbf{c})\,\delta(\xi),$$

where $\delta(\xi)$ is the Dirac delta at $\xi = 0$. Passing this through the second spectral transformation,

$$\begin{aligned} y &= \mathcal{F}^{-1}\Big(M_2(\xi)\big[\alpha(\mathbf{c})\,M_1(\xi)\,\mathcal{F}\{x\} + \beta(\mathbf{c})\,\delta(\xi)\big]\Big) \\ &= \underbrace{\mathcal{F}^{-1}\big(\alpha(\mathbf{c})\,M_2(\xi)M_1(\xi)\,\mathcal{F}\{x\}\big)}_{\text{term (A)}} + \underbrace{\mathcal{F}^{-1}\big(\beta(\mathbf{c})\,M_2(\xi)\,\delta(\xi)\big)}_{\text{term (B)}}. \end{aligned}$$

Term (A) equals a convolution by the convolution theorem:

$$\text{(A)} = x * \mathcal{F}^{-1}\big(\alpha(\mathbf{c})\,M_2(\xi)M_1(\xi)\big) = x * f_{|\mathbf{c}}.$$

Term (B) reduces to a condition-dependent bias:

$$\text{(B)} = \beta(\mathbf{c})\,\mathcal{F}^{-1}\big(M_2(\xi)\big) = b_{|\mathbf{c}}.$$

Combining the two terms gives

$$y = x * f_{|\mathbf{c}} + b_{|\mathbf{c}}.$$

If $\beta(\mathbf{c}) \equiv 0$, the bias term vanishes, yielding the pure convolution form. □