Data-Centric Engineering (2026), 7: e45
doi:10.1017/dce.2026.10064

RESEARCH ARTICLE

Hierarchical Fourier Neural Operator (HFNO): an interpretable data-driven decomposition strategy for turbulent flows

Marco Cayuela[1], Vincent Le Chenadec[2], Peter Schmid[3] and Taraneh Sayadi[4]

[1]Institut Jean le Rond d'Alembert, Sorbonne Université, France
[2]Universite Gustave Eiffel, France
[3]King Abdullah University of Science and Technology, Saudi Arabia
[4]Modélisation mathématique et numérique (M2N), Conservatoire National des Arts et Metiers, France
Corresponding author: Marco Cayuela; Email: marco.cayuela@gmail.com

Received: 12 December 2025 UTC; **Revised:** 10 April 2026 UTC; **Accepted:** 29 May 2026 UTC

Keywords: Fourier neural operator; neural operator; scale decomposition; turbulent flow; supervised learning

Abstract

Fourier Neural Operators (FNOs) have demonstrated exceptional accuracy in mapping functional spaces by leveraging Fourier transforms to establish a connection with underlying physical principles. However, their opaque inner workings often constitute an obstacle to physical interpretability. This work introduces Hierarchical Fourier Neural Operators (HFNOs), a novel FNO-based architecture tailored to the reduced-order modeling of turbulent fluid flows and designed to enhance interpretability by explicitly separating fluid behavior across scales. The proposed architecture processes wavenumber bins in parallel, enabling the approximation of dispersion relations and nonlinear interactions. Inputs are lifted to a higher-dimensional space, Fourier-transformed, and partitioned into wavenumber bins. Each bin is processed by a Fully Connected Neural Network (FCNN), with outputs subsequently padded, summed, and inverse-transformed back into physical space. A final transformation refines the output in physical space as a correction model, by means of one of the following architectures: a Convolutional Neural Network (CNN) or an Echo State Network (ESN). We evaluate the proposed model on a series of increasingly complex dynamical systems: first on the one-dimensional Kuramoto–Sivashinsky equation, then on two-dimensional Kolmogorov flow, and finally, on the prediction of the wall shear stress in turbulent channel flow, given the near-wall velocity field. In all test cases, the model demonstrates its ability to decompose turbulent flows across various scales, opening up the possibility of increased interpretability and multiscale modeling of such flows.

Impact Statement

This work introduces the Hierarchical Fourier Neural Operator (HFNO) architecture, which decomposes the flow field into multiple spectral scales that are learned and processed independently. This hierarchical treatment of scales significantly reduces model complexity while maintaining predictive accuracy in turbulent flow simulations and providing interpretability across various scales. Compared to standard Fourier Neural Operators, the HFNO achieves comparable statistical performance with five times fewer parameters.

1. Introduction

Modeling high-dimensional dynamical systems governed by partial differential equations (PDEs), such as incompressible fluid flows or reactive transport phenomena, remains a central challenge in scientific

© The Author(s), 2026. Published by Cambridge University Press. This is an Open Access article, distributed under the terms of the Creative Commons Attribution licence (http://creativecommons.org/licenses/by/4.0), which permits unrestricted re-use, distribution and reproduction, provided the original article is properly cited.

https://doi.org/10.1017/dce.2026.10064 Published online by Cambridge University Press

computing. These systems exhibit complex nonlinear dynamics and span a broad range of spatial and temporal scales. Accurately resolving all relevant scales requires extremely high resolutions in both space and time, resulting in numerical models that routinely involve billions of degrees of freedom. As a result, high-fidelity simulations—such as Direct Numerical Simulations (DNS; Moin and Mahesh, 1998) or Large Eddy Simulations (LES; Piomelli, 1999)—become prohibitively expensive, particularly at high Reynolds numbers (Pope, 2000). Even on modern high-performance computing infrastructures, such simulations often require massive parallelization and remain mostly restricted to limited time horizons or simplified geometries.

Recent advances in neural operator learning have opened new avenues for data-driven surrogate modeling of PDEs. Traditional neural network–based approaches, such as Physics-Informed Neural Networks (PINNs; Raissi et al., 2019), model the solution by taking spatial and temporal coordinates as input and enforcing the governing equations through a loss function. While effective for low-dimensional problems, these methods require retraining for each new geometry or discretization, which limits their scalability. More recently, neural operator frameworks have been introduced to learn mappings directly between infinite-dimensional function spaces rather than pointwise quantities. Among them, DeepONet (Lu et al., 2021) represents an operator using two subnetworks: a branch network sampling the input function at sensor points and a trunk network depending on the output coordinates. Although DeepONet provides theoretical guarantees for a universal operator approximation, its accuracy can be sensitive to the choice and number of sensor points, as well as to the sampling of the training functions.

In contrast, the Fourier Neural Operator (FNO; Li et al., 2021) achieves mesh and resolution invariance by parameterizing integral kernels in the Fourier domain. This formulation enables the learned operator to generalize across different discretizations of the same underlying function space, making FNOs particularly attractive for scalable, resolution-independent surrogate modeling of complex dynamical systems (Choubineh et al., 2023; Serrano et al., 2023; Wang et al., 2024). Since its introduction, the FNO framework has inspired a wide family of variants, among them the Adaptive FNO (AFNO; Guibas et al., 2022) and the Amortized FNO (AM-FNO; Xiao et al., 2024). Other classes of neural operators have also been developed, such as the Wavelet Neural Operator (WNO; Tripura and Chakraborty, 2023), which improves multiscale feature representations through localized wavelet bases, and the Graph Neural Operator (GNO; Li et al., 2020), which extends operator learning to irregular domains and unstructured meshes. Despite their strong learning capabilities, Fourier Neural Operators (FNOs) remain deep architectures characterized by limited interpretability. Although they embed physical knowledge through the use of Fourier transforms, it is still difficult to assign a clear physical meaning to the learned parameters after training.

This limitation motivates the design of a new architecture, inspired by FNOs, that retains the advantages of a spectral representation while promoting a more interpretable and scale-aware structure. Turbulent flows are inherently multiscale phenomena, characterized by a broad range of interacting spatial and temporal structures. Traditional modeling approaches in fluid mechanics, such as the Reynolds-Averaged Navier–Stokes equations (RANS; Launder and Spalding, 1974; Pope, 2000) and Large-Eddy Simulations (LES; Meneveau and Katz, 2000; Pope, 2000; Sagaut, 2006), explicitly exploit this separation of scales to design different modeling strategies for the small- and large-small dynamics—for instance, resolving the energy-containing eddies while modeling the subgrid-scale dynamics. This principle of scale separation has long proven effective in promoting both interpretability and consistency across models operating at different resolutions. Inspired by this idea, the present work aims at designing a neural architecture that can automatically identify and model these scales separately, bridging physical insight from turbulence theory with modern operator-learning techniques. The primary objective of this study is therefore to enforce a clear separation of scales across the parallel layers of the network, both at the input and output levels. This design choice is deliberate, as it enables (i) the explicit reconstruction of the contribution associated with each scale, and (ii) the selective removal of specific wavenumbers when constructing reduced-order representations.

To this end, this contribution introduces the Hierarchical Fourier Neural Operator (HFNO), a novel deep learning architecture inspired by FNOs, designed both to learn mappings between infinite-

https://doi.org/10.1017/dce.2026.10064 Published online by Cambridge University Press

dimensional function spaces and to decompose predictions across different scales. Our model divides the Fourier domain into disjoint wavenumber bins, each of which is processed by a dedicated Fully Connected Neural Network (FCNN; Hornik et al., 1989). This approach enables the model to learn scale-specific nonlinear transformations, leading to improved generalization and physical interpretability. A final correction model—implemented as either a CNN (Lecun et al., 1998; Goodfellow et al., 2016) or an Echo State Network (Jaeger, 2010)—further refines the output via residual modeling.

This article is organized as follows. Section 2 presents the general framework of operator learning, with a focus on FNOs. Section 3 introduces the HFNO architecture proposed in this study. Section 4 presents the results of our architecture compared to baseline methods when applied to several cases of variable complexity. Finally, conclusions of the work are discussed in Section 5.

2. Neural operator

In the general case of operator learning, the goal is to learn a mapping between two infinite-dimensional function spaces. Let $\mathcal{D} \subset \mathbb{R}^d$ be a bounded open set, and let $\mathcal{U} = \mathcal{F}\left(\mathcal{D}; \mathbb{R}^{d_u}\right)$ and $\mathcal{Y} = \mathcal{F}\left(\mathcal{D}; \mathbb{R}^{d_y}\right)$ be Banach spaces of functions taking values in $\mathbb{R}^{d_u}$ and $\mathbb{R}^{d_y}$, respectively. Here, D denotes the subset of the domain where the function is observed, and $\mathcal{U}$ and $\mathcal{U}$ represent the input and output function spaces, respectively. The objective is therefore to learn a nonlinear mapping $\mathcal{M}: \mathcal{U} \to \mathcal{Y}$, which may represent, for example, the solution of a PDE or the evolution of a trajectory, that we usually call an operator.

To this end, we introduce Neural Operators, which are parametric numerical models denoted by $\mathcal{M}_\theta : \mathcal{U} \to \mathcal{Y}$. Suppose we are given a dataset of observation pairs $(u_i, y_i)_{i=1}^N \in (\mathcal{U} \times \mathcal{Y})^N$, such that $y_i = \mathcal{M}(u_i)$. The goal is to find parameters θ such that $\mathcal{M}_\theta$ approximates the true operator M as accurately as possible, in the sense of minimizing a cost functional $J : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$. This setup defines a supervised learning problem.

In the present case, the function spaces involved are continuous. To enable numerical treatment, the domain D is discretized using a Cartesian mesh. Functions $u \in \mathcal{U}$ or $y \in \mathcal{Y}$ are then represented by their evaluations at a set of points $\mathcal{G} = \left(\mathbf{x}_j\right)_{j=1}^n \subset \mathcal{D}$, which sample the domain $\mathcal{D}$. For a Cartesian mesh, we typically define n_k as the number of discretization points along each dimension $k \in \{1, \ldots, d\}$, so that the total number of grid points is given by $n = \prod_k n_k$. The coordinates of the grid points in each dimension are specified by vectors $x^{(k)} \in \mathbb{R}^{n_k}$. Thus, any grid point can be identified by its multi-index $(i_1, \ldots, i_d) \in \{1, \ldots, n_1\} \times \cdots \times \{1, \ldots, n_d\}$.

In most cases, the learning process depends on the chosen discretization, and the evaluation of solutions is restricted to the discrete points $\mathbf{x}_j$, as is the case with DeepONets (Lu et al., 2021). These points are commonly referred to as sensors. As we will see later, the use of the Fourier transform helps mitigate this dependence on specific sensor locations, enabling the design of mesh-invariant solvers.

One architecture that has shown remarkable performance and impact in recent years is the Fourier Neural Operator (FNO, Li et al., 2021). This architecture is defined iteratively through a sequence of mappings $v_0 \mapsto v_1 \mapsto \ldots \mapsto v_T$, where $v_j \in \mathcal{F}\left(\mathcal{D}; \mathbb{R}^{d_v}\right)$. The input field u is first lifted to a higher-dimensional representation by a local transformation P, such that $\forall x, v_0(x) = P(u(x))$. The lifting operator P is typically implemented as a shallow, fully connected neural network (FCNN). Each transformation $v_t \mapsto v_{t+1}$, referred to as a Fourier layer, is defined as:

$$v_{t+1}(x) = \sigma(W_t v_t(x) + K_t(v_t)(x)), \tag{2.1}$$

where σ is a nonlinear activation function (typically a tanh or ReLU), $W_t \in \mathbb{R}^{d_v \times d_v}$ is a local linear transformation, and K_t is an integral operator acting on the functional space $\mathcal{F}\left(\mathcal{D}; \mathbb{R}^{d_v}\right)$, defined by

$$K_t(v_t)(x) = \mathcal{F}^{-1}(R_t \mathcal{F}(v_t))(x), \tag{2.2}$$

where $\mathcal{F}$ and $\mathcal{F}^{-1}$ denote the Fourier and inverse Fourier transforms, respectively. The tensor R_t parametrizes the operator K_t and is defined by $R_t(k) \in \mathbb{C}^{d_v \times d_v}$. To better understand Equation 2.2, we can write explicitly

https://doi.org/10.1017/dce.2026.10064 Published online by Cambridge University Press

$$[R_t\mathcal{F}(v_t)](k) = R_t(k)\mathcal{F}(v_t)(k), \tag{2.3}$$

which shows that the operation consists of a channel-wise multiplication in Fourier space. Finally, the output of the last layer v_T is projected onto the output space through another shallow FCNN Q, yielding the final output $y(x) = Q(v_T(x))$. The reader is referred to the original article (Li et al., 2021) for more information regarding the details of the procedure.

As mentioned previously, a key advantage of the FNO architecture lies in its mesh invariance: the operator acts globally over the entire spatial domain rather than locally on discrete points. All operations are either performed in physical space—independent of the specific spatial coordinates—or in Fourier space, where the low-frequency Fourier coefficients remain consistent across discretizations that satisfy the Nyquist–Shannon sampling criterion.

3. Hierarchical Fourier Neural Operator

The Hierarchical Fourier Neural Operator (HFNO) operates on a latent space that differs from that of a classical autoencoder or a variational autoencoder (Hinton and Salakhutdinov, 2006; Kingma and Welling, 2022); instead, the latent space is defined in the Fourier domain to exploit physical structure, particularly in periodic problems. This choice not only aligns with the underlying structure of periodic problems but also guarantees exact and lossless reconstruction of signals via analytically defined Fourier transform and inverse Fourier transform operations. In addition, the architecture treats the scales separately by performing a different transformation for different ranges of Fourier coefficients. The aim is to learn how to model phenomena specific to each range of scales, and to reduce the complexity of each component's task.

Beyond its spectral formulation, the HFNO remains intrinsically meshless. It operates independently of spatial discretization and is thus adaptable to complex or irregular geometries. Although its design naturally aligns with periodic problems, several extensions have demonstrated its ability to handle nonperiodic boundary conditions effectively (Li et al., 2021; Pathak et al., 2022; Kovachki et al., 2024).

The architecture proposed in this work is divided into two components: (i) the spectral block, which processes low-frequency modes grouped into blocks of submodes and predicts coarse-scale behavior, and (ii) the residual block, which processes the remaining modes in order to predict a correction term, that is, a residual closure. The second component will be discussed for different design choices. A visualization of the generic HFNO architecture is shown in Figure 1. Each component will be further described in what follows.

3.1. Spectral block

As in the case of the FNO, we first apply a lifting layer P that projects the input u into a higher-dimensional space, yielding $a \in \mathcal{F}\left(\mathcal{D};\mathbb{R}^{d_a}\right)$ with $d_a > d_u$, such that $\forall x \in \mathcal{D}, a(x) = P(u(x))$. The lifting operator P is typically implemented as a shallow Fully Connected Neural Network (FCNN) without nonlinear activation, which amounts to a linear transformation from $\mathbb{R}^{d_u}$ to $\mathbb{R}^{d_a}$.

The function $x \mapsto a(x)$ is then transformed into the frequency domain using a Fast Fourier Transform (FFT) over the spatial variable $x \in \mathcal{D}$. Using the previously introduced Cartesian grid notation, we numerically obtain $\widehat{a}$, the Fourier representation of a, defined by $\mathbf{k} \mapsto \hat{a}(\mathbf{k}) \in \mathbb{C}^{d_a}$ for $\mathbf{k} \in \mathbb{Z}^d$. The frequency modes $\mathbf{k}$ are then partitioned into blocks. We select a sequence of positive real numbers $0 = c_0 < c_1 < \ldots < c_p$, and define the l-th block of modes by

$$\mathcal{K}_l = \left\{\mathbf{k} \in \mathbb{Z}^d; c_{l-1} \le \|\mathbf{k}\| < c_l\right\}, \quad \text{for } l \in \{1,\ldots,p\}.$$

The choice of norm $\|\cdot\|$ is arbitrary, but the ℓ_2-norm is typically used in practice. Examples of partitioning strategies are shown in Figure 2. Since we use an increasing sequence of thresholds to define the blocks, each block corresponds to a range of modes associated with progressively finer scales. This construction enables a flexible separation of scales, where the thresholds c_l act as hyperparameters of the model.

https://doi.org/10.1017/dce.2026.10064 Published online by Cambridge University Press

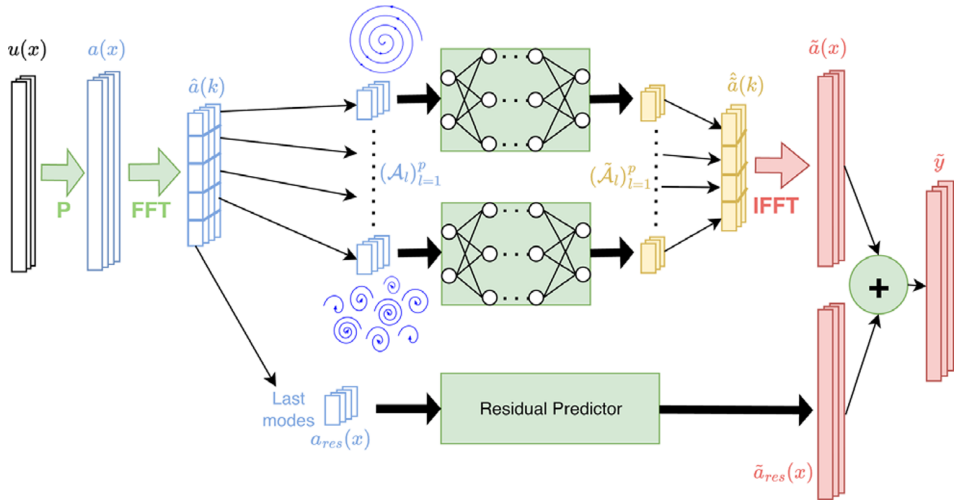


Figure 1. *Generic HFNO architecture proposed in this work.*

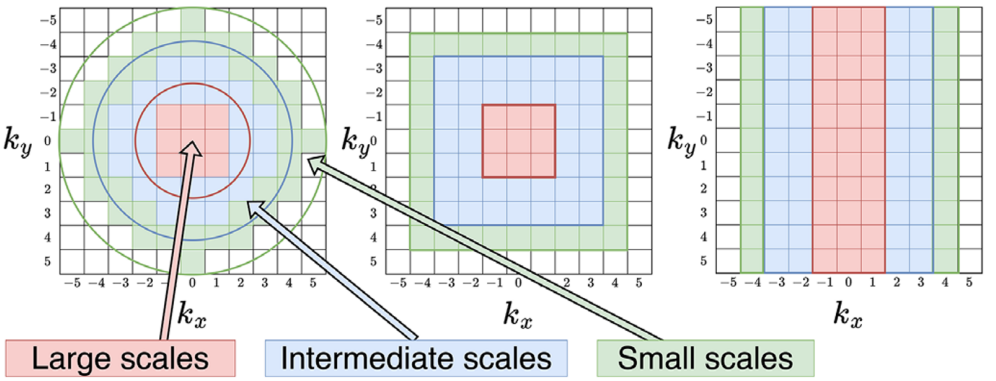


Figure 2. *Three different partitioning strategies are shown from left to right: based on the L_2-norm, the L_{inf} norm, and a semi-norm depending on a single axis.*

The Fourier-transformed function $\hat{a}$ can now be divided into bins of Fourier modes according to the previously defined partitioning of Fourier space. These are defined as $\mathcal{A}_l = \{\hat{a}(\mathbf{k}) | \mathbf{k} \in \mathcal{K}_l\}$ for $l \in \{1, \ldots, p\}$. A nonlinear transformation is then applied independently to each of these bins, within a space that preserves the same Fourier modes but maps them to a new function valued in the output space. Concretely, each bin is flattened into a C-valued vector of size $s_l = \text{card}(\mathcal{K}_l) \cdot d_a$ and is mapped to a C-valued vector of size $q_l = \text{card}(\mathcal{K}_l) \cdot d_y$ using a mapping $g_l : \mathbb{C}^{s_l} \to \mathbb{C}^{q_l}$. We choose these mappings to be fully connected neural networks (FCNNs), for which we can configure hyperparameters such as depth and width. By allowing each mapping to be tuned independently, we can assign greater representational capacity to specific bins as needed. The output of each transformation now represents a bin of Fourier modes of a new function $\tilde{a}$, following the same partitioning used previously. We define this as $\tilde{\mathcal{A}}_l = \{\hat{\tilde{a}}(\mathbf{k}) | \mathbf{k} \in \mathcal{K}_l\}$. These bins are then aggregated using the inverse of the partitioning mapping, resulting in the complete Fourier representation of $\tilde{a}$. We finally inverse-Fourier transform using an IFFT to revert into the output space to obtain $\tilde{a}$ that will be the first part of the output, representing the lowest scales of the prediction. In the following, we refer to each scale-independent transform as a "Fourier layer."

https://doi.org/10.1017/dce.2026.10064 Published online by Cambridge University Press

As mentioned previously, this architecture is particularly well suited for periodic problems. However, it is possible to extend its applicability to non-periodic signals using the padding strategy. This involves applying zero-padding to the input signal, effectively enlarging the domain by adding zeros around the original data. By doing so, the signal can be treated as approximately periodic within the extended domain, enabling the model to exploit the Fourier-based latent space while mitigating boundary artifacts.

An important distinction between HFNO and FNO lies in the placement of activation functions, which play a crucial role in deep learning architectures by enhancing their expressive power. In the case of HFNO, activation functions are applied directly in the Fourier domain. This design choice enables the model to produce outputs that are naturally separated, interpretable, and additive, as each intermediate mapping directly corresponds to the Fourier transform of the final layer output. Indeed, representing a function in the Fourier domain is equivalent to representing it in the physical domain, since the Fourier transform is a bijective mapping, and therefore, this choice should not fundamentally alter the model's learning capabilities.

3.2. Residual block

The second part of the architecture serves as a residual corrector targeting the finest scales. We explore two different approaches. The first applies corrections directly in physical space, rather than in Fourier space as done in the previous stage. The second employs a Recurrent Neural Network (RNN) in Fourier space to incorporate memory and enhance the model's ability to capture high-frequency components more effectively.

3.2.1. Prediction in physical space

The goal is to apply a transformation in physical space to account for the scales not handled by the first part of the model. To do so, we extract the Fourier modes that were not included in the previous partitioning, up to a user-defined threshold c_{max}. These modes define an additional block in the partition, given by $\mathcal{K}_{res} = \{\mathbf{k} \in \mathbb{Z}^d \mid c_p \leq \|\mathbf{k}\| < c_{max}\}$. We thus consider a truncated version of $\hat{a}$ defined as $\hat{a}_{res} : \mathcal{K}_{res} \to \mathbb{C}^{d_a}$. This representation is brought back to physical space via the inverse Fourier transform, yielding a_{res} with values in $\mathbb{R}^{d_a}$, that is, $a_{res}(\mathbf{x}) \in \mathbb{R}^{d_a}$.

The final step consists of mapping this latent-space representation to the output space. This transformation is of the form $\mathcal{W} : \mathcal{F}(\mathcal{D}; \mathbb{R}^{d_a}) \to \mathcal{F}(\mathcal{D}; \mathbb{R}^{d_y})$. We investigate two types of transformations. The first is a simple linear map $W : \mathbb{R}^{d_a} \to \mathbb{R}^{d_y}$, constant in space, such that $\forall x \in \mathcal{D}, W a_{res}(x) = \tilde{a}_{res}(x)$. Due to its limited expressiveness, we also explore a more flexible model based on a Convolutional Neural Network (CNN), which operates directly on the full field a_{res} rather than pointwise on $a_{res}(x)$.

Finally, the two contributions are summed to produce the final prediction: $\tilde{y} = \tilde{a} + \tilde{a}_{\text{res}}$. The complete process is described in Algorithm 1.

Algorithm 1 HFNO with Residual Prediction

Require: Input field $u(x)$ on domain D
Ensure: Prediction $\tilde{y}(x)$
Lifting: $a(x) \leftarrow P(u(x))$
FFT: $\hat{a}(k) \leftarrow \mathcal{F}[a(x)]$
Partition Fourier modes: define $(K_\ell)_{\ell=1}^{p}$
for $\ell = 1$ to p **do**
    Extract bin $A_\ell = \{\hat{a}(k) \mid k \in K_\ell\}$
    Flatten A_ℓ into vector
    Apply nonlinear map $\tilde{A}_\ell \leftarrow g_\ell(A_\ell)$
end for
Reconstruction in Fourier space: assemble $\hat{\tilde{a}}$ from $(\tilde{A}_\ell)$
IFFT: $\tilde{a}(x) \leftarrow \mathcal{F}^{-1}[\hat{\tilde{a}}]$
Residual modes: define K_{res}
Extract $\hat{a}_{\text{res}}(k)$ for $k \in K_{\text{res}}$

https://doi.org/10.1017/dce.2026.10064 Published online by Cambridge University Press

13: $a_{\text{res}}(x) \leftarrow \mathcal{F}^{-1}[\hat{a}_{\text{res}}]$
14: **Residual mapping:**
15: either $\tilde{a}_{\text{res}}(x) \leftarrow W(a_{\text{res}}(x))$ (linear)
16: or $\tilde{a}_{\text{res}}(x) \leftarrow \text{CNN}(a_{\text{res}}(x))$
17: **Output:** $\tilde{y}(x) \leftarrow \tilde{a}(x) + \tilde{a}_{\text{res}}(x)$

3.2.2. Prediction in wave-number space

Echo State Networks (ESNs, Jaeger, 2010) are a specific class of Recurrent Neural Networks (RNNs) belonging to the reservoir computing paradigm. They seek to simplify the training of recurrent architectures while preserving their capability to model complex temporal dynamics. Unlike traditional RNNs, where all recurrent weights are trained, ESNs employ a large, randomly connected, and fixed recurrent layer known as the reservoir. Only the output connections are learned, resulting in efficient and numerically stable training. ESNs have been widely employed in the scientific community to predict dynamical trajectories, both in full and reduced-order spaces (Heyder and Schumacher, 2021; Özalp and Magri, 2025). Their proven ability to capture complex temporal dependencies motivates their consideration for future extensions of the proposed model, particularly in the context of predicting dynamical trajectories.

An ESN is composed of three key components, illustrated in Figure 3: an **input layer**, which projects the external input signal into the high-dimensional reservoir space; a **reservoir**, that is, a sparsely connected recurrent network providing rich nonlinear dynamics; and an **output layer**, which linearly combines the reservoir states to generate the network output.

Let $\mathbf{u}(t) \in \mathbb{R}^{N_u}$ denote the input vector at time step t, $\mathbf{x}(t) \in \mathbb{R}^{N_x}$ the reservoir state vector, and $\mathbf{y}(t) \in \mathbb{R}^{N_y}$ the output vector. The ESN dynamics are governed by the following discrete-time equations:

$$\tilde{\mathbf{x}}(t+1) = f(W_{\text{in}}\mathbf{u}(t+1) + W\mathbf{x}(t) + W_{\text{fb}}\mathbf{y}(t)), \tag{3.1}$$

$$\mathbf{x}(t+1) = (1-\alpha)\mathbf{x}(t) + \alpha\tilde{\mathbf{x}}(t+1), \tag{3.2}$$

$$\mathbf{y}(t+1) = W_{\text{out}}\begin{bmatrix}\mathbf{x}(t+1)\\ \mathbf{u}(t+1)\\ 1\end{bmatrix}, \tag{3.3}$$

where

- $W_{\text{in}} \in \mathbb{R}^{N_x \times N_u}$ is the input weight matrix,
- $W \in \mathbb{R}^{N_x \times N_x}$ is the recurrent (reservoir) weight matrix,

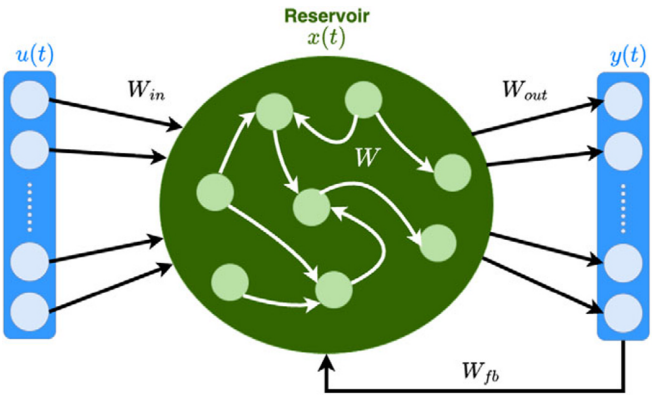


Figure 3. *Architecture of an Echo State Network (ESN).*

https://doi.org/10.1017/dce.2026.10064 Published online by Cambridge University Press

- $W_{\text{fb}} \in \mathbb{R}^{N_x \times N_y}$ is an optional feedback matrix,
- $W_{\text{out}} \in \mathbb{R}^{N_y \times (N_x + N_u + 1)}$ is the trainable output matrix,
- $f(\cdot)$ is a nonlinear activation function, typically tanh,
- α is the leaking rate.

During training, the internal weights (W_{in}, W, W_{fb}) remain fixed, while W_{out} is optimized using linear or ridge regression to minimize the prediction error. The matrices W_{in} and W are typically initialized randomly, with W_{in} that can be either dense or sparse and W sparse (Lukoševičius, 2012).

The reservoir acts as a nonlinear dynamical system that maps the input sequence into a high-dimensional state space. Its key property, the *echo state property* (ESP), ensures that the influence of past inputs on the current state vanishes over time. Formally, the ESP requires that, for any bounded input sequence $\{\mathbf{u}(t)\}$, the difference between two state trajectories $\mathbf{x}_1(t)$ and $\mathbf{x}_2(t)$, initialized differently but driven by the same input, converges to zero as $t \to \infty$. A sufficient (though not necessary) condition for the ESP is that the spectral radius $\rho_{spectral}(W)$ of the reservoir weight matrix satisfies

$$\rho_{spectral}(W) < 1, \tag{3.4}$$

which guarantees that the internal dynamics are contractive and stable. ESNs thus provide an efficient and robust framework for modeling temporal and sequential data. Their main advantages include fast training, numerical stability, and the ability to capture complex temporal dependencies through high-dimensional nonlinear reservoir dynamics. They have been successfully applied to time-series prediction, signal processing, system identification, and control tasks.

Echo State Networks (ESNs) are particularly well suited for low-dimensional data, meaning that the input and output spaces, where u and y lie, should have relatively small dimensions. This is because the expressive power of ESNs relies on the large size of the reservoir compared to the input and output spaces. Moreover, the trainable parameters are obtained through a least-squares optimization, whose computational complexity grows linearly with the output dimension. For this reason, ESNs cannot be applied directly in the physical space, which typically involves an extremely high number of degrees of freedom, often reaching billions in practical fluid dynamics problems. A reduced representation of the data is thus required. Following the methodology introduced in the previous section, the Fourier transform of the physical-space data provides a suitable reduced representation. Consequently, we use the remaining high-frequency modes—those not already employed in the Fourier layers—as the input for the ESN residual predictor.

3.3. Training process

Regarding the different nature of the two architectures, we adopt distinct training procedures for each. In both cases, the cost functional $\mathcal{J}$ is defined as the L^2-norm integrated over the domain $\mathcal{D}$. Given two fields y and $\tilde{y}$, the loss is expressed as

$$\mathcal{J}(y,\tilde{y}) = \frac{1}{|\mathcal{D}|}\|y-\tilde{y}\|_{L^2}^2 = \frac{1}{|\mathcal{D}|}\int \|y(\mathbf{x})-\tilde{y}(\mathbf{x})\|^2 d\mathbf{x} \approx \frac{1}{n}\sum_{j=1}^{n} \|y(\mathbf{x}_j)-\tilde{y}(\mathbf{x}_j)\|^2, \tag{3.5}$$

where $\|\cdot\|$ denotes the Euclidean norm in the codomain of y, and $|\mathcal{D}|$ stands for the volume of $\mathcal{D}$.

Due to the limited number of trainable parameters in the ESN and the linearity of its output layer, training can be efficiently performed using a least-squares approach, which is both faster and more accurate than the gradient-based methods commonly employed in standard deep learning models. Consequently, the residual component is trained separately using the ESN. In cases where the ESN is not used, all model components are trained jointly in an end-to-end manner.

In cases where the residual corrector is either linear or implemented as a CNN, we are given a dataset of input/output pairs $\left(u_j, y_j\right)_{j=1}^{N_{data}}$ and aim to approximate the operator M mapping u to y using a parameterized model $\mathcal{M}_\theta$, constructed as previously described. The objective is to minimize the loss

https://doi.org/10.1017/dce.2026.10064 Published online by Cambridge University Press

$\sum_j \mathcal{J}\left(y_j, \mathcal{M}_\theta(u_j)\right)$ over the parameter space of θ. The model was trained using the Adam optimizer, which belongs to the family of stochastic gradient-based optimization methods.

In the case where the residual corrector is an ESN, the training procedure differs slightly. Let $\mathcal{M}^{spec}$ denote the spectral binned component of the model, and $\mathcal{M}^{res}$ the residual correction component. We first train $\mathcal{M}^{spec}$ by minimizing the cost functional. Once trained, we use it to compute predictions $y_{pred,j} = \mathcal{M}^{spec}(u_j)$. The ESN-based corrector $\mathcal{M}^{res}$ is then trained to predict the residual $y_j - y_{pred,j}$ using the ESN training procedure.

4. Results

Results are presented for three distinct cases, each selected to reflect varying levels of complexity and application objectives. The first case involves the Kuramoto–Sivashinsky (KS) equation, providing a simplified scenario that facilitates a baseline comparison of the proposed method. In the second case, involving the Kolmogorov flow, the model is employed as an emulator to predict subsequent snapshots in a time-resolved sequence, thereby demonstrating the suitability of Echo State Networks (ESNs) for temporal forecasting tasks. The model's sole purpose is to predict the mapping from the current snapshot to the next; it is not used in an autoregressive fashion. Finally, in the turbulent channel flow case, the model is utilized to map filtered data from an off-wall plane to wall shear stress, serving as an *a priori* test for wall model implementations in large-eddy simulations.

4.1. Kuramoto–Sivashinsky equation

We first evaluate our model on the one-dimensional Kuramoto–Sivashinsky (KS) equation, a standard benchmark in nonlinear dynamics due to its ease of numerical simulation and its rich nonlinear behavior. The KS equation is formulated as follows:

$$\begin{cases} \dfrac{\partial u}{\partial t} = -\dfrac{1}{2}\nabla \cdot u^2 - \Delta u - \nu \Delta^2 u, \\ u(x+L,t) = u(x,t), \\ u(x,0) = g(x), \end{cases} \tag{4.1}$$

where L denotes the length of the one-dimensional domain, ν controls the dissipative properties of the system, and g represents the prescribed initial condition. The first equation can be reorganized as

$$\frac{Du}{Dt} = -\Delta u - \nu \Delta^2 u \tag{4.2}$$

where

$$\frac{Du}{Dt} := \frac{\partial u}{\partial t} + \frac{1}{2}\nabla \cdot u^2 \tag{4.3}$$

is the material derivative. Our objective is to predict the material derivative as a function of u. In particular, we seek to learn the operator

$$u \mapsto -\Delta u - \nu \Delta^2 u,$$

which is a linear operator parameterized by ν.

To generate the datasets, 10 numerical simulations of the Kuramoto–Sivashinsky (KS) equation are performed using different initial conditions. The first eight trajectories are used for training and the last two remaining for testing. The initial conditions are periodic signals that are represented as weighted sums of sine and cosine functions, and different simulations are generated by varying these coefficients. The spatial domain is defined by $L = 22$, and the viscosity parameter is set to $\nu = 0.7$. Each simulation produces a trajectory consisting of $N_t = 2000$ temporal snapshots on a spatial grid with $n_x = 1024$ points. The system

https://doi.org/10.1017/dce.2026.10064 Published online by Cambridge University Press

is spatio-temporally chaotic with a Lyapunov exponent $\lambda \approx 0.08$ and a Lyapunov time $t_l \approx 12.5$ computed as recommended in Edson et al. (2019). In all the figures presented in the following, we non-dimensionalize the time using t_l.

We employ an HFNO architecture composed of two parallel Fourier layers coupled with a linear residual predictor. The architectural design, particularly the number of layers and the allocation of Fourier modes, is treated as a hyperparameter optimization problem, where model capacity must be carefully balanced against performance degradation. Empirical observations indicate that, beyond an optimal depth, increasing the number of layers leads to a decline in predictive accuracy. This behavior is attributed to the progressive loss of effective intermodal interactions, which disrupts the underlying dependencies governing the flow dynamics, as shown in Appendix A.

Accordingly, a two-layer configuration is retained as it provides an effective compromise between scale separation and overall model performance. In this setting, the first 16 Fourier modes are considered and partitioned between the two layers: the first layer processes the lowest 6 modes, while the second handles the remaining 10. This choice is guided by the observed elbow in Figure 4, where a separation at mode 6 yields a favorable trade-off between minimizing error and preserving coherent groupings of modes. Each Fourier layer is parameterized by a multilayer perceptron composed of two fully connected layers with 64 neurons and GeLU activation functions, and the lifting dimension is fixed at 64. The remaining architectural details are provided in Appendix A, along with those of the FNO architecture used as a baseline for comparison.

The trained HFNO model is subsequently evaluated against a standard FNO baseline using relative root mean square error (RMSE) and relative mean absolute error (MAE) metrics, along with model size, as summarized in Table 1. The results indicate that the HFNO exhibits slightly higher prediction errors compared to the baseline, which is consistent with its deliberately shallow architecture. Nevertheless, this

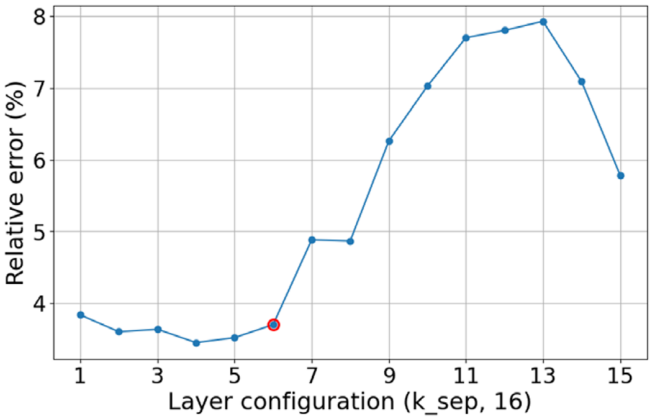


Figure 4. *Relative MSE depending on the separation mode for two-layer HFNO models. The red dot indicates the number of modes used in the selected architecture.*

Table 1. *KS equation: comparison between FNO and HFNO in terms of relative errors and number of parameters*

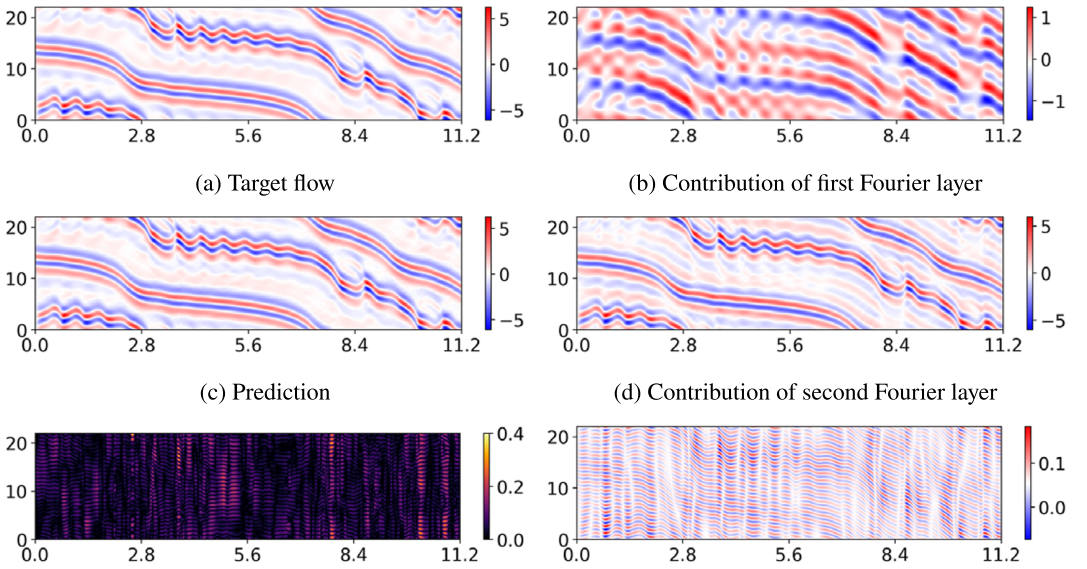
	FNO	HFNO
Rel. RMSE (%)	1.54	3.70
Rel. MAE (%)	1.20	3.61
Number of parameters	850k	36k

https://doi.org/10.1017/dce.2026.10064 Published online by Cambridge University Press

reduction in depth yields a substantial decrease in the number of parameters and, more importantly, enables a clearer interpretation of scale-dependent contributions to the flow representation. These properties provide a meaningful trade-off, where improved efficiency and interpretability offset the modest loss in predictive accuracy.

Figure 5 shows the model predictions in a spatiotemporal representation, compared to the ground truth. The corresponding error field is also reported. The last three plots show the contributions of each component of the architecture to the final prediction. As expected, the first Fourier layer predominantly captures the large-scale structures, the second layer resolves the smaller-scale features, while the residual predictor accounts for the remaining discrepancies.

In order to provide further context, Figure 6 presents the phase-space density in the $(u_x - u_{xx})$ plane, illustrating the distribution of states for both the ground truth and the predictions from different models. Each subplot corresponds to the trajectory in phase space as reconstructed by the respective model. We observe that the first Fourier layer, which represents the low-wavenumber modes, captures the densest regions of the plane. Incorporating the second layer and the residual predictor progressively refines this distribution, resulting in a phase-space representation that more closely resembles the ground truth.

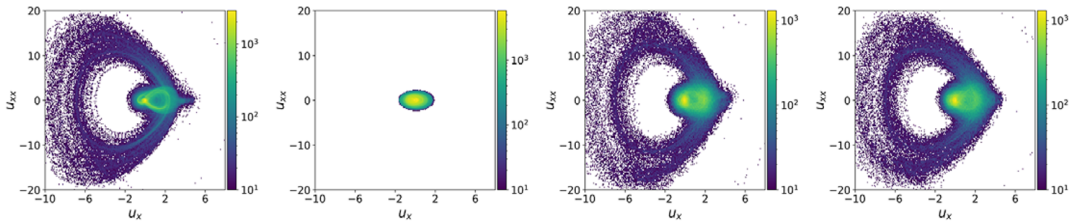


(a) Target flow (b) Contribution of first Fourier layer

(c) Prediction (d) Contribution of second Fourier layer

(e) Absolute value of total error $|\hat{u} - u|$ (f) Contribution of linear residual predictor

Figure 5. *KS equation: contour plots of* $\frac{Du}{Dt}$.



(a) Ground truth (b) First layer only (c) First two layers (d) Full model

Figure 6. *KS equation: density plot in the* (u_x-u_{xx}) *phase plane for different predictions.*

https://doi.org/10.1017/dce.2026.10064 Published online by Cambridge University Press

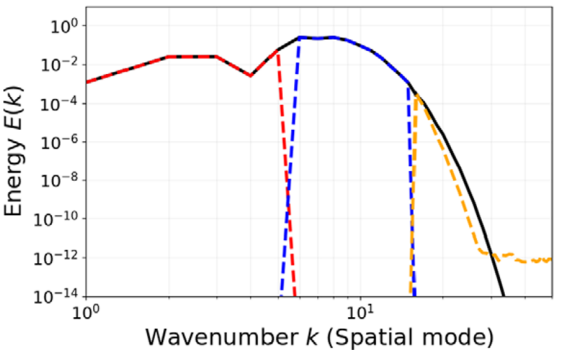


Figure 7. *KS equation: energy spectrum of the reference solution and the contributions of each component of the model. The spectra corresponding to the ground truth, the first Fourier layer, the second Fourier layer, and the linear residual predictor are displayed in black, red, blue, and orange, respectively.*

To complement these results, Figure 7 shows the energy spectrum of the Kuramoto–Sivashinsky test trajectory, together with the energy spectra predicted by each component of the model (the two Fourier layers and the linear residual predictor). As illustrated, each Fourier layer primarily reconstructs the flow features within its targeted spectral range, effectively capturing the corresponding structures at those characteristic scales. Notably, the residual predictor also contributes to these modes; although it is slightly less accurate than the Fourier layers, it successfully recovers the high-wavenumber portion of the energy spectrum, ensuring a faithful representation of the smaller-scale dynamics. These observations substantiate that the HFNO architecture achieves the intended scale separation: different layers operate on complementary regions of the energy spectrum, collectively enabling an accurate multi-scale representation of the chaotic flow dynamics.

4.2. Kolmogorov flow

In this section, we analyze the performance of our model on Kolmogorov flow. We apply it to two distinct simulations conducted at different Reynolds numbers. Kolmogorov flow describes the motion of an incompressible fluid in a two-dimensional periodic domain, driven by a sinusoidal forcing. The flow is governed by the incompressible Navier–Stokes equations:

$$\begin{cases} \nabla \cdot \mathbf{u} = 0, \\ \dfrac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{f} \end{cases} \tag{4.4}$$

where $\mathbf{u}$ denotes the velocity field, p is the pressure, ν represents the kinematic viscosity, and $\mathbf{f}$ stands for the external forcing term. In the classical Kolmogorov setup, the forcing is unidirectional and sinusoidal in space:

$$\mathbf{f}(x,y) = (F_0 \sin(kx), 0), \tag{4.5}$$

where F_0 is the forcing amplitude and $k = \frac{2\pi}{L}$ is the wavenumber corresponding to the periodicity length L of the domain in the x-direction. The domain is typically taken as a two-dimensional square periodic domain, $\Omega = [0, L]^2$, with periodic boundary conditions in both directions. This setup allows for the study of transition to turbulence, nonlinear energy transfer, and the behavior of turbulence models within a controlled setting. In our case, $L = 2\pi$ and the Reynolds number is $Re = 90$.

The overall objective of our model is to predict the velocity field at time $t + 1$ given the configuration at time t. To do so, we generate simulations of Kolmogorov flow using a pseudospectral solver based on a Fourier–Galerkin approach (Canuto et al., 1988; Magri, 2021; Kelshaw and Magri, 2024). The

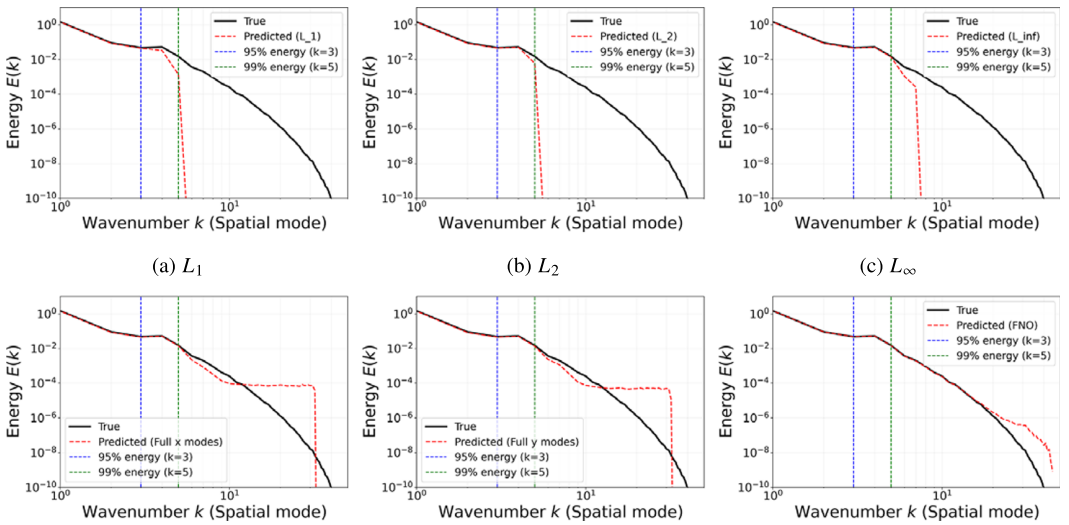
https://doi.org/10.1017/dce.2026.10064 Published online by Cambridge University Press

initialization is chosen randomly, and the simulation is run without saving snapshots during the transient phase. We then gather a sequence of snapshots $\mathbf{u}_0, \ldots, \mathbf{u}_N$, where $\mathbf{u}_0$ corresponds to the state recorded at time $t_0 = 0$ (i.e., the end of the transient phase). For all i, $\mathbf{u}_i$ denotes the snapshot recorded at time $t_i = i \times \Delta t$. The total simulation time is $T = N \times \Delta t$. All the details regarding the simulation can be found in Appendix B.

For all HFNO architectures considered in this study, we employ two parallel spectral layers without attempting to optimize this hyperparameter. The scale-separation thresholds are selected based on the energy spectrum of the reference simulations. Specifically, the first layer is designed to capture ~95% of the total energy, while the second layer extends the representation to reach 99%. The corresponding energy spectrum and cutoff values are reported in Appendix B. We have at our disposal seven simulations that are used for training and one simulation that is used to evaluate the models.

Accordingly, Fourier modes $\mathbf{k}$ such that $\|\mathbf{k}\| \in [0,3]$ are assigned to the first layer, whereas modes with $\|\mathbf{k}\| \in (4,5]$ are allocated to the second layer. As discussed in Section 3.1, several definitions of the wavenumber norm $\|\cdot\|$ can be considered to perform this spectral partitioning. We therefore conduct a comparative analysis of these different norm choices. Figure 8 presents the energy spectra obtained with different HFNO models. All models share the same layer configuration described above, while differing only in the choice of separation norm: $\|\cdot\|_{L_2}$, $\|\cdot\|_{L_1}$, and $\|\cdot\|_{L_\infty}$. In addition, semi-norms based solely on the x- and y-directions are also considered. We do not use any residual predictor for this experiment. For reference, a standard FNO model using five modes is included in the comparison. The reconstruction errors are reported in Table 2, which provides both the relative root mean square error (RMSE) and the relative mean absolute error (MAE).

We first observe that the choice of norm directly impacts the number of Fourier modes retained. In particular, the $\|\cdot\|_{L_\infty}$ norm covers a broader set of modes than $\|\cdot\|_{L_2}$, while the semi-norms selecting all modes along a given spatial direction also result in a significantly larger set of retained modes. This explains the lower reconstruction errors observed for some of these configurations. However, a closer inspection of the energy spectra reveals important differences. The models based on $\|\cdot\|_{L_1}$ and $\|\cdot\|_{L_2}$ do not accurately follow the reference spectrum up to the maximum resolved wavenumber ($k = 5$), whereas the $\|\cdot\|_{L_\infty}$-based model provides a much closer match. The semi-norm configurations are discarded in the



(a) L_1 (b) L_2 (c) L_∞

(d) Directional Mode Partitioning ($|k_y|$ Semi-Norm) (e) Directional Mode Partitioning ($|k_x|$ Semi-Norm) (f) FNO model with $k_{max} = 5$

Figure 8. *Kolmogorov flow: energy spectrum of HFNO models with different wavenumbers norm separation and a classical FNO.*

https://doi.org/10.1017/dce.2026.10064 Published online by Cambridge University Press

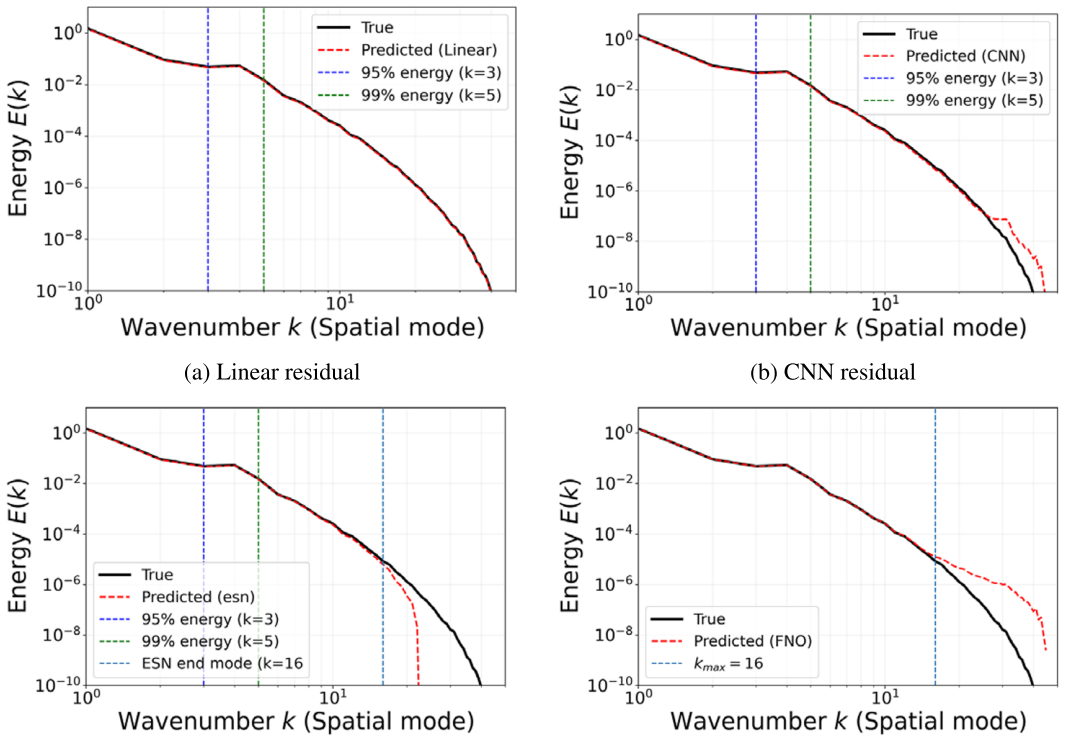
Table 2. *Kolmogorov flow: comparison between different wavenumbers norm separation and FNO in terms of relative errors*

Model	L_1	L_2	L_{inf}	Full x modes	Full y modes	FNO
Rel. RMSE (%)	16.15	11.53	9.15	8.40	8.22	2.35
Rel. MAE (%)	14.83	10.57	8.36	7.65	7.45	2.05

following analysis, as they rely on an excessively large number of modes and exhibit significant deviations at high wavenumbers.

Finally, the FNO model shows a very good agreement with the reference spectrum. Interestingly, although it is restricted to five modes, it is still able to approximate the energy distribution at higher wavenumbers, which can be attributed to the presence of skip connections in the physical space.

For the following experiments, we compare several choices for the residual predictor. As in the previous configurations, all HFNO architectures are composed of two Fourier layers, responsible respectively for the first three modes and the subsequent two modes. Models employing linear and CNN-based residual predictors are trained according to the procedure described in Section 3.2.1. In contrast, the model using an ESN-based residual predictor is trained *a posteriori*, that is, after training the HFNO model without any residual correction. For comparison, we also include results obtained with a standard FNO architecture composed of multiple stacked Fourier layers. The detailed architectures used in this comparison are provided in Appendix B. Figure 9 presents the energy spectra computed from the



(a) Linear residual

(b) CNN residual

(c) ESN residual

(d) FNO

Figure 9. *Kolmogorov flow: energy spectrum of HFNO models with different residual predictor choice and a classical FNO.*

https://doi.org/10.1017/dce.2026.10064 Published online by Cambridge University Press

Table 3. *Kolmogorov flow: comparison between different residual predictor choice and a classical FNO*

Model	Linear residual	CNN residual	ESN residual	FNO
Rel. RMSE (%)	6.75	6.83	6.52	2.10
Rel. MAE (%)	5.74	5.74	5.44	1.72
Number of parameters	203k	245k	203k	3.2M

predictions of the four models. In all plots, the ground truth is shown in black, while the predictions are displayed as red dashed curves. For the HFNO-based models, the blue and green vertical lines indicate the cutoff wavenumbers associated with the two Fourier layers. For the FNO model and the HFNO model with ESN residual, an additional light blue vertical line denotes the maximum retained mode. The linear residual predictor is observed to reconstruct the energy spectrum remarkably well. The CNN and ESN-based predictors also provide accurate reconstructions, although their agreement with the reference spectrum deteriorates slightly at higher wavenumbers. However, these observations should be interpreted in light of the results reported in Table 3. In terms of pointwise error metrics, the ESN-based model achieves slightly better performance than both the CNN and linear predictors. From a complexity standpoint, all HFNO variants yield significantly more compact representations than the FNO model, requiring more than an order of magnitude fewer parameters. It is also worth noting that the ESN is trained *a posteriori*, outside of the standard deep learning optimization pipeline, making it computationally efficient compared to the fully trained FNO model. In light of these observations, we select the HFNO model with an ESN-based residual predictor for the following analysis. Figure 10 shows the vorticity fields computed from the predictions of each component of the model, namely the two Fourier layers and the ESN residual predictor. These predictions are visually compared with the ground truth at different time

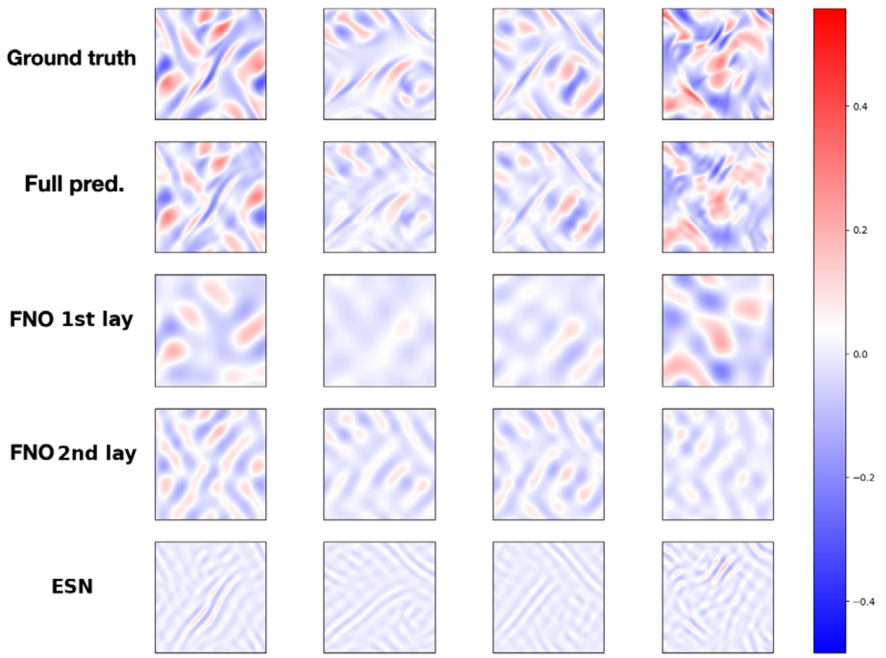


Figure 10. *Vorticity predicted by each part of the model compared to the ground truth for different snapshots; Kolmogorov flow at* $Re = 90$.

https://doi.org/10.1017/dce.2026.10064 Published online by Cambridge University Press

snapshots. As in the previous case, the results clearly illustrate how the different components of the model effectively separate and reconstruct the spatial scales of the flow.

4.3. Turbulent channel flow at $Re_\tau = 180$

We next address a more challenging problem involving turbulent channel flow, with a specific focus on wall modeling. In this context, the objective is to infer accurate wall shear stresses from filtered flow data obtained in an off-wall plane, which can then serve as a boundary condition at the wall, thereby enabling coarser near-wall resolution in simulations. Direct Numerical Simulation (DNS) of wall-bounded turbulence becomes intractable at high Reynolds numbers, as the near-wall structures require extremely fine grids. Large-Eddy Simulation (LES) reduces the computational cost by resolving only the large scales, but the near-wall region remains too demanding to capture explicitly (Choi and Moin, 2012). Wall models provide an efficient remedy by replacing the no-slip condition with a closure that accounts for the unresolved part of the boundary layer (Cabot and Moin, 2000; Bae et al., 2019). Using the HFNO architecture, our goal is to learn the underlying operator that maps off-wall filtered data to the corresponding wall shear stress. The broader motivation is that, by successfully learning this operator, we can parameterize it and generalize its application to varying flow conditions. This approach represents a significant advancement beyond traditional data-driven models, offering a more general and physically informed framework than simple input–output mappings.

To evaluate the performance of our proposed HFNO model, we first reproduce the experimental setup and results of Dupuy et al. (2023), ensuring that our implementation closely follows their architecture, training procedure, and data preprocessing. The reproduced results are consistent with those reported in the original work, providing a reliable baseline for comparison. We then compare our model's predictions to this baseline to explore improvements in accuracy and scale-specific interpretability under an *a priori* test condition. The baseline model selected for comparison is a Multilayer Perceptron (MLP), which predicts the wall shear stress τ locally. Specifically, it uses the velocity field sampled at a given distance from the wall and within a local neighborhood surrounding each target point to produce the corresponding τ prediction.

The data considered in this numerical experiment consist of snapshots of the DNS of a channel flow at a friction Reynolds number $Re_\tau = 180$. The numerical simulations were performed using the SIMSON flow software, a pseudo-spectral solver for the incompressible Navier–Stokes equations (Chevalier, 2007). The domain size is $8h\pi \times 2h \times 4h\pi$, the geometry is periodic in the streamwise and spanwise directions, and the Cartesian mesh furnishes snapshots with dimensionality $n_x \times n_y \times n_z$ where $n_x = 193$, $n_y = 65$, and $n_z = 193$.

For each snapshot, we compute the wall shear stress tensor defined as

$$\vec{\tau} = \Sigma \cdot e_n - (e_n \cdot \Sigma \cdot e_n) e_n, \tag{4.6}$$

where

$$\Sigma = \mu \left(\nabla u + (\nabla u)^T - \frac{2}{3} (\nabla \cdot u) I_d \right). \tag{4.7}$$

Our objective is to predict the norm of the wall shear stress,

$$\tau = \| \vec{\tau} \|. \tag{4.8}$$

The model is designed to predict τ using as input the velocity field evaluated at any distance Δy from the wall. To this end, from a single snapshot we generate multiple velocity fields corresponding to different distances from the wall, but restricted to the range $\Delta y^+ \in [25, 50]$, where the notation $(\cdot)^+$ refers to wall units. The conversion from physical units to wall units is given by

$$\Delta y^+ = \Delta y \sqrt{\frac{\tau}{\rho}} \frac{1}{\nu}. \tag{4.9}$$

https://doi.org/10.1017/dce.2026.10064 Published online by Cambridge University Press

Since we do not wish to explicitly provide the model with the information about the distance Δy, we instead normalize the pairs $(u(\Delta y), \tau)$ with respect to Δy, such that

$$(\tilde{u}, \tilde{\tau}) = \left(\rho \left(\frac{\Delta y}{\mu} \right) u(\Delta y),\ \rho \left(\frac{\Delta y}{\mu} \right)^2 \tau \right). \tag{4.10}$$

Figure 11 provides a schematic representation of the wall model, showing a side view of the instantaneous velocity field u, a top view of the velocity field at a distance Δy^+ from the wall used as input to the model, and the corresponding wall shear stress τ to be predicted. The wall model thus represents a mapping from the velocity field to the wall shear stress. In this setting, we employ an HFNO architecture with two parallel Fourier layers: one retaining the first eight modes, the other the subsequent six modes, while a CNN residual predictor accounts for the remaining scales. The modal ranges are selected in order to isolate flow structures associated with different streak lengths. In particular, we determine these ranges by inspecting the energy spectrum (see Appendix C) and identifying distinct levels of energy concentration across scales. We further adopt a directional partitioning of Fourier modes along the z-axis, consistent with the underlying physics of the flow. Indeed, the streaks are predominantly oriented along the x-axis, and their characteristic variations therefore occur primarily in the spanwise (z) direction. All architectural details are provided in Appendix C (Table C1 and Table C2). A classical FNO model with three sequential Fourier layers is also trained in order to compare the obtained results.

As illustrated in Figure 12, we can clearly observe that each layer captures the corresponding scales, consistent with the behavior observed in the Kuramoto–Sivashinsky and Kolmogorov flow cases. Figure 13 shows the $u^+(\Delta y^+)$ distributions obtained from different predictions of τ. To generate these curves, we convert the velocity u and the wall-normal distance Δy into wall units using the corresponding shear stress τ. The transformations are given by

$$u^+ = u\sqrt{\frac{\rho}{\tau}}, \quad \Delta y^+ = \Delta y \sqrt{\frac{\tau}{\rho}} \frac{1}{\nu}. \tag{4.11}$$

We perform this conversion using the true τ, the values predicted by the MLP baseline model, the classical FNO model, and the predictions of the HFNO model. We additionally include two intermediate configurations of the HFNO model: one using only the first Fourier layer, and another using the first two layers. We also compare all results to the Reichardt law (Reichardt, 1951), which has the following analytical expression:

$$u^+(\Delta y^+) = \frac{1}{\kappa} \log(1 + \kappa \Delta y^+) + 7.8 \left[1 - \exp\left(-\frac{\Delta y^+}{11} \right) - \frac{\Delta y^+}{11} \exp\left(-\frac{\Delta y^+}{3} \right) \right]. \tag{4.12}$$

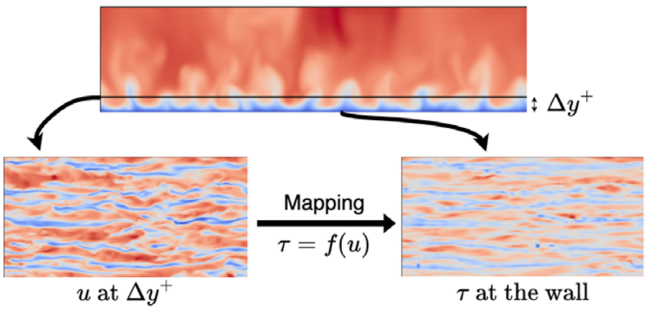


Figure 11. *Representation of the wall model (side view of the channel flow at a given snapshot; top view of the velocity field* u *off the wall and the corresponding shear stress field* τ*).*

https://doi.org/10.1017/dce.2026.10064 Published online by Cambridge University Press

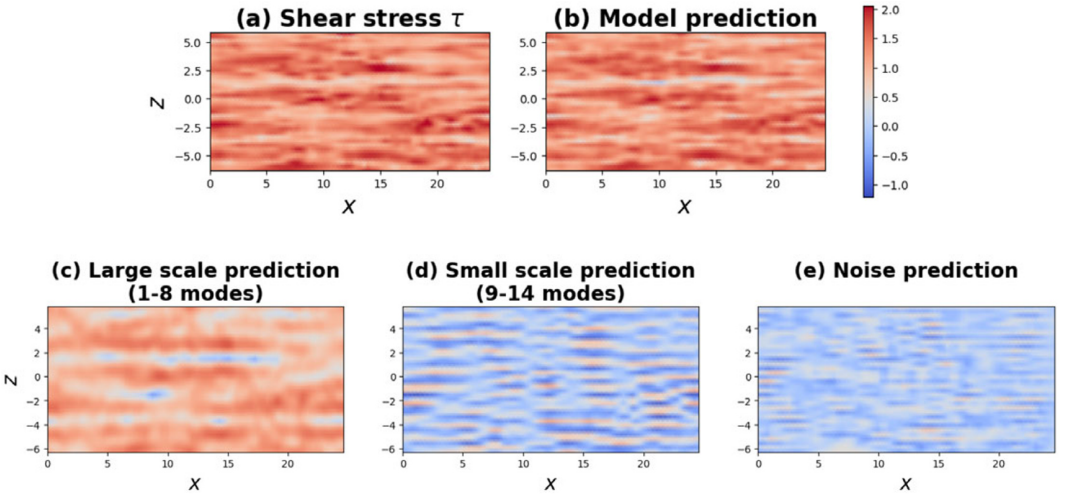
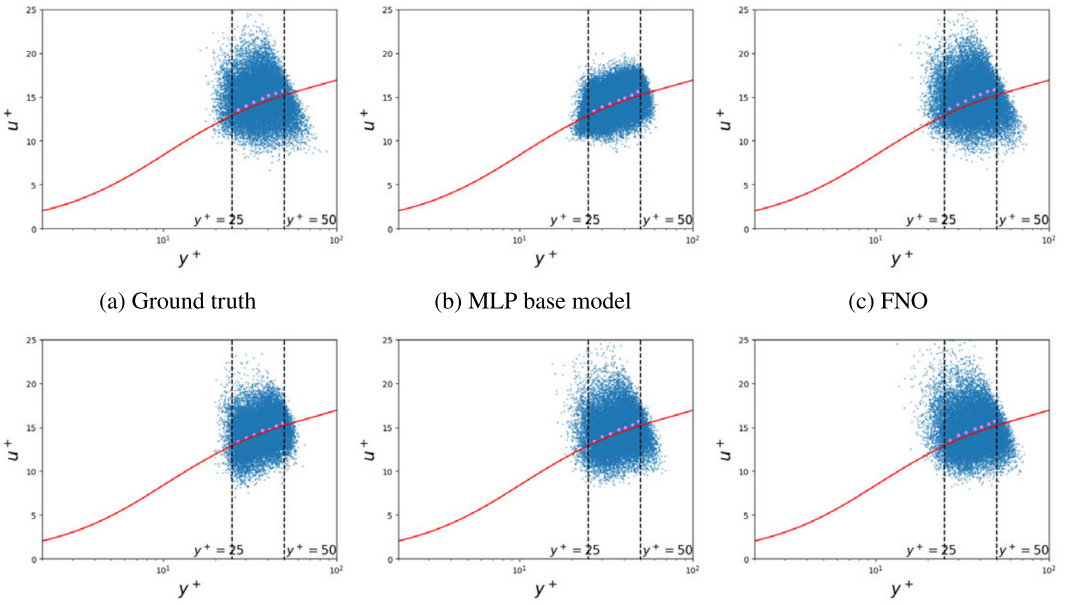


Figure 12. *A random snapshot of τ at the wall, its prediction by the HFNO model, and each part of the model.*



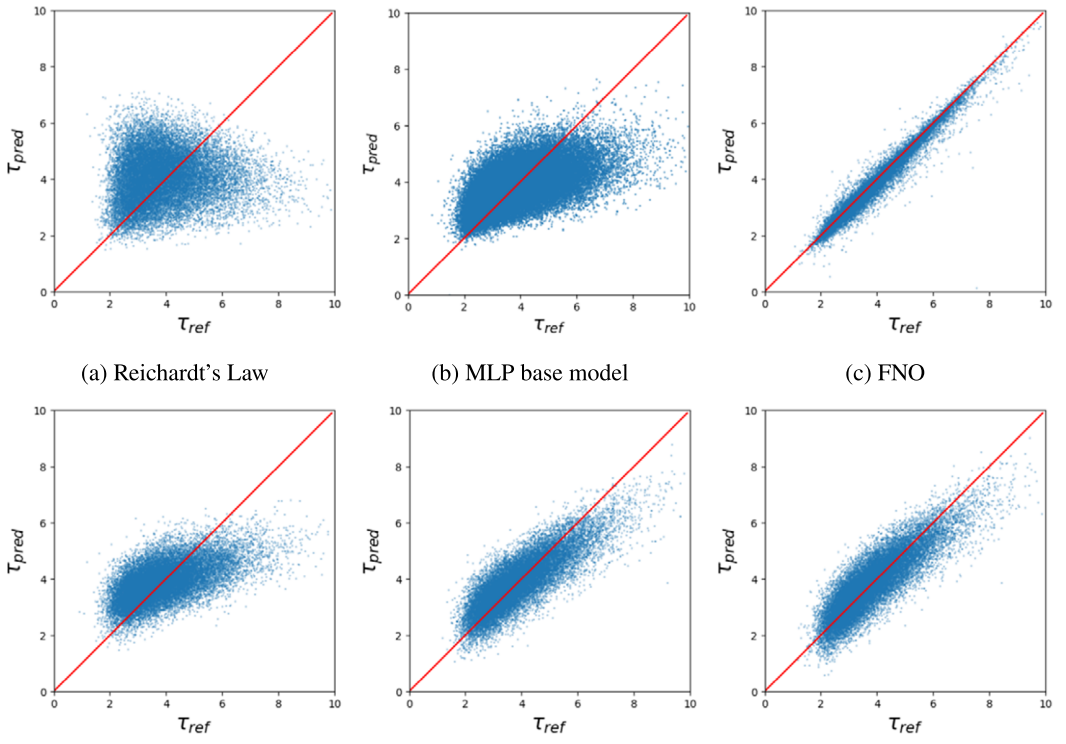
(a) Ground truth (b) MLP base model (c) FNO

(d) HFNO first layer only. (e) HFNO, two Fourier layers (f) Complete HFNO.

Figure 13. *$u^+(\Delta y^+)$ graphs for different predictions of τ; the red curve represents Reichardt's profile and the pink points the mean profile computed from the point cloud.*

with $\kappa = 0.41$ as the Von Kármán constant. For both models, the mean profiles (shown as pink points) appear to follow the Reichardt law reasonably well, even though a significant variance is observed, similar to that obtained using the true τ distribution. It is worth noting that the FNO and HFNO models exhibit a wider variance compared to the local MLP model, which allows it to capture a broader range of flow behaviors. We also note that truncated HFNO models, utilizing only the initial layers, remain competitive

https://doi.org/10.1017/dce.2026.10064 Published online by Cambridge University Press



(a) Reichardt's Law (b) MLP base model (c) FNO

(d) HFNO, first layer only. (e) HFNO, two Fourier layers. (f) Complete HFNO.

Figure 14. *Parity plots for several predictions of τ; the red curve represents $y = x$.*

with the MLP baseline. While they exhibit some loss in variance prediction relative to the full HFNO or FNO models, their mean predictions are still satisfactory.

It is also instructive to examine the parity plot of the wall shear stress τ, which compares the model predictions with the ground truth. Figure 14 presents these parity plots for our model in comparison with the local MLP baseline. Each point corresponds to the predicted value of τ at a grid location for a single snapshot, and the red line represents the reference line $y = x$. The closer the point cloud lies to this line, the more accurate the model predictions are. We again include the two intermediate configurations of the HFNO model. The results clearly show how the prediction accuracy improves as additional layers and modes are incorporated. Notably, even when using only the first set of modes, our model achieves performance comparable to that of the MLP model. As expected, the FNO achieves very high accuracy. As mentioned earlier, this can be explained in part by the loss of generality caused by the decorrelation between scales in the HFNO. However, this behavior can be partly attributed to the greater depth of the FNO compared to the HFNO, which remains relatively shallow, as well as to the intrinsic design of the HFNO, where transformations are constrained to the Fourier space. In contrast, the FNO preserves connections to the physical space at each layer. Table 4 reports several quantitative metrics that quantify and confirm the visual observations drawn from the different figures. For the MLP baseline, the classical FNO, and the HFNO together with its intermediate variants (HFNO(1) uses only the first layer, and HFNO (1 + 2) the first two layers), we compute five different metrics.

The first two metrics assess the model performance in terms of pointwise prediction accuracy, which corresponds to what the models are directly trained for. The relative root mean square error (RMSE) is defined as

https://doi.org/10.1017/dce.2026.10064 Published online by Cambridge University Press

Table 4. *Comparison of five models across five metrics (bold values indicate the best result for each metric)*

Model	RMSE (%)	r	Mean error (%)	Energy distance	Wasserstein distance
MLP base model	25.44	0.56	0.947	0.129	2.262 ± 0.117
FNO	**5.48**	**0.98**	1.205	**0.009**	**0.775** ± **0.161**
HFNO (1)	24.39	0.58	1.993	0.099	1.925 ± 0.103
HFNO (1+2)	16.71	0.83	0.947	0.024	1.086 ± 0.149
HFNO	15.10	0.86	**0.536**	0.010	0.931 ± 0.112

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{i=1}^{N} \left(\tau(\mathbf{x}_i) - \hat{\tau}(\mathbf{x}_i)\right)^2}{\sum_{i=1}^{N} \tau(\mathbf{x}_i)^2}}, \tag{4.13}$$

and the correlation coefficient r is given by

$$r = \frac{\langle \tau\hat{\tau} \rangle - \langle \tau \rangle \langle \hat{\tau} \rangle}{\sqrt{\langle \tau^2 \rangle - \langle \tau \rangle^2} \sqrt{\langle \hat{\tau}^2 \rangle - \langle \hat{\tau} \rangle^2}}, \tag{4.14}$$

where $\langle \cdot \rangle$ denotes the spatial mean over the domain.

The remaining three metrics quantify differences in the distributions of points in the $u^+(y^+)$ diagrams shown in Figure 13. For each model, we obtain a distribution in $\mathbb{R}^2$ that we compare with the ground truth. The mean error corresponds to the RMSE between the mean profiles; the energy distance (Rizzo and Székely, 2016) measures the discrepancy between two probability distributions μ and ν, and is defined as

$$E(X, Y) = 2\mathbb{E}\|X - Y\| - \mathbb{E}\|X - X'\| - \mathbb{E}\|Y - Y'\|, \tag{4.15}$$

where X and X' are i.i.d. random variables following μ, and Y and Y' are i.i.d. random variables following ν. Finally, the Wasserstein distance (Villani, 2009; Peyré and Cuturi, 2020) quantifies the optimal transport cost required to transform one distribution into the other.

These results indicate that, although the FNO remains the most accurate model in terms of pointwise prediction, the HFNO achieves statistically comparable performance. On average, the HFNO yields slightly better results than the FNO, which appears to exhibit a higher mean error, while both models perform similarly in terms of distributional metrics. Moreover, these results were obtained with a significantly more compact architecture: the trained FNO contains ~5 million parameters, compared to only 1 million for the HFNO. Interestingly, this number can be further reduced by considering truncated versions of the HFNO (HFNO(1) and HFNO(1+2) in Table 4), which remain competitive and satisfactory depending on the desired level of accuracy.

5. Conclusion

In this work, we have presented a Hierarchical Fourier Neural Operator (HFNO) architecture tailored to dynamical systems with multiscale spatial structures. By decomposing the Fourier modes into separate layers corresponding to physically meaningful ranges—energy-containing, inertial, and dissipative scales—the model is able to capture the dominant dynamics at each scale. The use of MLPs for each Fourier layer, complemented by a residual predictor that can take on different forms, allows the network to faithfully reconstruct both large-scale coherent structures and small-scale fluctuations.

Applications to the Kuramoto–Sivashinsky equation, the Kolmogorov flow, and wall modeling in a turbulent channel flow demonstrate the effectiveness of the proposed approach: each Fourier layer predominantly contributes to its assigned wavenumber range, leading to accurate predictions and satisfying generalization capabilities. The hierarchical design not only improves interpretability by linking model components to physical scales but also enhances efficiency by reducing the complexity of each subtask.

https://doi.org/10.1017/dce.2026.10064 Published online by Cambridge University Press

Overall, these results highlight the potential of HFNOs for modeling complex turbulent flows in a meshless and physics-informed manner. Future work will focus on extending this framework to higher Reynolds numbers and more realistic three-dimensional flows, as well as exploring adaptive strategies for mode separation to further optimize performance across a broader range of dynamical regimes.

Data availability statement. Replication data and code can be found at https://github.com/marcocayuela/HFNO.

Acknowledgments. The authors gratefully acknowledge Ardeshir Hanifi, Ricardo Vinuesa, and Arivazhagan Geetha Balasubramanian from the Division of Fluid Mechanics, KTH Royal Institute of Technology (Sweden), for kindly providing the open-channel dataset used in this study. The authors also acknowledge the use of the generative AI tool ChatGPT (OpenAI, GPT-5.1, accessed in October 2025) to assist with correcting typographical and grammatical errors in the manuscript. All suggestions generated by the tool were reviewed and validated by the authors.

Author contributions. **M.C.**: Conceptualization, data curation, data visualization, formal analysis, methodology, validation, and writing original draft. **V.L.C.**: Resources, supervision, validation, and writing—review and editing. **P.J.S.**: Resources, supervision, validation, and writing—review and editing. **T.S.**: Conceptualization, formal analysis, methodology, resources, supervision, validation, and writing—review and editing. All authors approved the final submitted draft.

Funding statement. This research was supported by a grant from the Agence Nationale de la Recherche (ANR), under the funding ID: ANR-22-CE94-0001; EDNNForest.

Competing interests. The authors declare none.

Ethical standard. The research meets all ethical guidelines, including adherence to the legal requirements of the study country.

References

Bae HJ, **Lozano-Durán A**, **Bose ST and Moin P** (2019) Dynamic slip wall model for large-eddy simulation. *Journal of Fluid Mechanics 859*, 400–432. https://doi.org/10.1017/jfm.2018.838.

Cabot W and Moin P (2000) Approximate Wall boundary conditions in the large-Eddy simulation of high Reynolds number flow. En. *Flow, Turbulence and Combustion 63*(1), 269–291. https://doi.org/10.1023/A:1009958917113.

Canuto C, **Hussaini MY**, **Quarteroni A and Zang TA** (1988) *Spectral Methods in Fluid Dynamics*. Berlin, Heidelberg: Springer.

Chevalier M (2007) *SIMSON: A Pseudo-Spectral Solver for Incompressible Boundary Layer Flows*. Stockholm: Mekanik, Kungliga Tekniska högskolan.

Choi H and Moin P (2012) Grid-point requirements for large eddy simulation: Chapman's estimates revisited. *Physics of Fluids 24*(1), 011702. https://doi.org/10.1063/1.3676783.

Choubineh A, **Chen J**, **Wood DA**, **Coenen F and Ma F** (2023) Fourier neural operator for fluid flow in small-shape 2D simulated porous media dataset *Algorithms 16*(1), https://doi.org/10.3390/a16010024.

Dupuy D, **Odier N and Lapeyre C** (2023) Data-driven wall modeling for turbulent separated flows. *Journal of Computational Physics 487*(1), 112173. https://doi.org/10.1016/j.jcp.2023.112173.

Edson RA, **Bunder JE**, **Mattner TW and Roberts AJ** (2019) *Lyapunov Exponents of the Kuramoto-Sivashinsky PDE.* Available at http://arxiv.org/abs/1902.09651 (accessed 20 March 2026).

Goodfellow I, **Bengio Y and Courville A** (2016) *Deep Learning*, Cambridge, MA; MIT Press; Available at http://www.deeplearningbook.org.

Guibas J, **Mardani M**, **Li Z**, **Tao A**, **Anandkumar A and Catanzaro B** (2022) *Adaptive Fourier Neural Operators: Efficient Token Mixers for Transformers.* Available at http://arxiv.org/abs/2111.13587 (accessed 19 March 2026).

Heyder F and Schumacher J (2021) Echo state network for two-dimensional turbulent moist Rayleigh-Bénard convection. *Physical Review E 103*(5), 053107. https://doi.org/10.1103/PhysRevE.103.053107; Available at http://arxiv.org/abs/2101.11325 (accessed 22 October 2025).

Hinton GE and Salakhutdinov RR (2006) Reducing the dimensionality of data with neural networks. *Science 313* (5786), 504–507. https://doi.org/10.1126/science.1127647.

Hornik K, **Stinchcombe M and White H** (1989) Multilayer feedforward networks are universal approximators. *Neural Networks 2*(5), 359–366. https://doi.org/10.1016/0893-6080(89)90020-8; Available at https://www.sciencedirect.com/science/article/pii/0893608089900208 (accessed 24 October 2025).

Jaeger H (2010) *The "echo state" Approach to Analysing and Training Recurrent Neural Networks – With an Erratum Note.*

Kelshaw D and Magri L (2024) Physics-constrained convolutional neural networks for inverse problems in spatiotemporal partial differential equations. En. *Data-Centric Engineering 2*, e43. https://doi.org/10.1017/dce.2024.46; Available at https://www.cambridge.org/core/journals/data-centric-engineering/article/physicsconstrained-convolutional-neural-networks-for-inverse-problems-in-spatiotemporal-partial-differential-equations/22C1675C179C3D8352BBC119F4196B7B (accessed 2 April 2026).

Kingma DP and Welling M (2022) *Auto-Encoding Variational Bayes.* Available at http://arxiv.org/abs/1312.6114 (accessed 20 October 2025).

Kovachki N, **Li Z**, **Liu B**, **Azizzadenesheli K**, **Bhattacharya K**, **Stuart A and Anandkumar A** (2024) *Neural Operator: Learning Maps Between Function Spaces.* https://doi.org/10.5555/3648699.3648788; Available at http://arxiv.org/abs/2108.08481 (accessed 22 October 2025).

Launder BE and Spalding DB (1974) The numerical computation of turbulent flows. *Computer Methods in Applied Mechanics and Engineering 3*(2), 269–289. https://doi.org/10.1016/0045-7825(74)90029-2; Available at https://www.sciencedirect.com/science/article/pii/0045782574900292 (accessed 24 October 2025).

Lecun Y, **Bottou L**, **Bengio Y and Haffner P** (1998) Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324. https://doi.org/10.1109/5.726791; Available at http://ieeexplore.ieee.org/document/726791/ (accessed 3 July 2025).

Li Z, **Kovachki N**, **Azizzadenesheli K**, **Liu B**, **Bhattacharya K**, **Stuart A and Anandkumar A** (2020) *Neural Operator: Graph Kernel Network for Partial Differential Equations.* Available at http://arxiv.org/abs/2003.03485 (accessed 24 October 2025).

Li Z, **Kovachki N**, **Azizzadenesheli K**, **Liu B**, **Bhattacharya K**, **Stuart A and Anandkumar A** (2021) *Fourier Neural Operator for Parametric Partial Differential Equations.* Available at http://arxiv.org/abs/2010.08895 (accessed 5 December 2023).

Lu L, **Jin P and Karniadakis GE** (2021) Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence 3*(3), 218–229. https://doi.org/10.1038/s42256-021-00302-5; Available at http://arxiv.org/abs/1910.03193 (accessed 7 October 2025).

Lukoševičius M (2012) A practical guide to applying Echo state networks. En. In Montavon G, Orr GB and Müller KR (eds), *Neural Networks: Tricks of the Trade*. vol *770*. Berlin, Heidelberg: Springer Berlin Heidelberg, 659–686. https://doi.org/10.1007/978-3-642-35289-8_36; Available at http://link.springer.com/10.1007/978-3-642-35289-8_36 (accessed 3 July 2025).

Magri L (2021) *GitHub - MagriLab/KolSol: Pseudospectral Kolmogorov Flow Solver.* Available at https://github.com/MagriLab/KolSol?tab=readme-ov-file#R1 (accessed 23 July 2025).

Meneveau C and Katz J (2000) Scale-invariance and turbulence models for large-Eddy simulation. *Annual Review of Fluid Mechanics 32*, 1–32. https://doi.org/10.1146/annurev.fluid.32.1.1; Available at https://www.annualreviews.org/content/journals/10.1146/annurev.fluid.32.1.1 (accessed 24 October 2025).

Moin P and Mahesh K (1998) DIRECT NUMERICAL SIMULATION: A tool in turbulence research. Fr. *Annual Review of Fluid Mechanics 30*, 539–578. https://doi.org/10.1146/annurev.fluid.30.1.539; Available at https://www.annualreviews.org/content/journals/10.1146/annurev.fluid.30.1.539 (accessed 3 July 2025).

Özalp E and Magri L (2025) Stability analysis of chaotic systems in latent spaces. En. *Nonlinear Dynamics 113*(11), 13791–13806. https://doi.org/10.1007/s11071-024-10712-w.

Pathak J, **Subramanian S**, **Harrington P**, **Raja S**, **Chattopadhyay A**, **Mardani M**, **Kurth T**, **Hall D**, **Li Z**, **Azizzadenesheli K**, **Hassanzadeh P**, **Kashinath K and Anandkumar A** (2022) *FourCastNet: A Global Data-driven High-resolution Weather Model using Adaptive Fourier Neural Operators.* Available at http://arxiv.org/abs/2202.11214 (accessed 22 October 2025).

Peyré G and Cuturi M (2020) *Computational Optimal Transport.* Available at http://arxiv.org/abs/1803.00567 (accessed 31 October 2025).

Piomelli U (1999) Large-eddy simulation: Achievements and challenges. *Progress in Aerospace Sciences 35*(4), 335–362. https://doi.org/10.1016/S0376-0421(98)00014-1; Available at https://www.sciencedirect.com/science/article/pii/S0376042198000141 (accessed 20 October 2025).

Pope SB (2000) *Turbulent Flows*. Cambridge: Cambridge University Press. Available at https://www.cambridge.org/highereducation/books/turbulent-flows/C58EFF59AF9B81AE6CFAC9ED16486B3A (accessed 3 July 2025).

Raissi M, **Perdikaris P and Karniadakis GE** (2019) 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. https://doi.org/10.1016/j.jcp.2018.10.045; Available at https://www.sciencedirect.com/science/article/pii/S0021999118307125 (accessed 8 October 2025).

Reichardt H (1951) Vollständige Darstellung der turbulenten Geschwindigkeitsverteilung in glatten Leitungen. de. *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik 31*(7), 208–219. https://doi.org/10.1002/zamm.19510310704; Available at https://onlinelibrary.wiley.com/doi/abs/10.1002/zamm.19510310704 (accessed 23 October 2025).

Rizzo ML and Székely GJ (2016) Energy distance. En. *WIREs Computational Statistics 8*(1), 27–38. https://doi.org/10.1002/wics.1375; Available at https://wires.onlinelibrary.wiley.com/doi/10.1002/wics.1375 (accessed 31 October 2025).

Sagaut P (2006) *Large Eddy Simulation for Incompressible Flows. Scientific Computation*. Berlin/Heidelberg: Springer-Verlag. Available at http://link.springer.com/10.1007/b137536 (accessed 24 October 2025).

Serrano L, **Migus L**, **Yin Y**, **Mazari JA and Gallinari P** (2023) *INFINITY:Neural Field Modeling for Reynolds-AveragedNavier-Stokes Equations.* Available at http://arxiv.org/abs/2307.13538 (accessed 24 October 2025).

Tripura T and Chakraborty S (2023) Wavelet neural operator for solving parametric partial differential equations in computational mechanics problems. *Computer Methods in Applied Mechanics and Engineering 404* 404, 115783. https://doi.org/10.1016/j.cma.2022.115783; Available at https://www.sciencedirect.com/science/article/pii/S0045782522007393 (accessed 24 October 2025).

Villani C (2009) Optimal Transport. In Berger M, Eckmann B, De La Harpe P, Hirzebruch F, Hitchin N, Hörmander L, Kupiainen A, Lebeau G, Ratner M, Serre D, Sinai YG, Sloane NJA, Vershik AM and Waldschmidt M (eds). *Grundlehren der mathematischen Wissenschaften*. Berlin, Heidelberg: Springer.

https://doi.org/10.1017/dce.2026.10064 Published online by Cambridge University Press

Wang Y, **Li Z**, **Yuan Z**, **Peng W**, **Liu T and Wang J** (2024) Prediction of turbulent channel flow using Fourier neural operator-based machine-learning strategy. *Physical Review Fluids 9* (8). https://doi.org/10.1103/PhysRevFluids.9.084604.

Xiao Z, **Kou S**, **Hao Z**, **Lin B and Deng Z** (2024) *Amortized Fourier Neural Operators.* Available at https://openreview.net/forum?id=a6em980M9x (accessed 19 March 2026).

Appendix A. Kuramoto–Sivashinsky

In this case, the spatial domain is defined by $L = 22$, and the viscosity parameter is set to $v = 0.7$. Each simulation produces a trajectory consisting of $N_t = 2000$ temporal snapshots on a spatial grid with $n_x = 1024$ points. The simulation time is $T = 150$ and $dt = 0.075$. For training, we discard the first 150 points that correspond to the transient phase.

In order to select an appropriate number of parallel layers, we trained several HFNO models with different layer configurations. The total number of modes was kept fixed, and the architectures were designed with 1, 2, 4, 8, and 16 layers, yielding the following configurations: [16], [8,16], [4,8,12,16]. … Figure A1 reports the relative RMSE on the test data for each of these models. We select the model corresponding to the knee of the curve, as it provides a good trade-off between overall reconstruction error and effective mode separation.

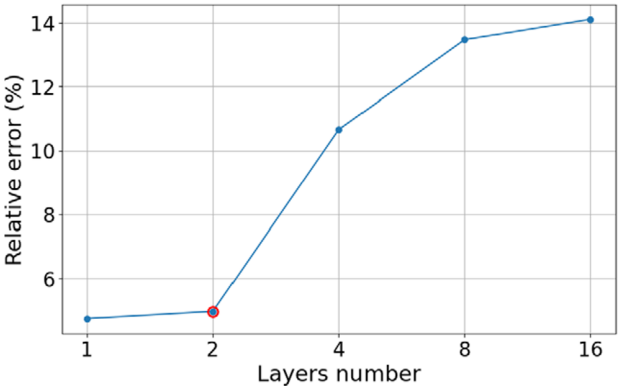


Figure A1. *Relative MSE depending on the number of layers. The red dot indicates the number of layers used in the selected architecture.*

A.1. Hyperparameters of the HFNO and FNO models

Tables A1 and A2 report the hyperparameterizations of the FNO used as baseline and the final selected HFNO trained for the KS equation case.

Table A1. *FNO*

Parameters	Value
Number of Fourier layers	3
Number of modes (k)	16
Lifted dimension (d_v)	128
Input dimension (d_{in})	1
Output dimension (d_{out})	1
Lifting layer type	Shallow MLP
Projection type	Shallow MLP
Activation function	GeLU

https://doi.org/10.1017/dce.2026.10064 Published online by Cambridge University Press

Table A2. *HFNO*

Parameters	Value
Number of parallel layers	2
Repartition of modes (k)	1–6 & 7–16
Lifted dimension (d_{model})	32
Input dimension (d_{in})	1
Output dimension (d_{out})	1
Lifting layer type	Shallow MLP
MLP characteristics	layers: 2; width: 32
Activation function	GeLU

Appendix B. Kolmogorov flow

In this case, the spatial domain is defined as a square $[0,2\pi]\times[0,2\pi]$. The Reynolds number is set to $Re=90$, and the forcing frequency is $n_f=4$. The dynamical system is solved using a pseudospectral solver based on a Fourier–Galerkin approach (Canuto et al., 1988; Magri, 2021; Kelshaw and Magri, 2024). The computational grid is a Cartesian grid of size $n_x\times n_y$, with $n_x=n_y=128$. Each simulation is initialized with a random velocity field, followed by an initial transient period of 180 time units, after which the subsequent 120 time units are recorded. The simulation time step is set to $\Delta t=0.001$, and the number of retained Fourier modes is $n_k=32$. The resulting trajectories are then preprocessed. They are downsampled in the physical space by a factor of 2 in both spatial directions, and in time by a factor of 60, yielding an effective time step of $\Delta t=0.06$ for the data used by the models.

B.1. Hyperparameters and architecture of the HFNO and FNO models

In this section, we summarize the architectures used for our experiments. All HFNO models share the same base structure consisting of multiple Fourier layers that are responsible for capturing specific spectral modes. The HFNO variants differ only in the

Table B1. *FNO: classical Fourier Neural Operator architecture*

Parameters	Value
Number of Fourier layers	3
Number of modes (k)	16
Lifted (d_v)	64
Input dimension (d_{in})	2
Output dimension (d_{out})	2
Lifting layer type	Shallow MLP
Projection type	Shallow MLP
Activation function	GeLU

Table B2. *HFNO: base architecture shared by all residual predictor variants*

Parameters	Value
Number of parallel Fourier layers	2
Repartition of modes (k)	1–3 & 4–5
Separation norm	L_∞
Lifted dimension (d_{model})	8
Input dimension (d_{in})	2
Output dimension (d_{out})	2
Lifting layer type	Shallow MLP
MLP characteristics	lays.: 1; width: 128
Activation function	GeLU

https://doi.org/10.1017/dce.2026.10064 Published online by Cambridge University Press

type of residual predictor employed: Linear, CNN, or ESN. The FNO model is composed of stacked Fourier layers without any residual predictor. Tables B1 and B2 below provide a concise description of each component and its parameters.

B.2. Residual predictors of the Kolmogorov flow

The HFNO variants use different residual predictors. Below are the parameters of each residual predictor.

- **Linear residual predictor:** A simple linear mapping applied to the HFNO output. Key parameters:
  - Input dimension: d_v (the lifted dimension)
  - Output dimension: d_{out} (matches HFNO output)
  - Training: end-to-end with HFNO base
- **CNN residual predictor:** A shallow convolutional network applied to the HFNO output.
  - 5 convolutional layers, all with kernel size 3 and padding "same"
  - Channels per layer: [input → 32 → 64 → 32 → 8 → output]
  - Activation: default PyTorch (or specify if applied)
  - Input/Output: matches HFNO output dimensions
  - Training: end-to-end with HFNO base
- **ESN residual predictor:** An Echo State Network applied to the HFNO output to capture temporal dependencies in the residuals. Key parameters:
  - Reservoir size: 10 k
  - Spectral radius: 0.9
  - Input dimension: 2054 (size of the mask used to cut modes × input dimension × 2)
  - Output dimension: 2054
  - Training: *a posteriori*, after the HFNO base has been trained

Appendix C. Channel flow

Figure C1 shows the ranges used to select the wavenumber of each Fourier layer. The first line corresponds to the cutoff of the first layer and the second to the second Fourier layer.

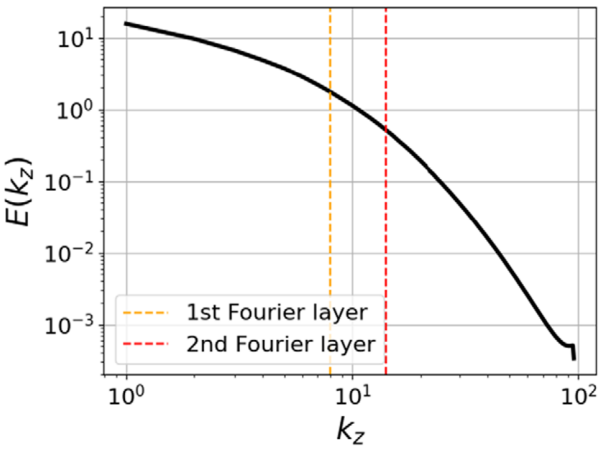


Figure C1. *Energy spectrum of the channel flow data. The colored dashed vertical lines indicate, from left to right, the 80% and 95%.*

https://doi.org/10.1017/dce.2026.10064 Published online by Cambridge University Press

C.1. Hyperparameters and architecture of the HFNO and FNO models

Table C1. *HFNO architecture details*

Parameter	Value
Number of parallel Fourier layers	2
Mode allocation	1–8 & 9–14
Separation norm	Directional Mode Partitioning along z axis
Lifted dimension (d_{model})	8
Input dimension (d_{in})	2
Output dimension (d_{out})	2
Lifting layer type	Shallow MLP
MLP characteristics	layers: 2; width: 128
Activation function	GeLU
Residual predictor	*CNN* (same as for Kolmogorov part)

Table C2. *FNO architecture details*

Parameter	Value
Number of fourier layers	3
Number of modes ($k_x \times k_y$)	24×24
Lifted dimension (d_v)	128
Input dimension (d_{in})	2
Output dimension (d_{out})	2
Lifting layer type	Shallow MLP
Projection layer type	Shallow MLP
Activation function	GeLU

Cite this article: Cayuela M, Le Chenadec V, Schmid P and Sayadi T (2026). Hierarchical Fourier Neural Operator (HFNO): an interpretable data-driven decomposition strategy for turbulent flows. *Data-Centric Engineering*, 7, e45. doi:10.1017/dce.2026.10064

https://doi.org/10.1017/dce.2026.10064 Published online by Cambridge University Press