Computer Methods in Applied Mechanics and Engineering 433 (2025) 117478

Contents lists available at ScienceDirect

Comput. Methods Appl. Mech. Engrg.

journal homepage: www.elsevier.com/locate/cma

Neural differentiable modeling with diffusion-based super-resolution for two-dimensional spatiotemporal turbulence

Xiantao Fan [a], Deepak Akhare [a], Jian-Xun Wang [a,b,*]

[a] Department of Aerospace and Mechanical Engineering, University of Notre Dame, Notre Dame, IN, United States of America
[b] Sibley School of Mechanical and Aerospace Engineering, Cornell University, Ithaca, NY, United States of America

ARTICLE INFO

Keywords:
Differentiable programming
Super-resolution
Scientific machine learning
Deep neural network
Conditional diffusion model
Generative AI

ABSTRACT

Simulating spatiotemporal turbulence with high fidelity remains a cornerstone challenge in computational fluid dynamics (CFD) due to its intricate multiscale nature and prohibitive computational demands. Traditional approaches typically employ closure models, which attempt to represent small-scale features in an unresolved manner. However, these methods often sacrifice accuracy and lose high-frequency/wavenumber information, especially in scenarios involving complex flow physics. In this paper, we introduce an innovative neural differentiable modeling framework designed to enhance the predictability and efficiency of spatiotemporal turbulence simulations. Our approach features differentiable hybrid modeling techniques that seamlessly integrate deep neural networks with numerical PDE solvers within a differentiable programming framework, synergizing deep learning with physics-based CFD modeling. Specifically, a hybrid differentiable neural solver is constructed on a coarser grid to capture large-scale turbulent phenomena, followed by the application of a Bayesian conditional diffusion model that generates small-scale turbulence conditioned on large-scale flow predictions. Two innovative hybrid architecture designs are studied, and their performance is evaluated through comparative analysis against conventional large eddy simulation techniques with physics-based subgrid-scale closures and purely data-driven neural solvers. The findings underscore the potential of the neural differentiable modeling framework to significantly enhance the accuracy and computational efficiency of turbulence simulations. This study not only demonstrates the efficacy of merging deep learning with physics-based numerical solvers but also sets a new precedent for advanced CFD modeling techniques, highlighting the transformative impact of differentiable programming in scientific computing.

1. Introduction

The simulation of turbulent fluid flows is crucial for advancing our understanding and enhancing predictive capabilities across a wide range of applications, from aerospace engineering to environmental sciences. Despite significant advancements in scientific computing and numerical modeling, achieving an effective and efficient predictive simulation of turbulence remains a substantial challenge, primarily arising from the inherent multi-scale nature of turbulence and the significant computational demands required for accurately capturing these scales. The direct numerical simulation (DNS), which resolves all scales of turbulence structures, is only feasible for very limited simple flow configurations. For more complex scenarios, especially those involving high Reynolds numbers, the computational costs for DNS exceed practical limits, prompting the need for alternative approaches. Methods such

* Corresponding author at: Sibley School of Mechanical and Aerospace Engineering, Cornell University, Ithaca, NY, United States of America.
E-mail addresses: jwang33@nd.edu, jw2837@cornell.edu (J.-X. Wang).

https://doi.org/10.1016/j.cma.2024.117478
Received 28 June 2024; Received in revised form 7 September 2024; Accepted 15 October 2024
Available online 25 October 2024

as Reynolds-Averaged Navier–Stokes (RANS) models and Large Eddy Simulations (LES) are viable solutions by approximating the effects of unresolved turbulence through closure modeling, effectively balancing computational demands with simulation accuracy [1]. Nonetheless, these strategies, especially LES, continue to present substantial computational challenges for practical engineering applications. Moreover, the generalizability of closure models is often constrained to limited flow conditions, leading to large prediction errors in many practical engineering flows that exhibit features such as flow separations, strong pressure gradients, and pronounced mean-flow curvatures.

The exploration of alternatives to conventional CFD methods has been significantly advanced by the emergence of scientific machine learning (SciML), fueled by the increasing availability of high-quality data. This has led to a surge in SciML applications to enhance or potentially replace traditional CFD solvers, offering the promise of accelerated computations and improved predictive performance. The development of deep learning (DL) based surrogate or reduced-order models for fluid flows represents a notable progress. These models leverage advanced DL architectures such as autoencoding-based convolutional neural networks (CNN) [2,3], graph neural networks (GNN) [4,5], Fourier neural operators (FNO) [6,7], DeepONet [8,9], among others, showing initial success in simulating canonical fluid dynamics problems. Despite the promise, the application of purely data-driven DL surrogate models to complex turbulent flows presents substantial challenges due to the inherent chaos and complexity of such scenarios. These models often struggle with maintaining long-term forecasting stability and reliability, as the chaotic dynamics of turbulence lead to rapid accumulation of prediction errors. Additionally, their "black-box" nature demands extensive labeled datasets for training, a prerequisite that is rarely met in most practical situations. Consequently, these purely data-driven DL models face difficulties in maintaining accuracy and generalizing beyond the training regimes.

Hybrid learning approaches, which synergize machine learning (ML) with physics-based modeling, emerge as a strategic solution to the pitfalls of purely data-driven black-box ML methods. This strategy aims to leverage the learning capabilities of ML and theoretical foundation of fluid mechanics, potentially reducing the reliance on extensive training datasets and enhancing model generalizability. One approach in this direction involves utilizing the governing physics as penalty terms to constrain the training process of the ML models, e.g., physics-informed neural networks (PINNs) [10,11], where the governing partial differential equations (PDEs) can be constructed by automatic differentiation (AD) [12–15] or numerical approximation [16–18]. This approach has gained popularity for its ease of implementation and effectiveness in solving many canonical PDE problems [19–23]. However, their efficacy in navigating more complex fluid dynamics scenarios, such as those involving irregular three-dimensional geometries, turbulent and multi-scale phenomena, remains challenging, primarily due to the complexities of balancing multiple objectives during the training process [24,25]. An alternative approach of hybrid modeling is to merge ML with an established physics-based numerical solver. This integration aims to directly enhance the conventional solver's capability to handle complex fluid dynamics by introducing ML-derived insights into the simulation process. By doing so, it not only preserves the fundamental physics represented by traditional numerical solvers but also leverages the predictive power of ML to assimilate available data and improve performance. Specifically in turbulence modeling, ML models have been embedded into traditional CFD solvers for capturing unresolved turbulence through data-driven closure [26–30] or wall modeling [31–33]. The majority of the existing literature focuses on learning the constitutive relationship between unresolved subgrid-scale (SGS) terms and mean-flow features in an *a priori* manner using various ML models, such as symbolic regression [27], random forest [28], and deep neural networks (DNN) [26]. Despite achieving some success, particularly in encoding physical constraints (e.g., Galilean invariance, symmetry, divergence-free) through innovative feature engineering and ML architecture design, these models often fall short in ensuring *a posteriori* accuracy and generalization across different flow conditions. Challenges include the stability of integrating offline-trained closure models with established RANS/LES frameworks [34,35] and the need for direct, accurately computed training labels like Reynolds stress or SGS stress, which are not always readily available or cannot be easily obtained [36,37].

Addressing the challenges mentioned earlier, the trend towards a unified hybrid modeling via differentiable programming is gaining recognition [38]. This innovative strategy enables a holistic optimization/training of both DL and numerical modeling components within a unified learning framework, which allows for direct interaction and feedback between these components, enhancing the overall model's ability to predict complex phenomena with greater accuracy and reliability. Through the integration of differentiable physics-based solvers and advanced DNNs, significant advances have been made across various scientific fields, demonstrating its effectiveness and robustness in learning physics with sparse and indirect training data [39–48]. For example, a JAX-based fully differentiable CFD solver has been integrated with CNNs for more effective and efficient coarse-grained simulations of 2D flows [42]. Adjoint methods have been used to enable end-to-end learning of turbulence closure in RANS and LES simulations, aiming to achieve improved *a posterior* prediction performance [49–52]. Wang and co-workers have introduced the hybrid learning framework from a different perspective, known as *neural differentiable modeling*, which is based on the profound relationship between the numerical representation of PDE operators and neural network architectural components, such as convolution layers, graph kernels, and residual connections [48]. From this perspective, the traditional numerical solvers can be viewed as specialized neural networks predefined by established physics and their numerical representations. The effectiveness of the neural differentiable modeling framework has been demonstrated in learning spatiotemporal dynamics of fluids [48], fluid–structure-interactions [47], and manufacturing processes of composite materials [44–46].

The neural differentiable modeling framework provides an integrated learning platform where physics, represented by discretized PDE operators, is intricately fused with trainable neural network components to create advanced hybrid DL architectures, extending beyond conventional numerical solvers. Despite the promising strides made so far, this field is still in its early stages and requires further development. Specifically, the design of hybrid modeling architectures, which can be highly versatile and critical to learning performance, has not yet been fully explored and optimized. Further research is required to explore different architectural designs of neural differentiable models and to pinpoint potential training bottlenecks, thus pushing the boundaries of this novel hybrid

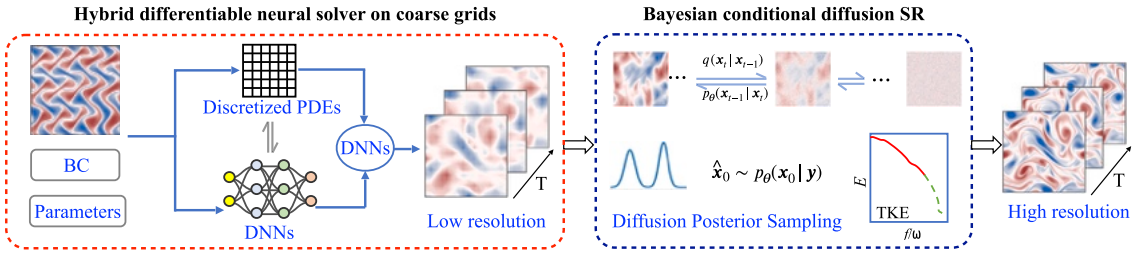

Fig. 1. Overview of the hybrid neural differentiable modeling framework used for simulating spatiotemporal turbulence. It begins by taking the initial and boundary conditions, along with physical parameters, as inputs into a hybrid neural differentiable solver, which combines deep neural networks with coarse-grained PDE operators from governing equations to predict low-resolution turbulent flow fields. The outputs are refined into high-resolution ones by generating small-scale details using a Bayesian conditional diffusion model, enabling super-resolution (SR) predictions.

learning paradigm in modeling complex turbulent flows. In this study, we introduce a data-driven surrogate model for predicting 2D turbulence using the differentiable neural modeling method. The proposed approach is structured into two key stages: (1) a hybrid differentiable neural solver is constructed to capture large-scale turbulent phenomena on a much coarser grid, and (2) a Bayesian conditional diffusion model is then developed to generate small-scale turbulent structures given the large-scale dynamics simulated by the hybrid solver. For the first stage, we explore and compare two innovative hybrid architecture designs for effectively and efficiently predicting large-scale turbulence. In one of our designs, the trainable neural networks are integrated within the differentiable PDE solver as correction terms, serving for SGS closures and truncation error corrections; the second hybrid design features a deeper fusion of trainable ML components and differentiable numerical PDEs, where discretization and interpolation schemes are parameterized as trainable neural networks. The proposed neural differentiable solver is compared against established methods: purely data-driven black-box models and conventional LES with physics-based SGS closures. Through comprehensive comparison and analysis, we aim to highlight the potential advantages in accuracy, computational efficiency, and versatile design offered by the neural differentiable modeling framework in the context of turbulence simulation.

2. Methodology

2.1. Problem formulation

Two-dimensional (2D) turbulence, an idealized model for various large-scale geophysical and environmental flows influenced by rotation and/or stratification, provides fundamental insights into turbulent dynamics relevant to climate modeling, weather forecasting, and oceanography [6,35,53]. Characterized by chaotic spatialtemporal behavior, 2D Kolmogorov turbulence serves as an ideal testbed for developing mathematical and ML-based turbulence models [30,42]. In this study, neural differentiable models are developed to capture the spatiotemporal dynamics of 2D Kolmogorov turbulence, which is governed by the incompressible Navier–Stokes (NS) equations,

$$\begin{aligned} \nabla \cdot \mathbf{u} &= 0, && \mathbf{x}, t \in \Omega_f \times [0, T] \\ \frac{\partial \mathbf{u}}{\partial t} &= -(\mathbf{u} \cdot \nabla)\mathbf{u} + \nu \nabla^2 \mathbf{u} - \frac{1}{\rho} \nabla p + \boldsymbol{f}, && \mathbf{x}, t \in \Omega_f \times [0, T] \end{aligned} \tag{1}$$

where t and $\mathbf{x} = [x, y]$ are time and Eulerian space coordinates, respectively; fluid velocity $\mathbf{u}(t, \mathbf{x})$ and pressure $p(t, \mathbf{x})$ are both spatiotemporal functions defined in $\Omega_f \subset \mathbb{R}^2$; ρ and ν represent fluid density and kinematic viscosity of the fluid, respectively; $\boldsymbol{f}$ represents Kolmogorov forcing term, defined as [54],

$$\boldsymbol{f} = \chi \left[f_x, f_y \right] = \chi \left[\sin(k_f x), 0 \right], \tag{2}$$

where $k_f = 5$ is the forcing wavenumber and $\chi = 1$ is the forcing magnitude. The Reynolds number is fixed as $Re = 500$ in the current work. Solutions for velocity and pressure are uniquely determined by specified initial and boundary conditions (IC/BCs). However, numerically obtaining the solutions at all scales through DNS is computationally infeasible for most practical scenarios.

To address this challenge, we propose a hybrid data-driven neural differentiable modeling framework as illustrated in Fig. 1, where large-scale turbulent flows are captured using a hybrid neural differentiable solver on coarse spatiotemporal resolution, while small-scale turbulent structures are generated by an offline-trained Bayesian conditional diffusion model. Initially, the hybrid neural solver is constructed to accurately predict the flow solution at coarse scales. This is achieved by seamlessly integrating machine learning with physical laws through differentiable programming, operating on coarser grids to reduce computational load while preserving essential large-scale dynamics. Subsequently, a probabilistic diffusion model is trained to generate small-scale turbulent structures conditioned on the large-scale flow predictions. Using Bayesian posterior sampling, the conditional diffusion model functions as a zero-shot super-resolution network to recover high-wavenumber features in the second stage without retraining. Overall, these two components can be integrated during both the training (end-to-end) and inference stages. Currently, we have decoupled these components during training, aimed at optimizing their individual strengths and reducing the computational burden of training. By training the hybrid solvers independently, we ensure they are finely tuned for large-scale flow prediction without the

added complexity of small-scale generation during training. Similarly, the unconditional diffusion model is independently trained to learn the stochastic and complex nature of fine-scale turbulence. This strategy allows the unconditional diffusion model to be trained in an unsupervised manner, which is not tied to one particular low-resolution/high-resolution scenario. After training, the two components are effectively coupled during inference, where the diffusion model stochastically generating small scales flow structures conditioned on the output from the hybrid solver, resulting in a high-fidelity, multiscale solution. Importantly, the integration of these two steps is not merely a sequential process; it demonstrates the feasibility of a one-way coupled framework where large-scale predictions guide the generation of small scales. This setup serves as an essential foundation for developing a fully integrated, two-way coupled modeling framework in the future, where feedback between small-scale generation and large-scale predictions can be incorporated to form a two-way coupling. This two-stage approach leverages the strengths of neural differentiable modeling and generative model to optimize computational efficiency, predictive accuracy, and model fidelity.

2.2. Hybrid neural differentiable solver on coarse grids

In the first stage, a hybrid neural solver is developed on coarse grids to capture the large-scale dynamics by integrating a numerical solver with deep neural networks through differentiable programming. We propose and explore two innovative hybrid learning architectural designs. In Design I, trainable deep neural networks are integrated with the PDE solver, serving as both SGS closures and truncation error corrections. In Design II, the interpolation schemes of the PDE solvers are parameterized as trainable neural networks, learning advanced high-order discretization from data. Details of these two architectural designs are provided below.

2.2.1. Design I: hybrid neural-PDE correction model

In this design, we incorporate trainable DNN-based correction terms into the governing fluid equations to enhance the accuracy of flow predictions on coarse grids. As illustrated in Fig. 2, the model adopts a recurrent neural network architecture, where the Navier–Stokes equations are encoded as fixed convolution operations, indicated as $\mathrm{ConvPDE}(\cdot)$. This is followed by a sequence of trainable convolutional LSTM (ConvLSTM) operators, $\mathrm{ConvLSTM}(\cdot)$ with trainable parameters $\boldsymbol{\theta}_{nn}$, which act as corrective adjustments to the flow predictions. Specifically, the flow solution $\boldsymbol{V}_t = [\mathbf{u}_t, p_t]^T$ at time t is modeled as,

$$\boldsymbol{V}_t = \boldsymbol{V}_{t-1} + \mathrm{ConvPDE}(\boldsymbol{V}_{t-1}) + \mathrm{ConvLSTM}\Big(\boldsymbol{V}_{t-1}, \mathrm{ConvPDE}(\boldsymbol{V}_{t-1}), \boldsymbol{H}_{t-1}; \boldsymbol{\theta}_{nn}\Big), \tag{3}$$

where $\boldsymbol{H}_{t-1}$ is the hidden state from previous step and the PDE-encoded convolution operation is defined as,

$$\mathrm{ConvPDE}(\boldsymbol{V}_{t-1}) = \begin{cases} \delta\mathbf{u}_t^* = \Delta t\Big(-(\mathbf{u}_{t-1}\cdot\nabla)\mathbf{u}_{t-1} + \nu\nabla^2\mathbf{u}_{t-1} + \boldsymbol{f}\Big), \\ \delta p_t^* = S^{-1}\rho\nabla\cdot(\mathbf{u}_{t-1} + \delta\mathbf{u}_t^*)/\Delta t, \end{cases} \tag{4}$$

where S is discretized Poisson operator and Δt is time step. The detailed schematic of the recurrent network unit is depicted in Fig. A.20. In this setup, flow predictions – including velocity and pressure from the previous step – are advanced using the Navier–Stokes equations discretized on coarse grids, functioning as a coarse explicit fluid solver; the outputs are then processed through several convolutional layers, which are optimized during model training. The hybrid neural differentiable model is constructed and trained in an autoregressive manner, where the loss function is derived from the model predictions over multiple time steps. This setup requires every component of the hybrid model to be differentiable, enabling holistic end-to-end training. Conceptually, this learning architecture can be viewed as a coarse fluid solver coupled with trainable DNNs, which function as corrective mechanisms for unresolved subgrid-scale structures and numerical discretization errors. However, this model is beyond a mere combination of a classic coarse solver and DNN corrections, due to its sophisticated integration as a recurrent neural networks with hidden connections. We refer to this design as "NeuralPDE-Corr". While prior works, such as List et al. [43], introduced the similar concept of using machine learning to learn corrections within numerical solvers, our NeuralPDE-Corr architecture differs significantly in its structure and implementation. Our proposed hybrid differentiable neural solver can be viewed as a convolutional LSTM network with trainable hidden memory, where the discretized governing PDEs are encoded as fixed convolution operations and integrated with trainable convolution operations at different layers. From a DL perspective, the model in List et al. [43] is a purely autoregressive next-step model based on the PISO scheme, with a few CNN layers embedded to modify the intermediate velocity $\boldsymbol{u}^*$.

2.2.2. Design II: Hybrid neural-PDE discretization model

In the second design, trainable DNNs are utilized to learn high-order numerical discretizations, facilitating effective discretized solution approximation at cell faces within a FVM framework to enhance the prediction accuracy with coarse grids [42]. While this architecture builds on a concept similar to the learned interpolation approach presented by Kochkov et al. [42], the hybrid architectural design is significantly different from that of [42]. Specifically, as depicted in Fig. 3, the entire model operates as a recurrent neural network, where certain parts of the Navier–Stokes equations are formulated as fixed convolution operations, coupled with a series of trainable ConvLSTM layers. In contrast to Design I, the flow solution $\boldsymbol{V}_t = [\mathbf{u}_t, p_t]^T$ at time t is modeled as,

$$\boldsymbol{V}_t = \begin{cases} \boldsymbol{u}_{t-1} + \mathrm{ConvPDE}\Big(\boldsymbol{u}_{t-1}, \boldsymbol{f}, \mathrm{ConvLSTM}_1(\boldsymbol{u}_{t-1}, \boldsymbol{H}_{t-1}; \boldsymbol{\theta}_{nn})\Big), \\ p_{t-1} + \mathrm{ConvLSTM}_2\Big(p_{t-1}, \boldsymbol{H}_{t-1}; \boldsymbol{\theta}_{nn}\Big), \end{cases} \tag{5}$$

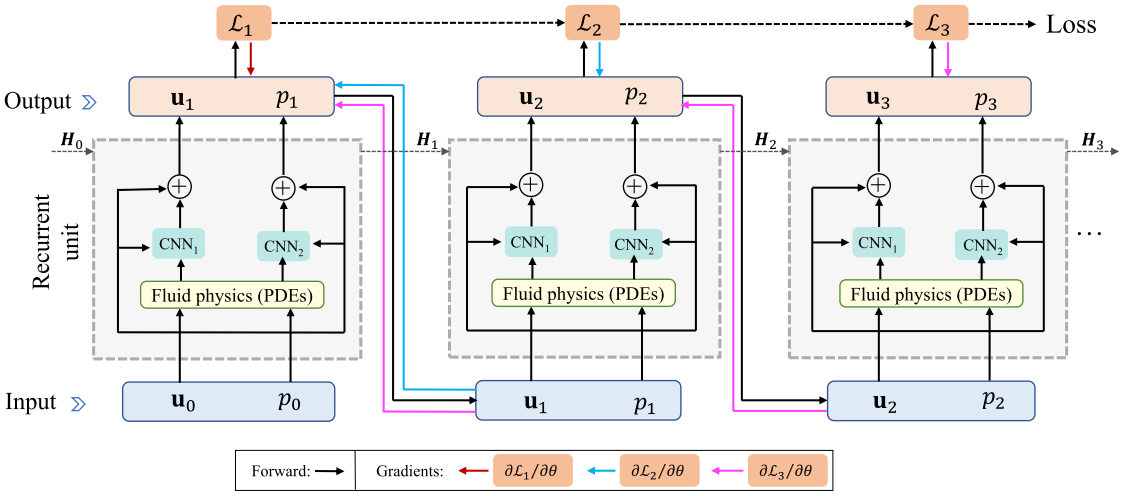

Fig. 2. The schematic of the hybrid learning architecture I: neural-PDE correction model (NeuralPDE-Corr).

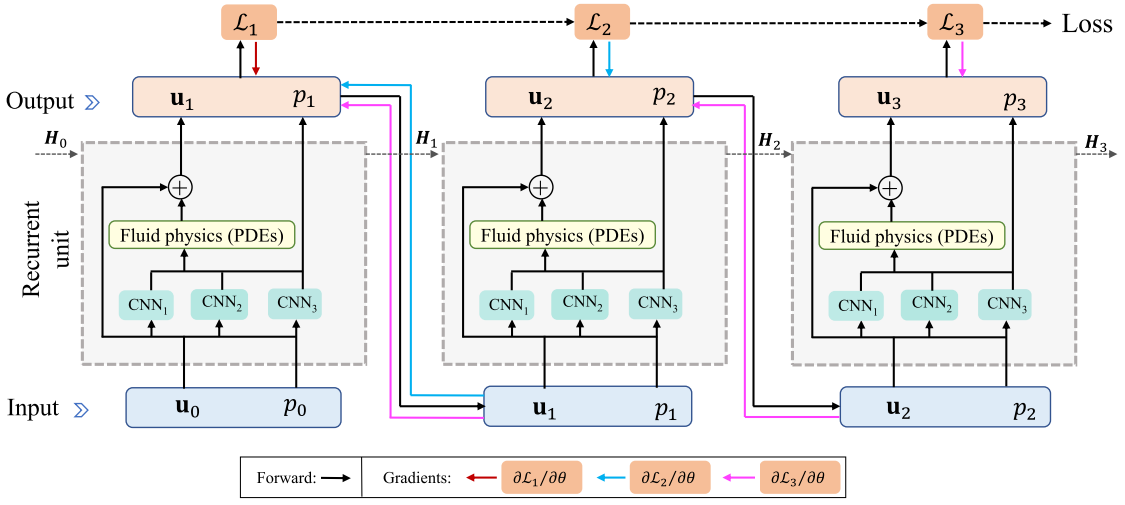

Fig. 3. The schematic of the hybrid learning architecture II: neural-PDE discretization model (NeuralPDE-Discretize).

where the PDE-encoded convolution operation is defined as,

$$\text{ConvPDE}(\boldsymbol{u}_{t-1}) = \delta\mathbf{u}_t^* = \Delta t\left(-(\mathbf{u}_{t-1}^N \cdot \nabla)\mathbf{u}_{t-1}^N + (\nu\nabla^2\mathbf{u}_{t-1} + \mathcal{D}_N) + \boldsymbol{f}\right) \tag{6}$$

where $\mathbf{u}_{t-1}^N = \text{ConvLSTM}_1(\boldsymbol{u}_{t-1}, \boldsymbol{H}_{t-1}; \boldsymbol{\theta}_{nn})$ and $\mathcal{D}_N = \text{ConvLSTM}_2(\nu\nabla^2\mathbf{u}_{t-1}, \boldsymbol{H}_{t-1}; \boldsymbol{\theta}_{nn})$. Details of the recurrent unit for this design are presented in Fig. A.21. In this model, the final predictions of velocity and pressure at time t are advanced by the Navier–Stokes equations, discretized on coarse grids. Unlike the first design, which relies on DNNs for corrective and closure modeling, this approach uses one trainable DNN to learn high-order numerical discretization in a data-driven manner, replacing traditional second-order central interpolations typically used for convection at cell faces. Numerical dissipation and truncation errors, exacerbated by coarse grids, are compensated by another trainable neural networks. Moreover, to address the low solution accuracy of the Poisson equation on very coarse grids, the time stepping scheme for pressure is directly learned by neural networks. This integration of trainable neural networks with discretized PDEs is referred to as "NeuralPDE-Discretize". As with the first architecture, the NeuralPDE-Discretize model is meticulously constructed and systematically trained in an autoregressive manner, offering a more integrated and potentially complex approach. However, this extensive integration poses risks of numerical instability if neural network outputs are not adequately constrained [55,56]. Comparative analyses in subsequent experiments will evaluate the performance of each design.

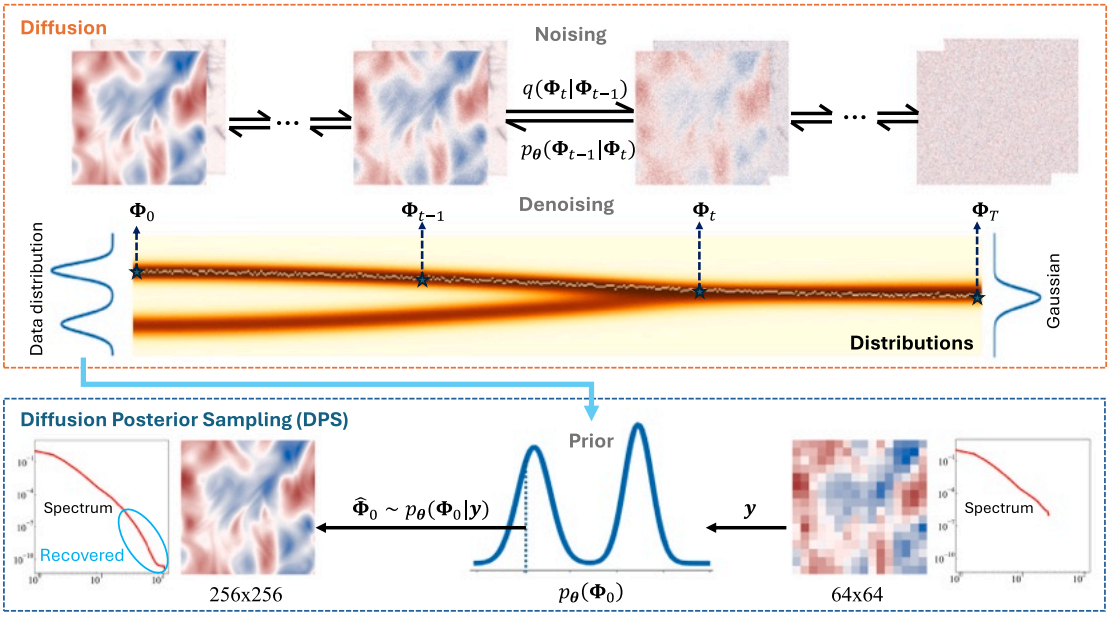

Fig. 4. Schematics of the spectrum-decomposed diffusion-based SR model, where low- and high-wavenumber components are performed in two separated channels. Diffusion posterior sampling allows for conditional generation of high-resolution flow predictions for provided low-resolution flow solutions without retraining.

2.3. Diffusion-based superresolution network for recovering small scales

Directly resolving all scales of turbulence, especially at high Reynolds numbers, is computationally infeasible. The proposed hybrid neural differentiable model overcomes this challenge by leveraging state-of-the-art (SOTA) generative AI techniques to recover high-frequency features associated with small scales. Specifically, we utilize probabilistic diffusion-based generative models, which have recently shown great promise in generating spatiotemporal turbulence [57,58] and reconstructing high-fidelity flow fields from sparse or low-resolution (LR) data [59]. As illustrated in Fig. 1, we propose a novel spectrum-decomposed diffusion-based super-resolution (SR) model, which is trained offline to enhance the low-resolution output from the coarse-grid hybrid neural differentiable solver by generating high-wavenumber turbulence. The diffusion model operates by learning a reverse Markovian diffusion process, which enables generation starting from a Gaussian random field and progressively converging to the target data distribution [60,61]. To achieve SR generation, a diffusion posterior sampling technique is used to enable conditional generation given the low-resolution coarse-grid flow predictions.

During training, the high-resolution flow data $\boldsymbol{\Phi}_0$ from the target distribution $q(\boldsymbol{\Phi}_0)$ are sequentially corrupted based on a Gaussian transition kernel $q(\boldsymbol{\Phi}_t|\boldsymbol{\Phi}_{t-1}) = \mathcal{N}(\boldsymbol{\Phi}_t; \sqrt{1-\beta_t}\boldsymbol{\Phi}_{t-1}, \beta_t I)$ parameterized by a noising schedule β_t, as shown in the upper section of Fig. 4). This forward process involves sequentially adding Gaussian noise ϵ, such that $\boldsymbol{\Phi}_{t+1} = \boldsymbol{\Phi}_t + \beta_t\epsilon, \quad \epsilon \sim \mathcal{N}(0, I)$, leading to a forward Markov process, $\boldsymbol{\Phi}_0 \to \boldsymbol{\Phi}_1 \to \dots \boldsymbol{\Phi}_{t-1} \to \boldsymbol{\Phi}_t \dots \to \boldsymbol{\Phi}_T$. The diffusion model learns to denoise the corrupted sample $\boldsymbol{\Phi}_t \to \boldsymbol{\Phi}_{t-1}$ by parameterizing the reverse Gaussian transition kernel using neural networks with trainable parameters θ as,

$$p_\theta(\boldsymbol{\Phi}_{t-1}|\boldsymbol{\Phi}_t) = \mathcal{N}\left(\boldsymbol{\Phi}_{t-1}; \frac{1}{\sqrt{1-\beta_t}}\left(\boldsymbol{\Phi}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\boldsymbol{\Phi}, t)\right), \beta_t I\right), \quad \bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s) \tag{7}$$

where $\epsilon_\theta(\boldsymbol{\Phi}, t)$ is a neural network approximating the noise ϵ from $\boldsymbol{\Phi}_t$. The spectrum-decomposed training process for ϵ_θ is detailed in Section 2.4. Once trained, the diffusion model is able to generate high-resolution flow samples from the learned distribution $\hat{\boldsymbol{\Phi}}_0 \sim p_\theta(\boldsymbol{\Phi}_0)$, which approximates the true distribution $q(\boldsymbol{\Phi}_0)$, underlying the training data. These distributions are referred to as prior distributions.

For SR generation given low-resolution flow predictions $\boldsymbol{\Phi}_\mathrm{L}$ from the coarse-grid hybrid neural solver, the diffusion posterior sampling (DPS) method [62] is utilized. As depicted in the lower part of Fig. 4, for a given mapping function $\mathcal{M} : \boldsymbol{\Phi}_0 \mapsto \boldsymbol{\Phi}_\mathrm{L}$, we can sample the trained diffusion model conditioned on low-resolution flow predictions $\boldsymbol{\Phi}_\mathrm{L}$ without retraining. From a Bayesian perspective, this process involves sampling the conditional probability $p_\theta(\boldsymbol{\Phi}_0|\boldsymbol{\Phi}_\mathrm{L})$, which can be connected to the unconditioned density $p_\theta(\boldsymbol{\Phi}_0)$ via Bayes' rule,

$$p_\theta(\boldsymbol{\Phi}_0|\boldsymbol{\Phi}_\mathrm{L}) \propto p_\theta(\boldsymbol{\Phi}_\mathrm{L}|\boldsymbol{\Phi}_0)p_\theta(\boldsymbol{\Phi}_0) \tag{8}$$

Since $p_\theta(\boldsymbol{\Phi}_0)$ has already been learned by the unconditional diffusion model on high-resolution DNS data, we only need a function $\mathcal{M}$ that relates the low-resolution data $\boldsymbol{\Phi}_\mathrm{L}$ to the SR data $\boldsymbol{\Phi}_0$, which can be defined as,

$$\boldsymbol{\Phi}_\mathrm{L} = \mathrm{ifft}(\mathrm{fft}(\hat{\boldsymbol{\Phi}}_0)[:k_\mathrm{cut}]) \tag{9}$$

where $\boldsymbol{\Phi}_\mathrm{L}$ represents the low-resolution predictions obtained from the hybrid neural solver, $\hat{\boldsymbol{\Phi}}_0$ is the generated high-resolution prediction, and k_cut is the cut-off wave number determined by the spatial resolution. This is achieved by modifying the sampling process of the unconditionally trained diffusion model $p_\theta(\boldsymbol{\Phi}_0)$ (i.e., prior) with a updated score function using the gradient of the log likelihood function $-\nabla_{\boldsymbol{\Phi}_i}\|\boldsymbol{\Phi}_\mathrm{L} - \mathcal{M}(\hat{\boldsymbol{\Phi}}_0)\|_2^2$. More details can be fund in Du et al. [58].

The proposed diffusion SR component significantly differs from prior work, such as Shu et al. [59], which utilized the standard denoising diffusion probabilistic model (DDPM) [61], for flow reconstruction from LR/sparse measurements by starting the denoising process from an intermediate noised state determined by LR/sparse data. In contrast, we designed a novel spectrum-decomposed diffusion process combined with Bayesian diffusion posterior sampling, enabling zero-shot conditional generation of small-scale turbulence directly from the LR predictions of hybrid neural solvers. This approach effectively captures the full spectrum of turbulence, significantly enhancing the fidelity of generated small-scale features, even when the energy levels are exceedingly low.

2.4. Training details of the entire framework

This subsection offers an overview of the training recipe, which largely influences the performance of the proposed model. Understanding and optimizing these training details are crucial to leveraging the full potential of the hybrid neural differentiable model in learning turbulent flows. Specifically, the training of low-resolution hybrid neural solver and diffusion-based SR module are decoupled in two stages. The source of our training and testing data is high-resolution DNS performed on a grid of $N_x \times N_y = 2048 \times 2048$, utilizing a fractional step method as detailed in [47]. For training the coarse grid neural solver, the high-resolution DNS data $\boldsymbol{\Phi}^\mathrm{DNS}$ undergoes filtering and down-sampling to a coarse resolution of 64×64, forming the low-resolution dataset, $\mathcal{A}_\mathrm{train} = \{\tilde{\boldsymbol{\Phi}}_i\}_{i=1}^{1600}$, where $\tilde{\boldsymbol{\Phi}} = [\tilde{\Phi}^1, \tilde{\Phi}^2, \tilde{\Phi}^3]^T$ corresponds to three distinct initial conditions.

Training for coarse-grid neural solver. Both hybrid Neural-PDE models are trained autoregressively in a sequence-to-sequence (Seq2Seq) manner, where the loss function is defined as follows,

$$\mathcal{L}(\theta) = \alpha_1 \frac{1}{N}\sum_{t=0}^{N-1}\left\|\tilde{\mathbf{u}}^t(\mathbf{x};\theta) - \mathbf{u}_d^t(\mathbf{x})\right\|_{L_2}^2 + \alpha_2 \frac{1}{N}\sum_{t=0}^{N-1}\left\|\tilde{p}^t(\mathbf{x};\theta) - p_d^t(\mathbf{x})\right\|_{L_2}^2 \tag{10}$$

where $\|\cdot\|_{L_2}$ represents the L2 norm, N indicates the total number of rollout time steps, $\tilde{\square}$ denotes the rollout predictions, and the subscript $\square_d$ indicates the labeled data. The specified weights, $\alpha_1 = 1$ and $\alpha_2 = 100$, are based on the relative importance of velocity and pressure magnitudes. This structured loss is minimized over a long rollout trajectory using stochastic gradient descent, enhancing both the accuracy and stability of the model's autoregressive inference performance as noted in [42,63]. This multi-step rollout training process is enabled by differentiable programming, which allows the gradient to be back-propagated over the entire rollout trajectory. However, implementing Seq2Seq training through differentiable programming may introduce specific challenges: (1) Training instability: the hybrid neural solver is prone to training instability, especially in the early training phase. This is because the initialization of trainable neural network parameters is nonphysical, which can result in numerical instability in the PDE-encoded portion and lead to training failure. (2) Extensive memory demands: training over a long trajectory require substantial GPU memory, since the gradients of nested functions from numerous model rollout steps have to be stored for end-to-end optimization. To address these challenges effectively, we employ the following training recipe: (i) Training is structured with a total of $N = 200$ unrolling steps, with the loss being calculated every 2 steps. The learning step, dt_neural, is defined as $8 \times dt_\mathrm{physics}$, matching the time step used in the training dataset $\mathcal{A}_\mathrm{train}$. (ii) We gradually increase the number of unrolling steps from $N = 40$ to $N = 200$ in increments of $\delta = 40$ through transfer learning, which helps in maintaining stability over long trajectories. (iii) To circumvent memory limitations, training is parallelized across multiple GPUs. The differentiable neural solver is developed in JAX, taking advantage of jax.pmap to parallelize the computations across batch dimensions. Currently, four batches with differing initial conditions are used, and the gradients from each GPU are aggregated to update the training parameters each epoch. In the hybrid neural models, all CNN blocks are uniformly configured with five layers, following the channel configuration of $[32, 64, 64, 32, 1]$ with a trainable 3×3 convolutional kernel. We intentionally kept the neural network architecture as streamlined as possible to ensure that the model remains anchored in the underlying physical principles. The depth and width of the network are designed to be sufficient to capture the essential features of the turbulent flows while avoiding unnecessary complexity that might make the model move towards a purely data-driven approach, potentially diminishing the embedded physics-based insights. A Leaky rectified linear unit (LeakyReLU) serves as the activation function for all layers except the last one, which is linear. We selected LeakyReLU over standard ReLU to address the "dying ReLU" problem, where neurons can become inactive and cease learning if they encounter a zero gradient. LeakyReLU maintains a small, non-zero gradient for negative inputs, which helps sustain learning across all neurons, especially in the context of complex, non-linear turbulence phenomena. The optimization settings are as follows: (a) Initial Learning Rate: 10^{-4}, (b) Optimizer: Adam, (c) Scheduler: Cosine Decay Schedule with alpha = 10^{-9}.

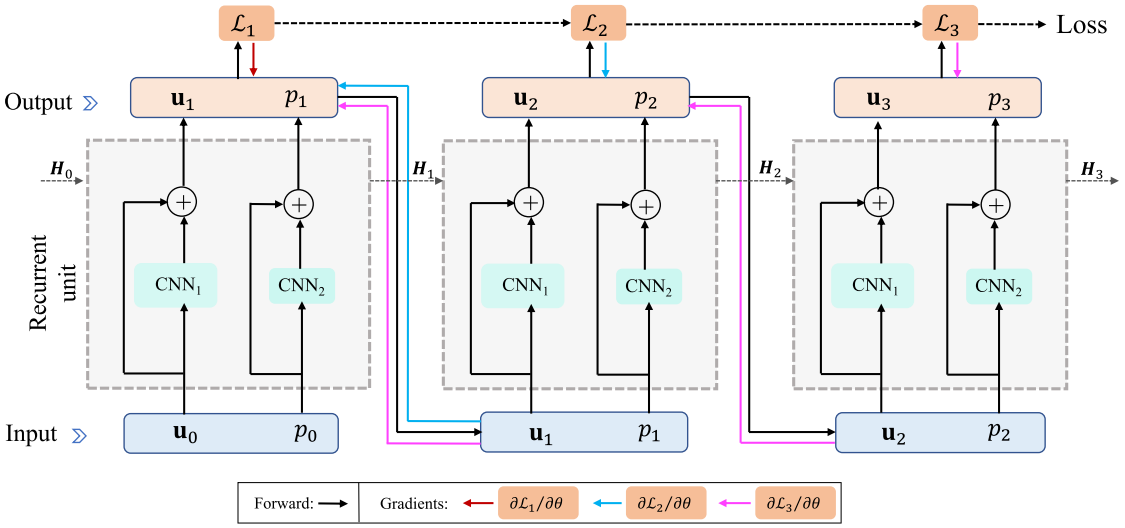

Fig. 5. Schematic of the purely data-driven model illustrating how velocity and pressure are predicted using CNN_1 and CNN_2. The model is trained and applied in an autoregressive manner.

Training for spectrum-decomposed diffusion-based SR model. The training strategy for the diffusion-based SR portion is illustrated in Fig. 4. The diffusion model is trained unconditionally to learn the prior distribution $p_\theta(\boldsymbol{\Phi}_{\mathrm{H}})$ directly from the un-labeled high-resolution DNS dataset. To enhance the model's capability of generating high-wavenumber flow structures, we spectrally decompose the high-resolution DNS data, $\boldsymbol{\Phi}^{\mathrm{DNS}}$, into two components: $\tilde{\boldsymbol{\Phi}}_{\mathrm{H}}$ and $\tilde{\boldsymbol{\Phi}}_{\mathrm{L}}$, representing two different energy scales. The decomposition is based on the cutoff wavenumber k_{cut}, which is determined by the spatial resolution of the coarse-grid neural solver. This spectral decomposition forms the training dataset $\mathcal{B}_{\mathrm{train}} = [\tilde{\boldsymbol{\Phi}}_{\mathrm{H}}, \tilde{\boldsymbol{\Phi}}_{\mathrm{L}}]^T$, allowing for focused learning on distinct wavenumber bands.

Our spectrum-decomposed diffusion-SR models are trained on $\mathcal{B}_{\mathrm{train}}$, and generation are performed from two separate channels corresponding the two distinct wavenumber bands, which are combined to for the generated high-resolution flow solution. Separate diffusion models are trained for each velocity component, $\boldsymbol{u} = [u, v]^T$, and pressure, p, focusing solely on the high-resolution dataset to ensure that the fine-scale features are precisely learned without any low-resolution input. The diffusion-SR models utilize a U-net architecture to approximate the noise component ϵ_θ, through the optimization of the following objective function [60],

$$L_{simple}(\theta) := \mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\|^2\right] \tag{11}$$

This spectrum-decomposed training strategy not only enhances the model's performance in terms of predicting small-scale flow features but also ensures stability and efficiency in handling extensive data scales, which effectively addresses the computational challenges posed by high-resolution data handling and long training trajectories.

2.5. Baseline models for comparison

In evaluating the effectiveness of the proposed hybrid neural models, it is essential to compare their performance against established modeling approaches. Two baseline models, one purely data-driven and the other purely physics-based, are utilized for this purpose. These two baseline models represent conventional approaches in ML and CFD, serving as reference points to study the performance of the proposed hybrid learning strategy.

2.5.1. Purely data-driven neural solver

To evaluate the role of integrating discretized flow physics into the neural solver, we compare our hybrid model against a purely data-driven neural solver that employs a similar ConvLSTM architecture but lacks any explicit incorporation of physical laws. This "black-box" approach utilizes the ConvLSTM to predict future flow states solely based on historical data, without any physics-based guidance. The recurrent network can be formulated as,

$$\boldsymbol{V}_t = \boldsymbol{V}_{t-1} + \mathrm{ConvLSTM}\Big(\boldsymbol{V}_{t-1}, \boldsymbol{H}_{t-1}; \boldsymbol{\theta}_{nn}\Big), \tag{12}$$

which illustrates how this model leverages sequential data to autoregressively update velocity and pressure fields at each time step. The primary distinction from the hybrid models shown in Figs. 2 and 3 lies in the absence of any fluid physics modules, highlighting the model's reliance on the learned patterns purely from data rather than governing equations (see Fig. 5).

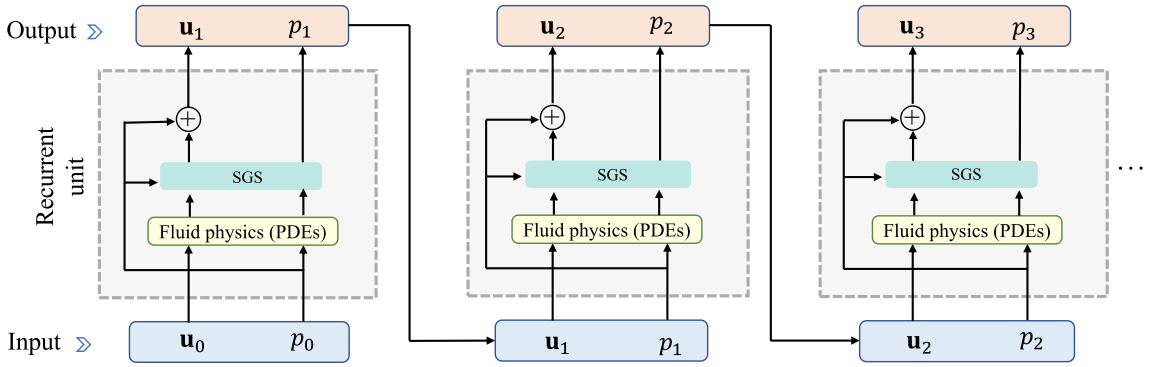

Fig. 6. Schematic of the purely physics-based solver with classic SGS closures.

2.5.2. Purely physics-based solver with classic SGS closures

In our comparison study, we also include a purely physics-based solver utilizing classic SGS closures, commonly known as LES. The LES model serves as a benchmark for evaluating the benefits of incorporating learned elements via DNNs against traditional numerical methods that rely entirely on established governing physics. Specifically, the well-established Smagorinsky SGS model is utilized to address unresolved physics due to coarse computational grids. Unlike data-driven models, the Smagorinsky model provides corrective terms by approximating the effects of smaller, unresolved scales through the calculation of turbulent viscosity based on local strain rates. This method offers a physics-derived approach for representing the dissipative behavior of the scales not captured on coarse grids. The filtered Navier–Stokes equations, which incorporate the SGS model, are expressed as follows,

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u}\cdot\nabla)\mathbf{u} + \nu\nabla^2\mathbf{u} - \frac{1}{\rho}\nabla p + \mathbf{f} - \nabla\cdot\boldsymbol{\tau}^{SGS} \tag{13}$$

where $\boldsymbol{\tau}^{SGS} = -2\nu_{SGS}\bar{\mathbf{S}}$ is the subgrid-scale stress tensor, $\bar{\mathbf{S}}$ is the strain rate and the SGS viscosity ν_{SGS} is given by Smagorinsky model:

$$\nu_{SGS} = (C_s\Delta)^2\left|\bar{S}\right| \tag{14}$$

where Smagorinsky constant is $C_s = 0.2$, Δ is the grid size. In essence, the traditional LES model operates similarly to a non-trainable recurrent neural network. As illustrated in Fig. 6, it systematically utilizes physical laws to define convolution operations that compute each new state based on the solutions at previous time step, mirroring the standard recurrent network structure without hidden memory and learning capability. Namely, the LES model can be mathematically expressed as an autoregressive recurrent network,

$$\boldsymbol{V}_t = \boldsymbol{V}_{t-1} + \mathrm{ConvPDE}(\boldsymbol{V}_{t-1}) + \mathrm{SGS}(\boldsymbol{V}_{t-1}), \tag{15}$$

3. Numerical results

3.1. Predictions at low resolution by hybrid neural models

3.1.1. Temporal forecasting starting from training ICs

The trained hybrid neural models were applied to forecast the spatiotemporal dynamics of Kolmogorov turbulence, beginning from the initial conditions used during training. To facilitate the comparison across different models, the superresolution components were excluded in this analysis. All results were generated using a coarse mesh of 64×64. The vorticity contours for each model are presented in Fig. 7, spanning both the training window from $T = 0$ to $T = 1600$ and the forecasting window from $T = 1600$ to $T = 4800$. Overall, both hybrid neural models, NeuralPDE-Corr and NeuralPDE-Discretize, successfully forecast the dynamics over a temporal trajectory three times longer than their training duration. However, there are still noticeable differences in their performance. The NeuralPDE-Corr model exhibits slight nonphysical oscillations in the vorticity predictions, particularly visible at $T = 3200$ and persisting through $T = 4800$, as highlighted by the blue frames. In contrast, NeuralPDE-Discretize model manages to eliminate these spurious oscillations, although the predicted patterns has a larger discrepancy in some regions after extensive rollouts. These performance differences can be attributed to the architectural differences between the two hybrid models. The NeuralPDE-Corr model, as detailed in Section 2.2.1, incorporates trainable convolutional layers as the final output layers, with each step's outputs directly produced by the CNNs. This configuration implies that while the predicted velocity and pressure fields are informed by the underlying physical laws, they are not strictly constrained by them, potentially leading to slight nonphysical oscillations. Conversely, the NeuralPDE-Discretize model, described in Section 2.2.2, begins with trainable CNNs, which are seamlessly integrated with the physics by forming the discretized governing PDEs. This effectively transform the model into a

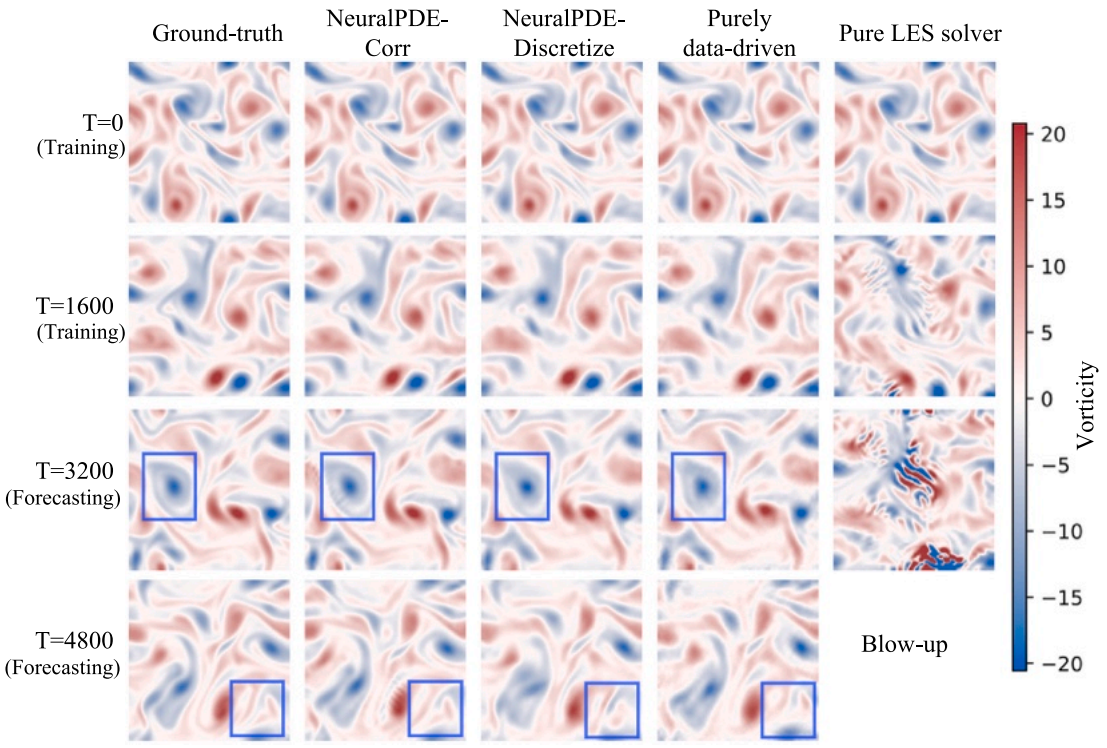

Fig. 7. Temporal progression of vorticity contours predicted by different models on a coarse mesh, presented at key stages from training through extended forecasting. Notably, blue boxes highlight specific instances in the forecasting phase where significant differences between the models become apparent, including a blow-up scenario observed in the purely physics-based LES solver at $T = 4800$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

FV-based solver, where traditional interpolation schemes are replaced by trainable neural networks. This integration ensures that outputs adhere closely to discretized physical laws. However, due to the chaotic nature of turbulence, even minor deviations in flux predictions can lead to significantly different patterns over an extended rollout trajectory, as evidenced by the differences observed at $T = 4800$.

The forecasts from two baseline models, one purely data-driven and the other entirely physics-based, are compared. The purely data-driven model, which relies solely on historical data without incorporating physical laws, demonstrates competent performance in temporal forecasting from training initial conditions. However, it still slightly underperforms compared to the hybrid models due to its tendency to exhibit vorticity diffusion over extended simulations, leading to blurred vortex boundaries, particularly among larger vortices, as shown in Fig. 7. Additionally, significant nonphysical oscillations and non-smoothness are apparent in the predicted pressure fields, as shown in Fig. 8. While it performs well with known initial conditions, the purely data-driven approach's effectiveness significantly declines when faced with unseen initial conditions, highlighting its limitations in generalizing beyond the training scenarios, which will be further discussed in the subsequent subsection. As for the purely physics-based baseline model, i.e., the traditional LES model with the Smagorinsky SGS closure, it fails to maintain stability on a coarse computational mesh and experiences a blow-up by $T = 3200$. Despite the Courant–Friedrichs–Lewy (CFL) number being 0.64, numerical instability occurs due to inadequate grid resolution and the use of a basic explicit time integration scheme. This comparison underscores the superiority of hybrid neural models that integrate ML with physics-based solvers, enabling them to operate effectively on much coarser grids and with larger time steps compared to conventional numerical solvers.

To further quantitatively assess the error propagation in time for all models, the relative errors ϵ^t of model prediction at each time step t is computed as follows,

$$\epsilon^t = \frac{1}{N_\lambda} \sum_{i=0}^{N_\lambda} \left(\frac{\|\tilde{\mathbf{u}}^t(\lambda_i) - \mathbf{u}_d^t(\lambda_i)\|_2}{\|\mathbf{u}_d^t(\lambda_i)\|_2} \right) \tag{16}$$

where N_λ is the size of parameter ensemble. The computed relative errors for each model, shown in Fig. 9, reveal that the hybrid neural models, NeuralPDE-Corr and NeuralPDE-Discretize, generally outperform the purely data-driven model throughout the forecasting phase. Notably, the error trajectory of the purely data-driven model (green line) starts low but quickly grows by an order of magnitude during the forecast period, a common issue in data-driven forecasting due to error accumulation. Of the two hybrid models, NeuralPDE-Corr (orange line) consistently exhibits lower errors than NeuralPDE-Discretize (blue line). This can be attributed to the different operational mechanics of the two hybrid architectural designs. As indicated by the loss defined in Eq. (10),

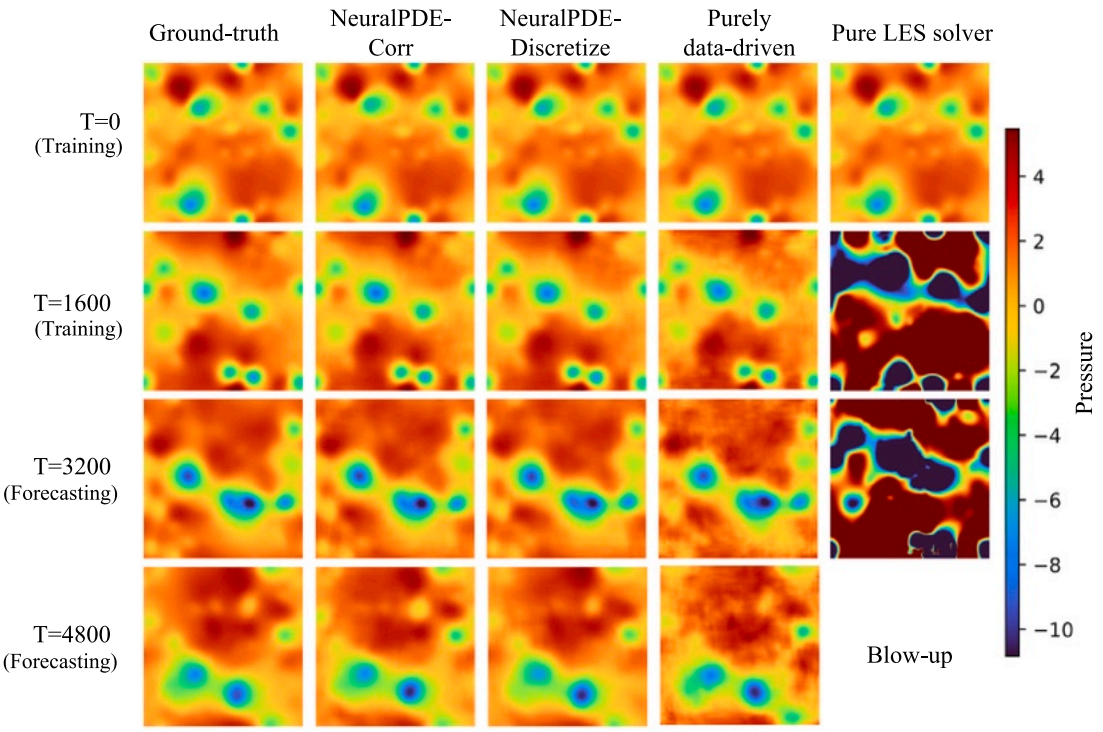

Fig. 8. Pressure predictions from different models on a coarse mesh. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

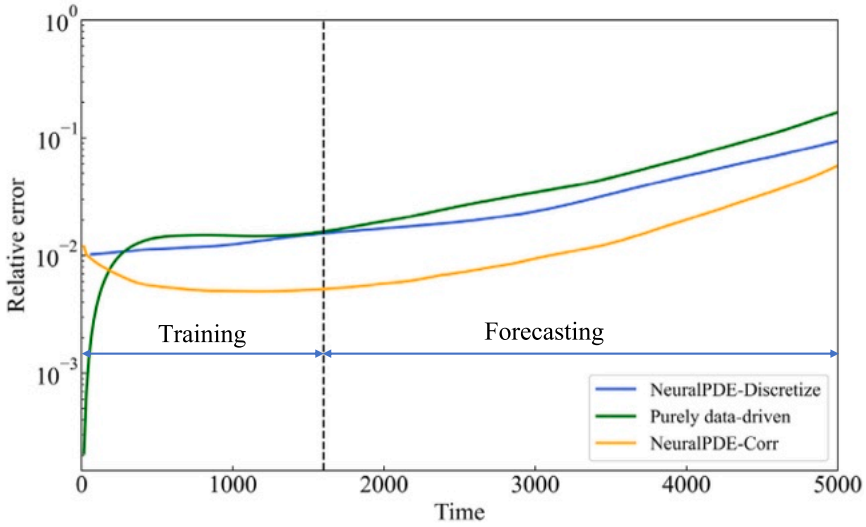

Fig. 9. Relative error comparison across all models for initial conditions seen during training. Due to its substantially higher error, the pure LES solver's results is excluded. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

both models aim to minimize the data mismatch in terms of instantaneous flow patterns. This objective is more straightforwardly achieved for the NeuralPDE-Corr model, which employs neural networks to refine outputs from the coarse PDE solver, effectively replicating the flow patterns for the training scenario. In contrast, NeuralPDE-Discretize, which learns interpolation schemes by ML within the numerical solver, is closer to a classic numerical solver, which can simulate different flow patterns after long-term

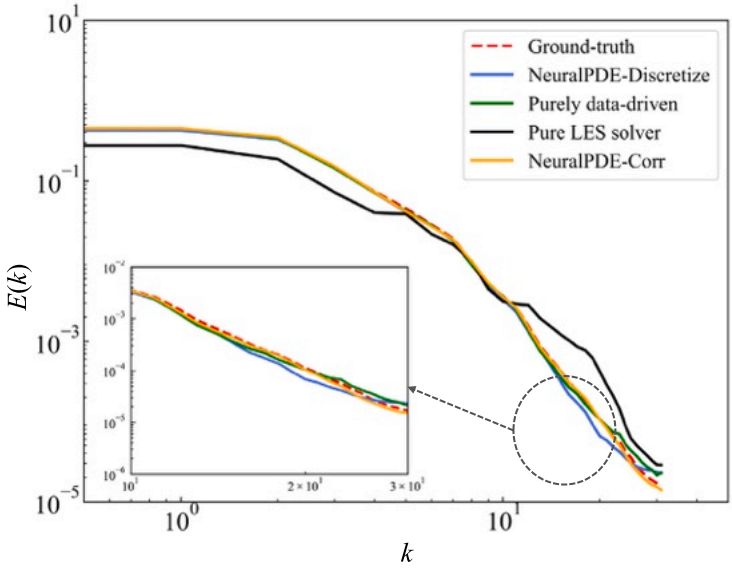

Fig. 10. Energy spectra comparison across different models, averaged over the period from $T = 2400$ to $T = 4800$ to ensure decorrelation from initial conditions. For the Pure LES solver, energy spectra are calculated from $T = 2400$ to $T = 3200$ only, due to model divergence beyond $T = 3200$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

model rollouts given the chaotic nature of turbulence. This difference becomes particularly pronounced in the forecasting phase, where the inherently chaotic nature of turbulence leads to divergent outcomes from similar initial conditions. Consequently, while both models aim to match instantaneous flow patterns, NeuralPDE-Corr achieves a more accurate alignment with the data when evaluated by the relative MSE.

Fig. 10 presents the comparison of energy spectra across different models, benchmarked against ground truth, obtained from high-resolution DNS with spectral filtering (red line). Overall, the LES on very coarse space–time resolution fails to capture the energy spectrum, while the turbulence predictions from both hybrid and purely data-driven models align well with the ground truth, demonstrating their robustness in capturing essential statistical features by incorporating data. From the zoomed-in view, slight discrepancies in the high wave number regions ($k > 10$) can be observed in the predictions by the NeuralPDE-Discretize and purely data-driven models, with the former underpredicting and the latter overpredicting energy at higher wave numbers. In contrast, the NeuralPDE-Corr model closely matches the filtered DNS statistics, precisely replicating the energy distribution.

3.1.2. Temporal forecasting from randomly generated ICs

The inherently chaotic nature of turbulence leads to diverse flow patterns from different initial conditions. To assess the generalizability of our trained models, we conducted tests using initial conditions that were not seen during the training phase. These initial velocity and pressure fields were synthesized by sampling a Gaussian process formulated through Karhunen–Loeve (KL) expansion [64]. Each model was then tasked with simulating the flow over a period of $T = 3200$, identical to the number of inference steps used in earlier scenarios.

As observed in the vorticity and pressure contours presented in Figs. 11 and 12, only the NeuralPDE-Corr model remains closely aligned with the ground-truth after a long-term rollout from unseen initial conditions. It effectively captures the dynamics of Kolmogorov turbulence with only minor deviations, which do not detract from the model's overall accuracy in replicating the turbulence dynamics. In stark contrast, the purely data-driven model fails to generalize under these unseen initial conditions, as its predictions show considerable divergence from the ground truth and the predicted flow patterns are completely different from the expected ones even at very early rollout stage. By $T = 1600$, the model begins to exhibit dispersed small-scale vortices and blurred large-scale structures, with these issues exacerbating over time. This highlights the limitations of purely data-driven model in handling unseen initial conditions despite its strong performance on training conditions. NeuralPDE-Discretize initially displays potential in simulating physical vorticity patterns but struggles with numerical instabilities beyond $T = 2400$. The failure of NeuralPDE-Discretize under unseen initial conditions arises from its use of a globally trained CNN to interpolate convection fluxes across the computational domain, which does not generalized well to novel flow patterns. This model integrates trainable neural networks deeply within the numerical solver, specifically in calculating convection flux at cell faces, a process critical for capturing fluid transport and mixing. When exposed to new conditions, the model's interpolation inaccurately predicts these fluxes due to its reliance on training-specific scenarios. In a FVM setting, where accurate flux calculation is crucial for stability, such errors in flux prediction lead to numerical instability, particularly in convection-dominated chaotic flows where slight inaccuracies can escalate into significant deviations from expected physical behavior. The purely physics-based model (i.e., LES) on such coarse space–time

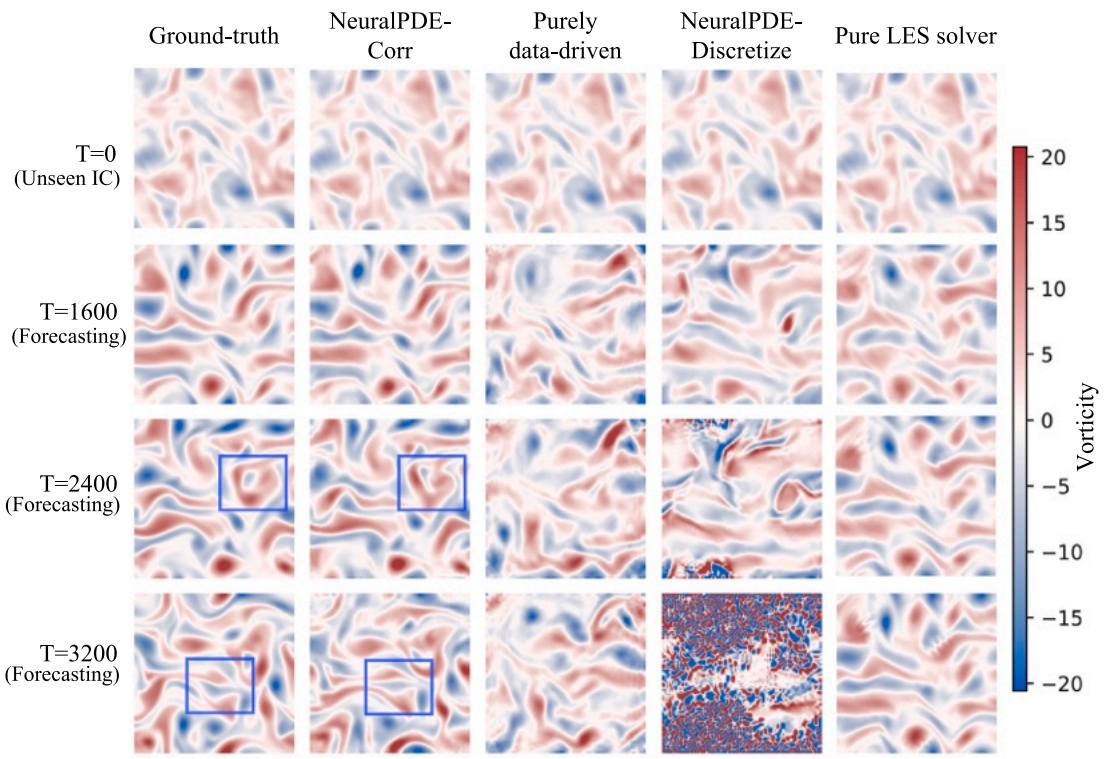

Fig. 11. Temporal progression of vorticity contours predicted by different models on a coarse mesh, starting from one randomly generated initial conditions which has not been seen during training.

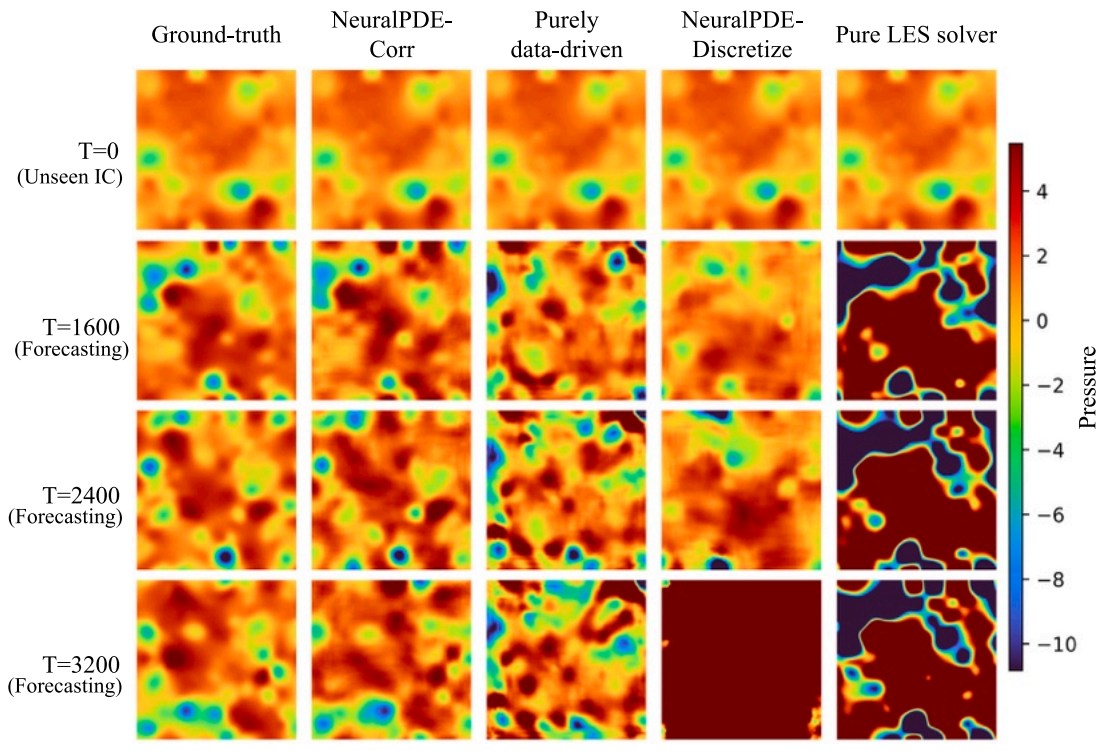

Fig. 12. Temporal progression of pressure contours predicted by different models on a coarse mesh, starting from one randomly generated initial conditions which has not been seen during training.

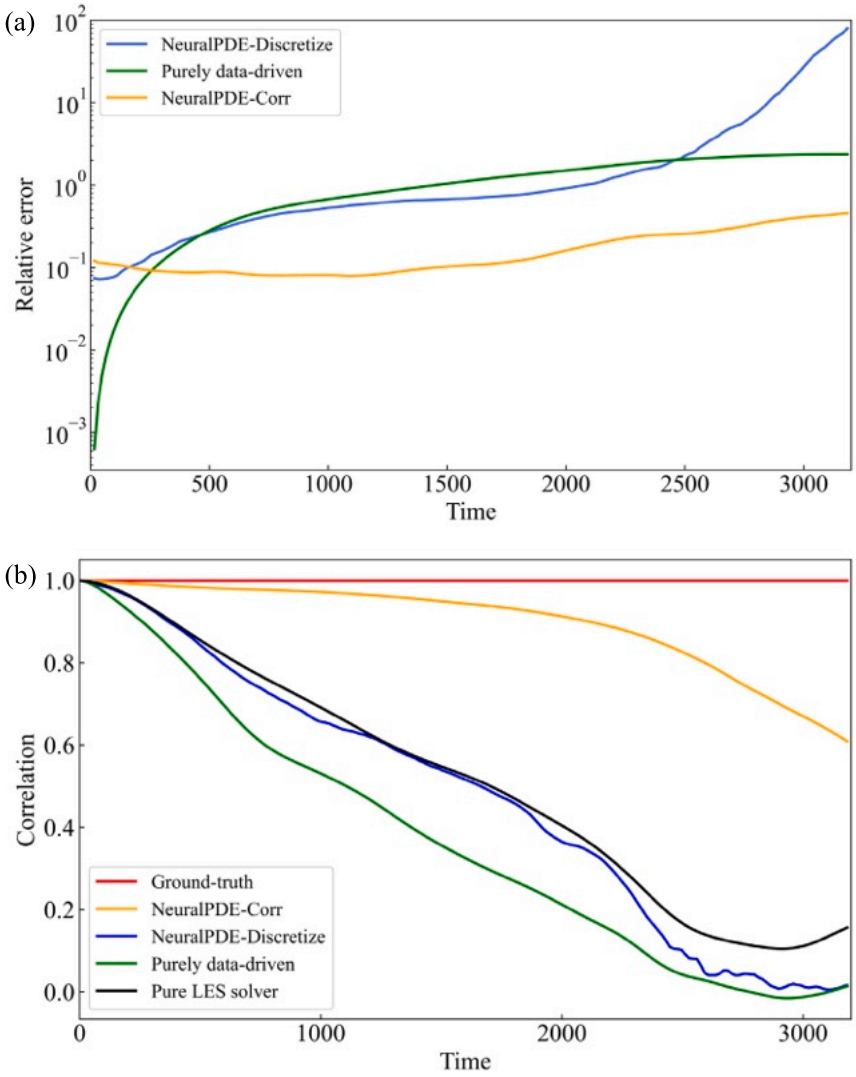

Fig. 13. Comparison of (a) relative prediction error and (b) correlation of predicted vorticity with ground truth across all models for unseen initial conditions.

resolution demonstrates the most drastic limitations, completely failing to capture the correct dynamics, particularly for the pressure. This is primarily due to insufficient grid resolution and very large timestep.

Fig. 13 further quantifies the performance of all models under unseen initial conditions through two key metrics: relative prediction error and vorticity correlation with the ground truth. Panel (a) displays the relative errors over time, where NeuralPDE-Corr consistently shows the lowest errors, underlining its superior predictive accuracy. Panel (b) depicts the correlation of predicted vorticity with ground truth, with NeuralPDE-Corr maintaining the highest correlation, suggesting that its predictions are most closely aligned with the true dynamics of turbulence. Despite initial promise, the purely data-driven model shows a marked increase in error and decline in correlation over time, reflecting its inability to generalize beyond the training conditions. Similarly, NeuralPDE-Discretize starts well but soon faces performance degradation due to its numerical stability issues. Lastly, the energy spectra comparison is presented in Fig. 14, which shows how closely each model approaches the true energy distribution across different wave numbers, particularly capturing the energy decay at higher wave numbers. The results further confirm previous observations.

The comparative analysis of models under unseen initial conditions underscores the promise of seamlessly integrating numerical PDEs into neural networks, exemplified by the hybrid NeuralPDE-Corr model. This hybrid model exhibits superior generalizability and stability over the purely data-driven or conventional numerical solvers by leveraging the strengths of both ML and physics-based approaches. However, the specific design of the hybrid architecture plays a crucial role in determining overall performance, highlighting the need for careful consideration in the configuration of such models to optimize their effectiveness.

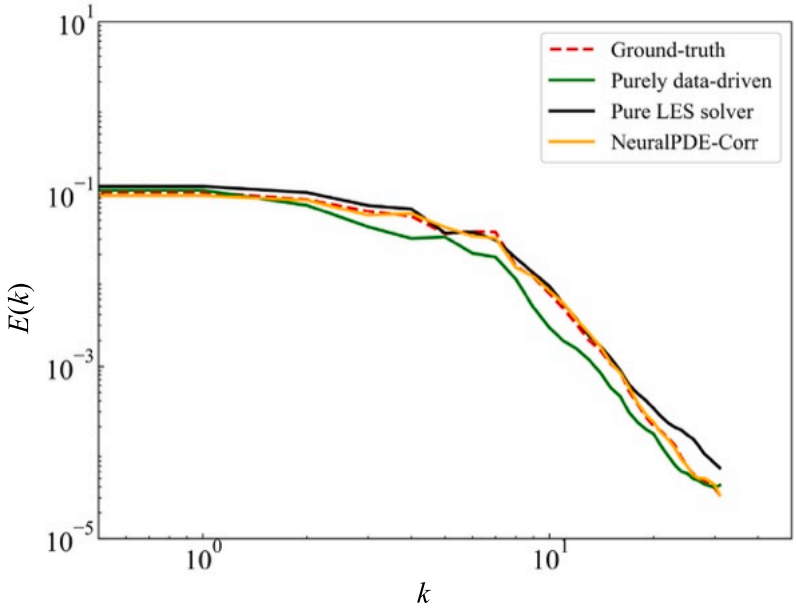

Fig. 14. Energy spectra comparison across different models, averaged over the period from $T = 2400$ to $T = 3200$ to ensure decorrelation from initial conditions. The spectrum for NeuralPDE-Discretize is excluded due to its stability issues.

3.1.3. Temporal forecasting from testing ICs with unseen forcing term

The above results demonstrate that hybrid neural solvers, particularly NeuralPDE-Corr, outperform traditional SGS closure and purely data-driven models, largely due to the preservation of physical principles inherent in numerical solvers. Apart from ICs, generalization over different forcing terms, which often drive various flow scenarios in turbulence modeling, is also of great interest. To this end, we further tested our model's performance on unseen forcing terms $\boldsymbol{f}$, setting $\boldsymbol{f} = 0$ to simulate a decaying turbulent flow, which is different from the training scenario (forced homogeneous isotropic turbulence). Notably, the forcing term was not parameterized during training. As shown in Fig. 15(a), our model, NeuralPDE-Corr, successfully captures the decaying turbulent flow patterns without retraining, thanks to the discretized physics embedded in the hybrid neural solver via differentiable programming. While some differences in flow patterns emerged after long rollout steps ($T \geq 3000$), as indicated by the correlation in Fig. 15(c), NeuralPDE-Corr still produced a reasonable and consistent energy spectrum (Fig. 15(b)) and accurately predicted pressure fields (Fig. 16). This outcome is particularly noteworthy because purely data-driven models, which do not include the forcing term as an input feature, would completely fail under such conditions. This additional experiment further demonstrates the generalizability of our proposed model, highlighting how the hybrid model successfully leverage the generalizability of physics-based solvers and the flexibility of trainable neural networks, thereby significantly improving the predictive performance with high efficiency.

3.2. Recovered high wavenumber feature by diffusion-SR model

Conditioned on the low-resolution velocity and pressure data obtained from the hybrid neural solver (NeuralPDE-Corr, 64×64), the diffusion-SR network outlined in Section 2.3 successfully restores high-wavenumber turbulence features and enhances the resolution of flow predictions to 256×256. Although constrained by current GPU memory to a super-resolution factor of $4\times$, the network has the theoretical capability to handle any higher resolutions if sufficient computational resources are available. The super-resolved velocity and pressure predictions are illustrated in Figs. B.22 and B.23. We further calculate the vorticity and energy spectrum from the super-resolved velocity fields, as depicted in Fig. 17, to show the SR performance. Overall, the diffusion-SR method not only recaptures but effectively accentuates small-scale turbulence details at higher wavenumbers. The energy spectra in Fig. 17(b) and (c) show that the energy levels ranging from 10^{-10} to 10^{-6} have been successfully restored, closely matching the energy spectra of the original high-resolution data. This is in contrast to the bicubic interpolation method, which, although simpler, introduces spurious artifacts not present in the actual turbulence data. In addition, it is important to highlight that the energy spectrum for high wavenumbers ($k > 40$) with small amplitude ($E < 10^{-6}$) is well predicted by our model, outperforming existing works [59]. This improvement stems from our spectrum-decomposed diffusion formulation, where the low-resolution and high-resolution data are generated through two separate channels. To demonstrate the efficacy, we compared our method with the vanilla denoising diffusion model (i.e., DDPM), which was trained unconditionally on the same HR dataset. As shown in Fig. 18, the vanilla diffusion-SR fails to recover the small-scale features while our spectrum-decomposed diffusion SR model significantly improved the performance, where the high-wavenumber flow features are accurately recovered even the energy levels are very low.

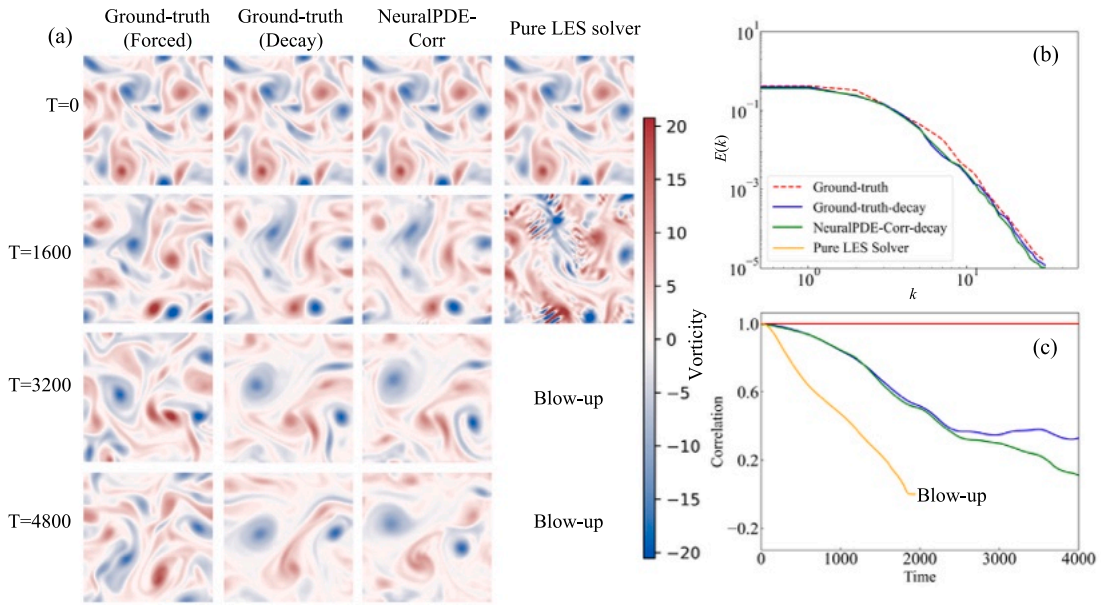

Fig. 15. Predictions for the decaying Kolmogorov flow with zero forcing terms, which is not seen during training: (a) vorticity patterns, with the forced Kolmogorov flow (training scenario) shown in the first column, highlighting the difference from the decaying case; (b) energy spectrum comparison across various models; (c) correlation comparison for various models against the forced Kolmogorov flow.

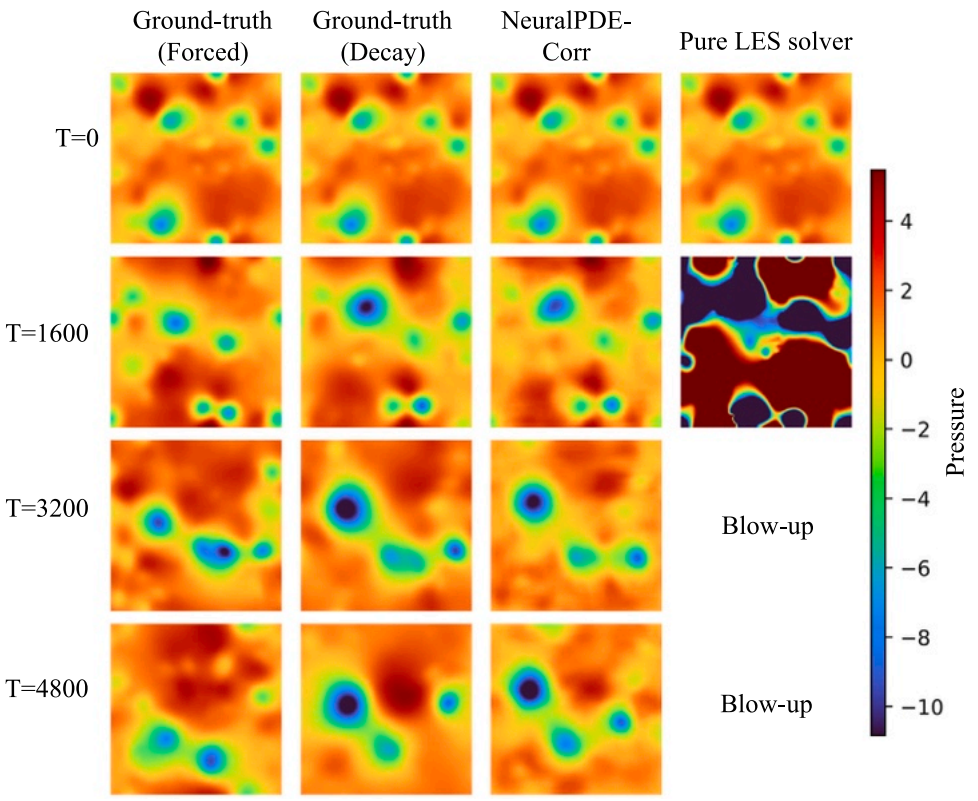

Fig. 16. Pressure predictions for the decaying Kolmogorov flow (unseen during training).

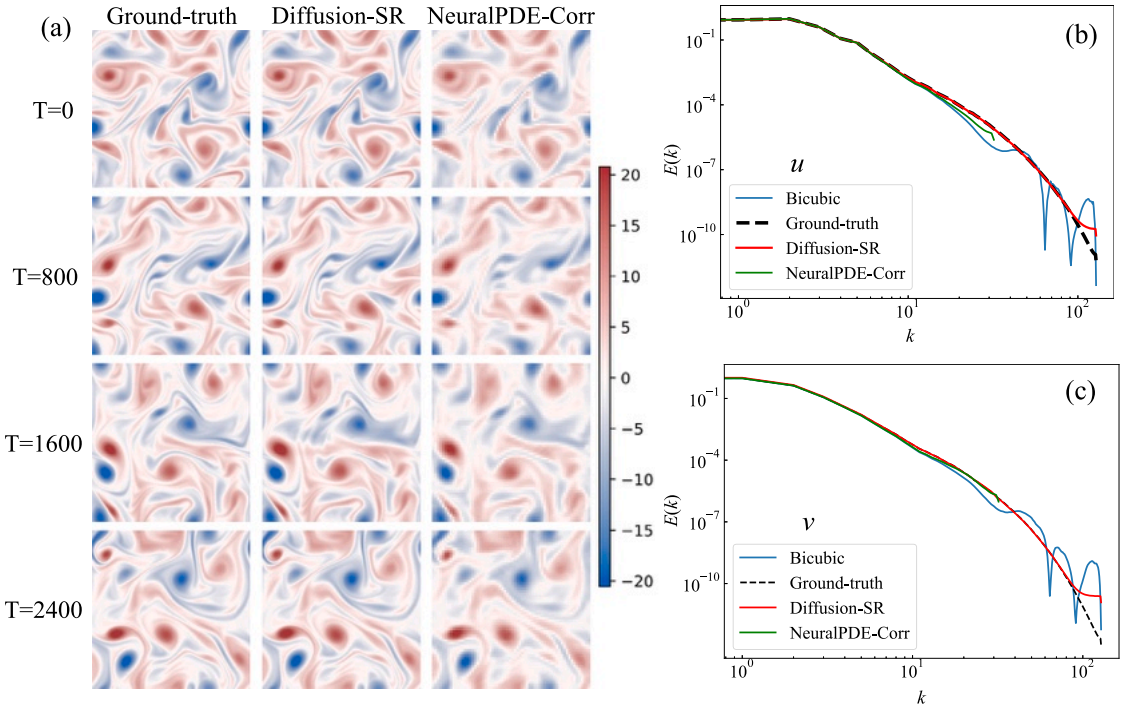

Fig. 17. Super resolution by diffusion-SR network: (a) contours of vorticity, where the resolution of PDE-Neural-Corr is 64 × 64 and the others are 256 × 256; (b) energy spectrum of $\mathbf{u}_x$; (c) energy spectrum of $\mathbf{u}_y$.

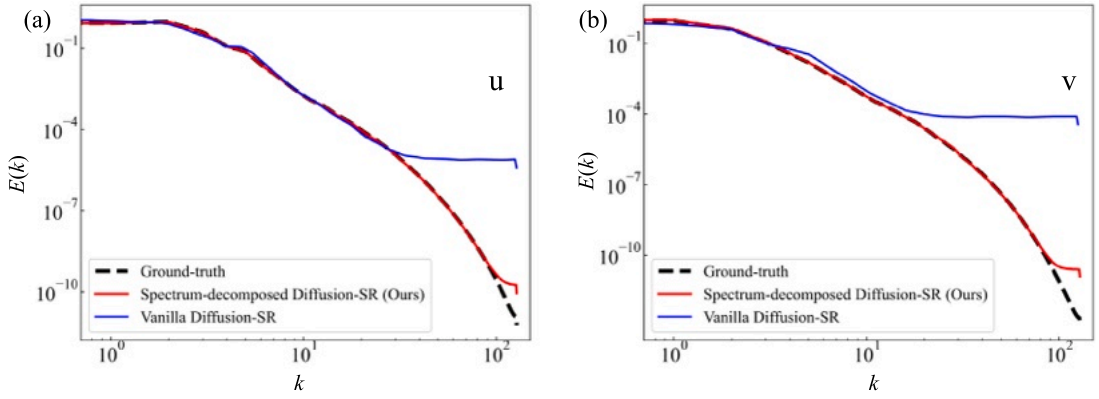

Fig. 18. Comparison of energy spectrum of the SR results of our spectrum-decomposed diffusion model and the vanilla diffusion SR model (DDPM).

In sum, we have showcased the efficacy of the concept of hybrid neural differentiable modeling that integrates physics in the form of discretized governing PDEs with the SOTA machine learning techniques via differentiable programming. The proposed modeling strategy can adeptly capture both large and small scales of turbulent dynamics, while preserving a high level of generalizability across various initial conditions.

4. Discussions

4.1. Assessment of enrichment through PDEs

In hybrid neural models, a important portion of the architecture dedicate itself to non-trainable, discretized physics, enhancing the model's generalizability compared to purely trainable neural networks. These hybrid differentiable models benefit from end-to-end training via differentiable programming, which ensure both accuracy and stability in *a posterior* evaluations. Disjointed physics and neural networks often encounter significant stability challenges across extended spatiotemporal dynamics, as evidenced in the

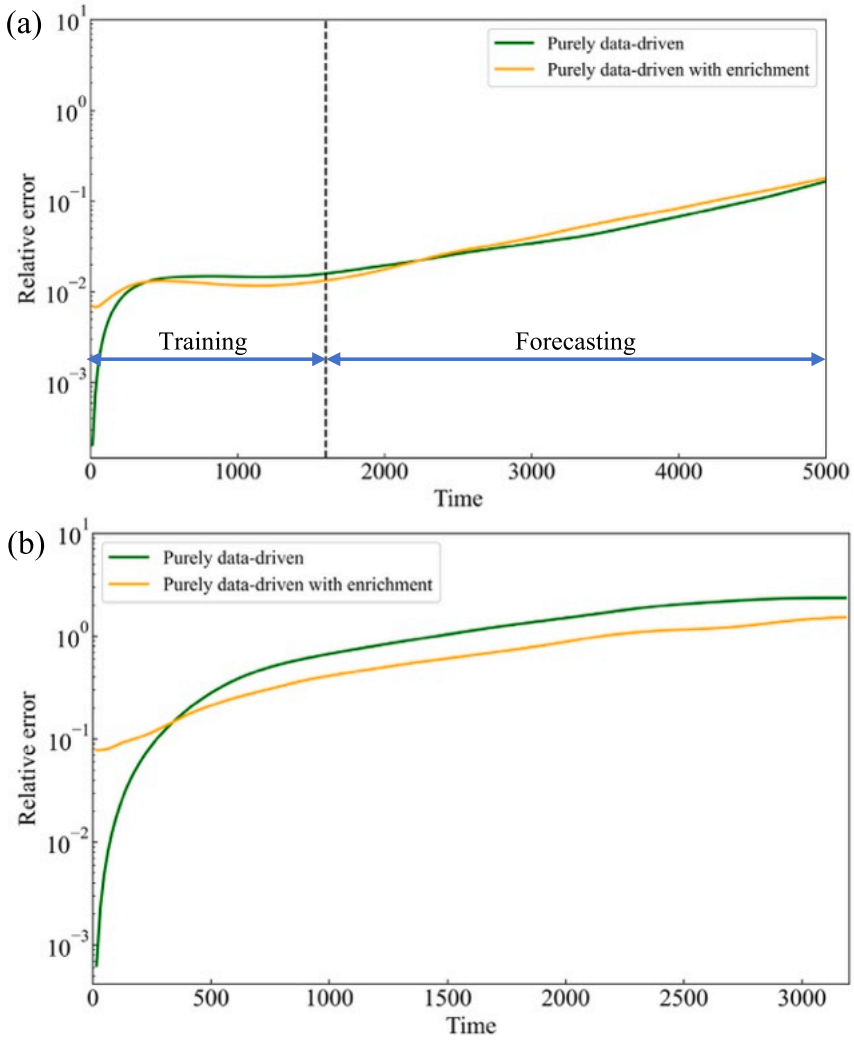

Fig. 19. Relative error comparison of purely data-driven model w and w/o PDE feature enrichment, tested on (a) training initial conditions (b) unseen initial conditions.

literature [34,47,63]. To further illustrate the effectiveness of hybrid neural models from different perspectives, an evaluation is conducted to assess the performance of purely data-driven models when they are augmented with physical features derived from the solver by input feature enrichment. This assessment will provide a comprehensive understanding of the advantages offered by hybrid neural models. For purely data-driven model, if the physics are only used to enrich the input features, the autogressive sequential net can be expressed as,

$$\boldsymbol{V}_t = \boldsymbol{V}_{t-1} + \mathrm{ConvLSTM}\Big(\boldsymbol{V}_{t-1}, \mathrm{ConvPDE}(\boldsymbol{V}_{t-1}), \boldsymbol{H}_{t-1}; \boldsymbol{\theta}_{nn}\Big), \tag{17}$$

The neural architecture and hyperparameters is kept the identical to those of purely data-driven models. The relative errors for both training and unseen ICs are presented in Fig. 19.

In the context of training initial conditions, the relative error trends of both purely data-driven models—with and without physics enrichment—are nearly identical, effectively capturing the spatiotemporal dynamics of the true data. However, when subjected to unseen initial conditions, the model enriched with physics-derived features demonstrates a slight reduction in error, highlighting its enhanced generalizability, although the improvement is modest. In comparison with the hybrid neural models, particularly the NeuralPDE-Corr variant, the error remains an order of magnitude higher even with physics enrichment. The physics incorporated serves to augment the input features of the purely data-driven model, but the overall structure remains predominantly black-box in nature. This underscores the superior performance and generalizability of the proposed hybrid neural models, which seamlessly integrate discretized physics with state-of-the-art ML techniques through differentiable programming, offering substantial advancements in both predictive accuracy and generalizability.

Table 1
The inference cost of hybrid neural solver, compared with DNS and purely data-driven model. Note that all inference is performed on an RTX A6000 GPU.

DNS	Hybrid neural solver	Purely data-driven model
52.74	2.03	2.31

4.2. Offline training and online inference cost analysis

We conducted a comparative evaluation of the computational cost of different models. Table 1 summarize the cost for simulating 2D Kolmogorov turbulence over an eight-second duration during the inference phase. Notably, our trained hybrid neural model achieved a substantial speedup, operating 26 times faster than DNS, which is itself an optimized GPU-accelerated, fully vectorized CFD solver in [42]. This speed advantage would likely increase further when compared to established CPU-based CFD solvers like OpenFOAM, as reported in applications of fluid–structure interaction applications [47].

The training cost for hybrid models, while comparable to running a high-fidelity DNS simulation on OpenFOAM using 8 CPUs, is a one-time offline investment. Subsequent uses of the trained model for predictions incur minimal costs, making this approach highly advantageous for applications requiring extended temporal forecasting and repetitive analysis across various conditions, such as different Reynolds numbers and initial states. It should be noted that the training cost associated with the hybrid neural model is higher than that of the purely data-driven model, as reported in our previous work [47]. This additional training overhead primarily arises from two factors. Firstly, the hybrid model demands more memory and training time for back-propagation due to the necessity of tracing gradients through complex, differentiable physics modules. Secondly, to address numerical instabilities inherent to the training process, we may implement transfer learning strategies that progressively increase the length of autoregressive steps during end-to-end training. While these methods elevate the offline training costs, they are justified by the significant improvements in inference performance and generalizability demonstrated in Section 3.1.2.

4.3. Novelty, limitation and future prospects of current framework

Integrating data-driven methods with discretized physics within a differentiable framework offers a promising direction for turbulence modeling. Similar concepts have been previously explored [42,43,55], primarily focusing on capturing large-scale structures and reducing computational costs on coarse grids. Our work builds upon these foundational studies, setting itself apart through novel architectural designs, refined training strategies, and comprehensive evaluations under challenging conditions, leading to substantial improvements in model performance, stability, and generalizability. Specifically, our unique two-stage framework effectively addresses the multiscale nature of turbulence, producing high-fidelity flow representations that capture the complete spectrum of turbulence dynamics. The physics-based hybrid neural solver predicts large-scale flow dynamics that are more generalizable across different flow geometries and boundary conditions, while the diffusion model accurately generates the small-scale turbulence details that are less dependent on specific geometries. Bayesian posterior sampling serves as a critical link between these components, allowing the integration of large-scale predictions with fine-scale generation to produce high-resolution, generalizable results efficiently. This approach not only enhances the model's predictive capabilities but also ensures robustness across various testing scenarios, making it a foundational step towards a fully integrated, two-way coupled model where small-scale features dynamically feedback into the hybrid solver to further refine predictions.

However, there are several limitations associated with the current framework that need to be addressed in future work. One limitation is the one-way coupling between the hybrid neural solver and the diffusion model; the generated small scales do not feed back into the neural solver to refine large-scale predictions, which could enhance accuracy and stability. Developing a fully integrated, two-way coupled framework is a critical next step. Additionally, the current focus on 2D turbulence, although valuable for demonstrating the framework's capabilities, does not fully capture the complexities of 3D turbulence seen in practical engineering applications. Extending this approach to three dimensions will involve addressing increased computational demands and additional modeling complexities inherent to 3D turbulence. Furthermore, the model's stability also depends on specific numerical schemes, and while it performs well under certain coarse-grid conditions, further exploration of stability-enhancing numerical schemes is essential, especially for highly unstructured or coarser grids. Lastly, while the diffusion model effectively generates small-scale features, its scalability to high-resolution spatiotemporal coherent structures remains a challenge. Exploring advanced conditional latent generative techniques could further improve the efficiency and quality of small-scale 4D turbulence generation.

5. Conclusions

In this paper, we introduced an innovative neural differentiable modeling framework aimed at enhancing turbulence simulation capabilities. Within this concept, we developed a hybrid differentiable neural solver configured on a coarse grid, capturing large-scale turbulent phenomena effectively. Subsequently, we applied a Bayesian conditional diffusion model to generate small-scale turbulence details, conditioned on the large-scale flow predictions. We engineered two unique hybrid architectural prototypes and evaluated their performance in simulating 2D Kolmogorov turbulence, comparing them against traditional purely data-driven models and physics-based models with conventional SGS closures.

Our comparative analysis revealed that the proposed hybrid neural models, comprehensively trained via differentiable programming, outperform all baseline models in accuracy and generalizability. The first architectural design, which appends additional

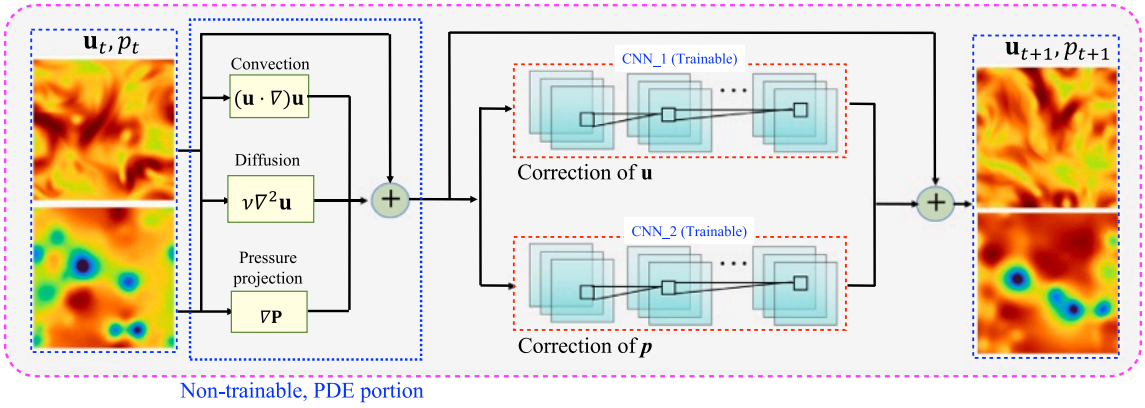

Fig. A.20. Details of NeuralPDE-Corr in one learning step.

trainable neural networks to the numerical PDE layers, and the second, which deeply merges neural networks with PDEs to learn flux interpolation schemes, both demonstrated the ability to predict turbulence accurately over extended periods, demonstrating their superiority over the baseline models. However, the second design displayed significant numerical instability when applied to unseen initial conditions, highlighting challenges in its generalizability. To mitigate these issues, further constraints and scaling techniques are needed. Furthermore, the proposed diffusion-based super-resolution network successfully reconstructs high-wavenumber turbulence features from low-resolution outputs generated by the coarse hybrid neural solver, effectively demonstrating the framework's capacity to simulate high-fidelity turbulence across all scales. Additionally, various aspects of the model construction and training have been analyzed and discussed, providing valuable insights into optimizing training efficiency and predictive accuracy. Our findings underscore the potential of hybrid neural differentiable modeling in simulating turbulence, showcasing their superior performance and architectural flexibility.

Looking forward, the insights gained from this study pave the way for further exploration into seamlessly integrating cutting-edge ML with classic CFD techniques via differentiable programming, aiming to advance the capabilities of next-generation data-augmented neural solvers. Future research will focus on expanding these methodologies to more complex applications, illustrating the robustness and adaptability of the neural differentiable framework in tackling real-world engineering challenges.

CRediT authorship contribution statement

Xiantao Fan: Writing – original draft, Software, Methodology, Investigation, Formal analysis, Conceptualization. **Deepak Akhare:** Writing – original draft, Methodology, Investigation. **Jian-Xun Wang:** Writing – review & editing, Supervision, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors would like to acknowledge the funds from Office of Naval Research, United States under award numbers N00014-23-1-2071 and National Science Foundation, United States under award numbers OAC-2047127. XF would also like to acknowledge the fellowship provided by the Environmental Change Initiative and Center for Sustainable Energy at University of Notre Dame.

Appendix A. Details of each hybrid neural models

See Figs. A.20 and A.21.

Appendix B. Super resolution of velocity and pressure

See Figs. B.22 and B.23.

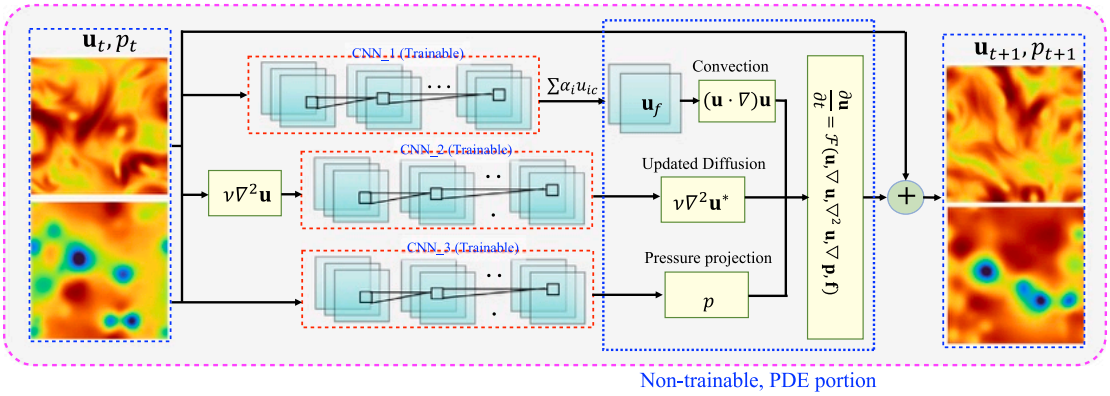

Fig. A.21. Details of NeuralPDE-Discretize in one learning step.

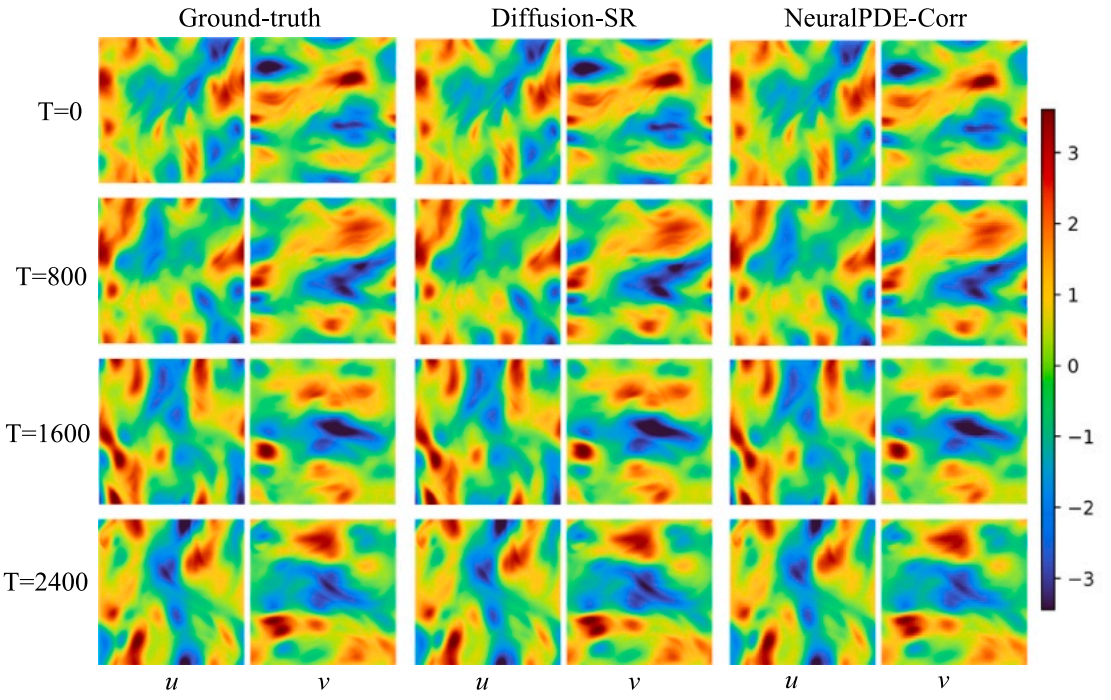

Fig. B.22. Super resolution of velocity by diffusion-SR network.

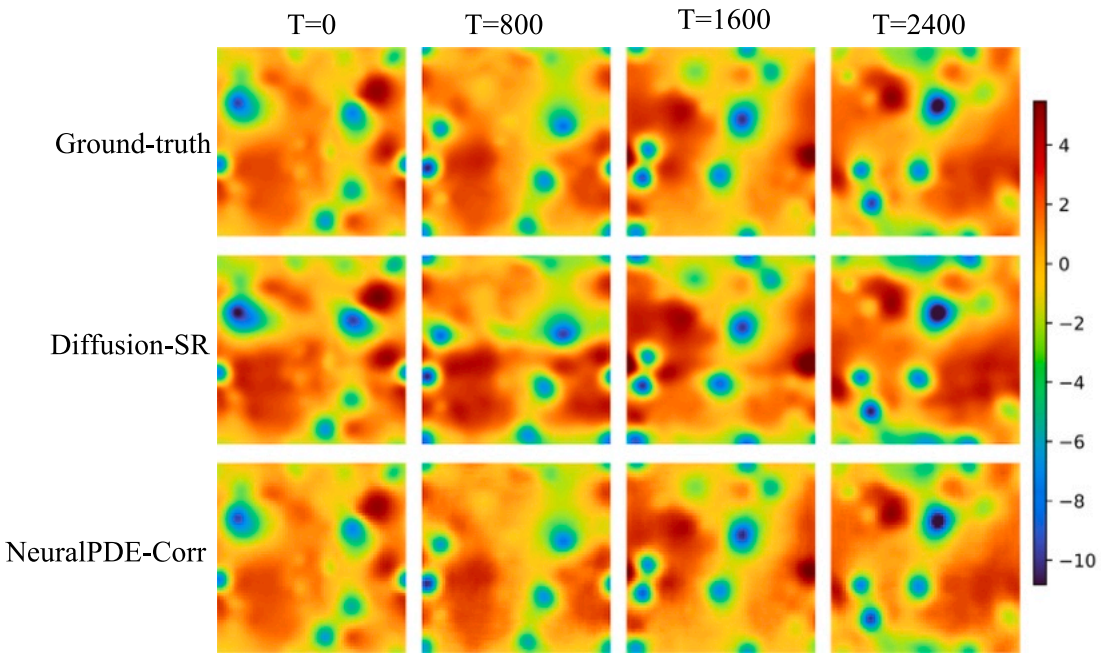

Fig. B.23. Super resolution of pressure by diffusion-SR network.

Data availability

Data will be made available on request.

References

[1] P.A. Durbin, Some recent developments in turbulence closure modeling, Annu. Rev. Fluid Mech. 50 (2018) 77–103.
[2] M. Morimoto, K. Fukami, K. Zhang, A.G. Nair, K. Fukagata, Convolutional neural networks for fluid flow analysis: toward effective metamodeling and low dimensionalization, Theor. Comput. Fluid Dyn. 35 (5) (2021) 633–658.
[3] L. Guastoni, A. Güemes, A. Ianiro, S. Discetti, P. Schlatter, H. Azizpour, R. Vinuesa, Convolutional-network models to predict wall-bounded turbulence from wall quantities, J. Fluid Mech. 928 (2021) A27.
[4] T. Pfaff, M. Fortunato, A. Sanchez-Gonzalez, P.W. Battaglia, Learning mesh-based simulation with graph networks, in: International Conference on Learning Representations, 2020.
[5] X. Han, H. Gao, T. Pfaff, J.-X. Wang, L. Liu, Predicting physics in mesh-reduced space with temporal attention, in: International Conference on Learning Representations, 2022.
[6] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, A. Anandkumar, Fourier neural operator for parametric partial differential equations, 2020, arXiv preprint arXiv:2010.08895.
[7] Z. Li, W. Peng, Z. Yuan, J. Wang, Long-term predictions of turbulence by implicit U-Net enhanced Fourier neural operator, Phys. Fluids 35 (7) (2023).
[8] L. Lu, P. Jin, G. Pang, Z. Zhang, G.E. Karniadakis, Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators, Nat. Mach. Intell. 3 (3) (2021) 218–229.
[9] N. Demo, M. Tezzele, G. Rozza, A DeepONet multi-fidelity approach for residual learning in reduced order modeling, 2023, arXiv preprint arXiv:2302.12682.
[10] G.E. Karniadakis, I.G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, L. Yang, Physics-informed machine learning, Nat. Rev. Phys. 3 (6) (2021) 422–440.
[11] S. Cai, Z. Mao, Z. Wang, M. Yin, G.E. Karniadakis, Physics-informed neural networks (PINNs) for fluid mechanics: A review, Acta Mech. Sin. 37 (12) (2021) 1727–1738.
[12] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707.
[13] L. Sun, H. Gao, S. Pan, J.-X. Wang, Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data, Comput. Methods Appl. Mech. Engrg. 361 (2020) 112732.
[14] L. Sun, D.Z. Huang, H. Sun, J.-X. Wang, Bayesian spline learning for equation discovery of nonlinear dynamics with quantified uncertainty, in: NeurIPS, PMLR, 2022.
[15] L. Sun, J.-X. Wang, Physics-constrained bayesian neural network for fluid flow reconstruction with sparse and noisy data, Theor. Appl. Mech. Lett. 10 (3) (2020) 161–169.
[16] H. Gao, L. Sun, J.-X. Wang, PhyGeoNet: physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state PDEs on irregular domain, J. Comput. Phys. 428 (2021) 110079.
[17] H. Gao, M.J. Zahr, J.-X. Wang, Physics-informed graph neural Galerkin networks: A unified framework for solving PDE-governed forward and inverse problems, Comput. Methods Appl. Mech. Engrg. 390 (2022) 114502.
[18] P. Ren, C. Rao, Y. Liu, J.-X. Wang, H. Sun, PhyCRNet: Physics-informed convolutional-recurrent network for solving spatiotemporal PDEs, Comput. Methods Appl. Mech. Engrg. 389 (2022) 114399.

[19] P. Ren, C. Rao, S. Chen, J.-X. Wang, H. Sun, Y. Liu, SeismicNet: Physics-informed neural networks for seismic wave modeling in semi-infinite domain, Comput. Phys. Comm. 295 (2024) 109010.
[20] A. Arzani, J.-X. Wang, R.M. D'Souza, Uncovering near-wall blood flow from sparse data with physics-informed neural networks, Phys. Fluids 33 (7) (2021) 071905, http://dx.doi.org/10.1063/5.0055600.
[21] M. Movahhedi, X.-Y. Liu, B. Geng, C. Elemans, Q. Xue, J.-X. Wang, X. Zheng, Predicting 3D soft tissue dynamics from 2D imaging using physics informed neural networks, Commun. Biol. 6 (1) (2023) 541.
[22] R. Li, J.-X. Wang, E. Lee, T. Luo, Physics-informed deep learning for solving phonon Boltzmann transport equation with large temperature non-equilibrium, Npj Comput. Mater. 8 (2022) 19.
[23] E. Kharazmi, D. Fan, Z. Wang, M.S. Triantafyllou, Inferring vortex induced vibrations of flexible cylinders using physics-informed neural networks, J. Fluids Struct. 107 (2021) 103367.
[24] A. Krishnapriyan, A. Gholami, S. Zhe, R. Kirby, M.W. Mahoney, Characterizing possible failure modes in physics-informed neural networks, Adv. Neural Inf. Process. Syst. 34 (2021) 26548–26560.
[25] S. Wang, X. Yu, P. Perdikaris, When and why PINNs fail to train: A neural tangent kernel perspective, J. Comput. Phys. 449 (2022) 110768.
[26] J. Ling, A. Kurzawski, J. Templeton, Reynolds averaged turbulence modelling using deep neural networks with embedded invariance, J. Fluid Mech. 807 (2016) 155–166.
[27] J. Weatheritt, R.D. Sandberg, Hybrid Reynolds-averaged/large-eddy simulation methodology from symbolic regression: formulation and application, AIAA J. 55 (11) (2017) 3734–3746.
[28] J.-X. Wang, J.-L. Wu, H. Xiao, Physics-informed machine learning approach for reconstructing Reynolds stress modeling discrepancies based on DNS data, Phys. Rev. Fluids 2 (3) (2017) 034603.
[29] J.-X. Wang, J. Huang, L. Duan, H. Xiao, Prediction of Reynolds stresses in high-mach-number turbulent boundary layers using physics-informed machine learning, Theor. Comput. Fluid Dyn. 33 (1) (2019) 1–19.
[30] R. Maulik, O. San, A. Rasheed, P. Vedula, Subgrid modelling for two-dimensional turbulence using neural networks, J. Fluid Mech. 858 (2019) 122–144.
[31] X. Yang, S. Zafar, J.-X. Wang, H. Xiao, Predictive large-eddy-simulation wall modeling via physics-informed neural networks, Phys. Rev. Fluids 4 (3) (2019) 034602.
[32] Z. Zhou, G. He, X. Yang, Wall model based on neural networks for LES of turbulent flows over periodic hills, Phys. Rev. Fluids 6 (5) (2021) 054610.
[33] A. Lozano-Durán, H.J. Bae, Self-critical machine-learning wall-modeled LES for external aerodynamics, 2020, arXiv preprint arXiv:2012.10005.
[34] J. Wu, H. Xiao, R. Sun, Q. Wang, Reynolds-averaged Navier–Stokes equations with explicit data-driven Reynolds stress closure can be ill-conditioned, J. Fluid Mech. 869 (2019) 553–586.
[35] Y. Guan, A. Chattopadhyay, A. Subel, P. Hassanzadeh, Stable a posteriori LES of 2D turbulence using convolutional neural networks: Backscattering analysis and generalization to higher re via transfer learning, J. Comput. Phys. 458 (2022) 111090.
[36] R. McConkey, E. Yee, F.-S. Lien, A curated dataset for data-driven turbulence modelling, Sci. Data 8 (1) (2021) 255.
[37] X.-L. Zhang, H. Xiao, X. Luo, G. He, Ensemble Kalman method for learning turbulence models from indirect observation data, J. Fluid Mech. 949 (2022) A26.
[38] A.G. Baydin, B.A. Pearlmutter, A.A. Radul, J.M. Siskind, Automatic differentiation in machine learning: a survey, J. Mach. Learn. Res. 18 (2018) 1–43.
[39] A. Mensch, M. Blondel, Differentiable dynamic programming for structured prediction and attention, in: International Conference on Machine Learning, PMLR, 2018, pp. 3462–3471.
[40] M. Innes, A. Edelman, K. Fischer, C. Rackauckas, E. Saba, V.B. Shah, W. Tebbutt, A differentiable programming system to bridge machine learning and scientific computing, 2019, arXiv preprint arXiv:1907.07587.
[41] F.D.A. Belbute-Peres, T. Economon, Z. Kolter, Combining differentiable PDE solvers and graph neural networks for fluid flow prediction, in: International Conference on Machine Learning, PMLR, 2020, pp. 2402–2411.
[42] D. Kochkov, J.A. Smith, A. Alieva, Q. Wang, M.P. Brenner, S. Hoyer, Machine learning–accelerated computational fluid dynamics, Proc. Natl. Acad. Sci. 118 (21) (2021) e2101784118.
[43] B. List, L.-W. Chen, N. Thuerey, Learned turbulence modelling with differentiable fluid solvers: physics-based loss functions and optimisation horizons, J. Fluid Mech. 949 (2022) A25.
[44] D. Akhare, T. Luo, J.-X. Wang, Physics-integrated neural differentiable (PiNDiff) model for composites manufacturing, Comput. Methods Appl. Mech. Engrg. 406 (2023) 115902.
[45] D. Akhare, T. Luo, J.-X. Wang, DiffHybrid-UQ: Uncertainty quantification for differentiable hybrid neural modeling, 2023, arXiv preprint arXiv:2401.00161.
[46] D. Akhare, Z. Chen, R. Gulotty, T. Luo, J.-X. Wang, Probabilistic physics-integrated neural differentiable modeling for isothermal chemical vapor infiltration process, npj Comput. Mater. 10 (1) (2024) 120.
[47] X. Fan, J.-X. Wang, Differentiable hybrid neural modeling for fluid-structure interaction, J. Comput. Phys. 496 (2024) 112584.
[48] X.-Y. Liu, M. Zhu, L. Lu, H. Sun, J.-X. Wang, Multi-resolution partial differential equations preserved learning framework for spatiotemporal dynamics, Commun. Phys. 7 (1) (2024) 31.
[49] J.F. MacArt, J. Sirignano, J.B. Freund, Embedded training of neural-network subgrid-scale turbulence models, Phys. Rev. Fluids 6 (5) (2021) 050502.
[50] C.A.M. Ströfer, H. Xiao, End-to-end differentiable learning of turbulence models from indirect observations, Theor. Appl. Mech. Lett. 11 (4) (2021) 100280.
[51] V. Shankar, V. Puri, R. Balakrishnan, R. Maulik, V. Viswanathan, Differentiable physics-enabled closure modeling for Burgers' turbulence, Mach. Learn.: Sci. Technol. (2023).
[52] V. Shankar, R. Maulik, V. Viswanathan, Differentiable turbulence II, 2023, arXiv preprint arXiv:2307.13533.
[53] W.D. McComb, Homogeneous, Isotropic Turbulence: Phenomenology, Renormalization and Statistical Closures, vol. 162, OUP Oxford, 2014.
[54] G.J. Chandler, R.R. Kerswell, Invariant recurrent solutions embedded in a turbulent two-dimensional Kolmogorov flow, J. Fluid Mech. 722 (2013) 554–595.
[55] Y. Bar-Sinai, S. Hoyer, J. Hickey, M.P. Brenner, Learning data-driven discretizations for partial differential equations, Proc. Natl. Acad. Sci. 116 (31) (2019) 15344–15349.
[56] L. Zhu, W. Zhang, J. Kou, Y. Liu, Machine learning methods for turbulence modeling in subsonic flows around airfoils, Phys. Fluids 31 (1) (2019) 015105.
[57] H. Gao, X. Han, X. Fan, L. Sun, L.-P. Liu, L. Duan, J.-X. Wang, Bayesian conditional diffusion models for versatile spatiotemporal turbulence generation, Comput. Methods Appl. Mech. Engrg. 427 (2024) 117023.
[58] P. Du, M.H. Parikh, X. Fan, X.-Y. Liu, J.-X. Wang, CoNFiLD: Conditional neural field latent diffusion model generating spatiotemporal turbulence, 2024, arXiv preprint arXiv:2403.05940.
[59] D. Shu, Z. Li, A.B. Farimani, A physics-informed diffusion model for high-fidelity flow field reconstruction, J. Comput. Phys. 478 (2023) 111972.
[60] P. Dhariwal, A. Nichol, Diffusion models beat gans on image synthesis, Adv. Neural Inf. Process. Syst. 34 (2021) 8780–8794.
[61] J. Ho, A. Jain, P. Abbeel, Denoising diffusion probabilistic models, Adv. Neural Inf. Process. Syst. 33 (2020) 6840–6851.
[62] H. Chung, J. Kim, M.T. Mccann, M.L. Klasky, J.C. Ye, Diffusion posterior sampling for general noisy inverse problems, 2022, arXiv preprint arXiv:2209.14687.
[63] K. Um, R. Brand, Y.R. Fei, P. Holl, N. Thuerey, Solver-in-the-loop: Learning from differentiable physics to interact with iterative pde-solvers, Adv. Neural Inf. Process. Syst. 33 (2020) 6111–6122.
[64] H. Gao, J.-X. Wang, A Bi-fidelity ensemble kalman method for PDE-constrained inverse problems in computational mechanics, Comput. Mech. 67 (4) (2021) 1115–1131.