RESEARCH ARTICLE | JUNE 26 2024

A transformer-based neural operator for large-eddy simulation of turbulence ⊘

Zhijie Li (李志杰) ⓘ ; Tianyuan Liu (刘天源) ✉ ; Wenhui Peng (彭文辉) ⓘ ; Zelong Yuan (袁泽龙) ⓘ ;
Jianchun Wang (王建春) ✉ ⓘ

Check for updates

View Online   Export Citation

Articles You May Be Interested In

Long-term predictions of turbulence by implicit U-Net enhanced Fourier neural operator
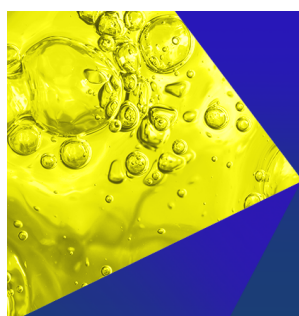
Physics of Fluids (July 2023)

Fourier neural operator for large eddy simulation of compressible Rayleigh–Taylor turbulence

Physics of Fluids (July 2024)

A symbolic regression-based implicit algebraic stress turbulence model: Incorporating the production of Reynolds stress anisotropy tensor

Physics of Fluids (September 2025)

28 November 2025 06:44:33

A transformer-based neural operator for large-eddy simulation of turbulence

Zhijie Li (李志杰),[1] Tianyuan Liu (刘天源),[2,a] Wenhui Peng (彭文辉),[3] Zelong Yuan (袁泽龙),[4] and Jianchun Wang (王建春)[1,a]

AFFILIATIONS

[1]Department of Mechanics and Aerospace Engineering, Southern University of Science and Technology, Shenzhen 518055, China
[2]College of Engineering, Peking University, Beijing 100091, China
[3]Department of Applied Mathematics, The Hong Kong Polytechnic University, Hung Hom, Kowloon, 999077, Hong Kong
[4]Harbin Engineering University Qingdao Innovation and Development Base, Qingdao 266000, China

[a]Authors to whom correspondence should be addressed: tianyuan@pku.edu.cn and wangjc@sustech.edu.cn

ABSTRACT

Predicting the large-scale dynamics of three-dimensional (3D) turbulence is challenging for machine learning approaches. This paper introduces a transformer-based neural operator (TNO) to achieve precise and efficient predictions in the large-eddy simulation (LES) of 3D turbulence. The performance of the proposed TNO model is systematically tested and compared with LES using classical sub-grid scale models, including the dynamic Smagorinsky model (DSM) and the dynamic mixed model (DMM), as well as the original Fourier neural operator (FNO) model, in homogeneous isotropic turbulence (HIT) and free-shear turbulent mixing layer. The numerical simulations comprehensively evaluate the performance of these models on a variety of flow statistics, including the velocity spectrum, the probability density functions (PDFs) of vorticity, the PDFs of velocity increments, the evolution of turbulent kinetic energy, and the iso-surface of the Q-criterion. The results indicate that the accuracy of the TNO model is comparable to the LES with DSM model and outperforms the FNO model and LES using DMM in HIT. In the free-shear turbulence, the TNO model exhibits superior accuracy compared to other models. Moreover, the TNO model has fewer parameters than the FNO model and enables long-term stable predictions, which the FNO model cannot achieve. The well-trained TNO model is significantly faster than traditional LES with DSM and DMM models and can be generalized to higher Taylor–Reynolds number cases, indicating its strong potential for 3D nonlinear engineering applications.

I. INTRODUCTION

Large-eddy simulation (LES) has gained increasing attention as a valuable tool for turbulent flow prediction, thanks to its balance between computational accuracy and efficiency. LES is renowned for its capability to simulate the evolution of large-scale structures and model sub-grid scale (SGS) motions.[1–5] LES can model complex flows where Reynolds-averaged Navier–Stokes (RANS) is not successful, showing superior accuracy in such cases. Additionally, it demonstrates superior computational efficiency compared to direct numerical simulation (DNS).[6–10] Properly addressing the unclosed sub-grid scale (SGS) terms is crucial for obtaining accurate results in LES. Since the pioneering work, various SGS models have been proposed,[11–13] including the dynamic Smagorinsky model (DSM),[3,14,15] the dynamic mixed model (DMM),[16,17] the velocity gradient model (VGM),[18] and the approximate deconvolution model (ADM).[19,20]

Recently, neural networks (NNs) have become increasingly popular in computational fluid dynamics (CFD).[21–23] Deep neural networks have shown remarkable capabilities in approximating highly nonlinear functions, making them particularly suitable for modeling complex phenomena.[24] Well-trained "black-box" neural network models can quickly make inferences on modern computers, surpassing the efficiency of traditional CFD methods. Various techniques based on NNs have been developed to improve the simulation of turbulence in RANS and LES methods.[19,20,25–32] Zhou *et al.* employed an artificial neural network to create a novel SGS model for LES of isotropic turbulent flows.[30] Beck *et al.* introduced convolutional neural networks (CNNs) and residual neural networks (RNNs) to develop precise SGS models for LES.[26] Guan *et al.* employed transfer learning and a deep convolutional neural network to enhance the precision and stability of turbulence prediction.[33] Han *et al.* introduced a hybrid deep neural network model to capture the spatial-temporal features of the unsteady

flow fields.[34] Furthermore, several researchers have investigated the integration of supplementary physical knowledge into deep learning.[35–39] Raissi *et al.* proposed a physics-informed neural network (PINN) to solve general nonlinear partial differential equations (PDEs).[40] Chen *et al.* introduced a method that combines theory-guided principles with hard constraint projection. This technique transforms governing equations into a more manageable form through discretization and then applies hard constraint optimization via projection within a localized patch.[41] Jin *et al.* developed Navier–Stokes flow nets (NSFnets) by integrating governing equations, initial conditions, and boundary conditions into the loss function.[42]

Although previous neural network architectures have shown proficiency in learning mappings within finite-dimensional Euclidean spaces, their ability to generalize across various initial or boundary conditions is constrained.[40,43–47] Li *et al.* recently proposed a novel framework known as the Fourier neural operator (FNO), which enables efficient learning of mappings between infinite-dimensional spaces using input–output pairs.[48] The FNO model has demonstrated superior performance in predicting two-dimensional (2D) turbulence compared to current state-of-the-art models including U-Net,[49] TF-Net,[50] and ResNet.[51] Wen *et al.* introduced an enhanced version of the FNO model called U-FNO, which combines the U-Net architecture with FNO for improved accuracy and efficiency in solving multiphase flow problems.[52] Choubineh *et al.* evaluated the FNO model to predict pressure distribution over a small, specified shape-data problem.[53] Peng *et al.* presented an improved FNO model coupled with attention, which showed highly effective in accurately reconstructing the statistical properties and instantaneous flow structures of 2D turbulence, especially at high Reynolds numbers.[54] While there has been a considerable amount of research on FNO-based models,[55–64] most of these works have concentrated on 2D problems. Modeling three-dimensional (3D) turbulence using deep neural networks poses a greater challenge due to the substantial increase in data size and parameters compared to 2D problems.[65] Training models with a large number of parameters can be computationally expensive and memory-demanding, presenting difficulties given hardware limitations.

Mohan *et al.* proposed two reduced models for 3D homogeneous isotropic turbulence (HIT) and scalar turbulence. These models were developed using deep learning techniques, specifically the convolutional generative adversarial network (C-GAN) and compressed convolutional long-short-term-memory (CC-LSTM) network.[66] Nakamura *et al.* integrated a 3D convolutional neural network autoencoder (CNN-AE) with a long short-term memory (LSTM) network to predict the 3D channel flow.[67] Li *et al.* applied the FNO approach for the prediction of large-scale dynamics of 3D homogeneous isotropic turbulence.[68] Peng *et al.* introduced a linear attention coupled Fourier neural operator (LAFNO) for simulating 3D isotropic turbulence and free-shear turbulence.[69] Li *et al.* proposed an implicit U-Net enhanced Fourier neural operator (IU-FNO) for the long-term prediction of large-scale dynamics of turbulence.[70] Li *et al.* proposed a geometry-informed neural operator (GINO) to efficiently learn the solution operator of large-scale PDEs with varying geometries.[71]

Due to the outstanding performance, the transformer model has received significant attention and application in natural language processing.[61,72–75] In recent years, an increasing number of researchers have started to adopt this powerful model in the domain of CFD to explore its potential.[76–81] Patil *et al.* proposed a convolutional encoder–decoder-based transformer model for autoregressive training on spatiotemporal data of turbulent flows.[82] Dang *et al.* introduced a learned simulator based on the transformer architecture, namely, turbulence neural transformer (TNT), to predict turbulent dynamics on coarsened grids.[83] Li *et al.* proposed a factorized transformer to accurately and efficiently simulate 2D Kolmogorov flow and 3D smoke buoyancy.[84] Jiang *et al.* proposed a transformer-based decoding architecture for flow field prediction.[85] However, general transformer methods often employ grid downsampling techniques to reduce computational costs. Unfortunately, this approach tends to sacrifice high-frequency information of turbulence, leading to a decrease in simulation accuracy. In this work, we develop a transformer-based neural operator (TNO) as a surrogate model for LES of 3D turbulence. The TNO model enables direct inference and prediction of flow dynamics at a million-grid scale, ensuring the preservation of multiscale features of turbulence. It achieves stable, efficient, and accurate predictions on large-scale dynamics of turbulence.

The rest of the paper is structured as follows: Sec. II introduces the governing equations for LES and the conventional SGS models. Section III briefly introduces the Fourier neural operator and some modifications. In Sec. IV, we introduce a transformer-based neural operator model, namely, TNO model. Section V focuses on conducting numerical simulations for forced homogeneous isotropic turbulence (HIT) and free-shear mixing layer turbulence. This section includes a description of the data generation process, an evaluation of the proposed model's performance through *a posteriori* analysis, and an assessment of the computational efficiency achieved. Finally, Sec. VI gives some further discussions on the proposed method and concludes the paper.

II. THE LARGE-EDDY SIMULATION AND SUB-GRID SCALE MODELS

In this section, the large-eddy simulation (LES) of the incompressible turbulence and classical models for the unclosed sub-grid scale (SGS) stress are briefly introduced.

A. The governing equations for LES of the turbulence

For three-dimensional incompressible turbulence, the mass and momentum equations are given by[86,87]

$$\frac{\partial u_i}{\partial x_i} = 0, \tag{1}$$

$$\frac{\partial u_i}{\partial t} + \frac{\partial (u_i u_j)}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \nu \frac{\partial^2 u_i}{\partial x_j \partial x_j} + \mathcal{F}_i. \tag{2}$$

Here, u_i is the ith velocity component, p denotes the pressure divided by the constant density, ν represents the kinematic viscosity, and $\mathcal{F}_i$ stands for the ith large-scale force component. In addition, the root mean square (rms) velocity is defined by $u_{rms} = \sqrt{\langle u_i u_i \rangle}$, and $\langle \rangle$ denotes a spatial average of the computational domain. $\varepsilon = 2\nu \langle S_{ij} S_{ij} \rangle$ is the average dissipation rate of kinetic energy, where $S_{ij} = \frac{1}{2}(\partial u_i/\partial x_j + \partial u_j/\partial x_i)$ is the strain rate tensor. With these variables, the Kolmogorov length scale η, the Taylor length scale λ, and the Taylor-scale Reynolds number Re_λ can be, respectively, defined as[86,88]

$$\eta = \left(\frac{\nu^3}{\varepsilon}\right)^{1/4}, \quad \lambda = \sqrt{\frac{5\nu}{\varepsilon}}u_{\mathrm{rms}}, \quad Re_\lambda = \frac{u_{\mathrm{rms}}\lambda}{\sqrt{3}\nu}. \quad (3)$$

Furthermore, the integral length scale L_I and the large-eddy turn-over time τ are given, respectively, by[86]

$$L_I = \frac{3\pi}{2(u_{\mathrm{rms}})^2}\int_0^\infty \frac{E(k)}{k}dk, \quad \tau = \frac{L_I}{u_{\mathrm{rms}}}, \quad (4)$$

where $E(k)$ is the energy spectrum and the kinetic energy E_k is given by $E_k = \int_0^\infty E(k)dk = \frac{1}{2}(u_{\mathrm{rms}})^2$. Here, k represents the wave number in Fourier space.

In order to derive the governing equations for the large-scale flow field, a filtering operation $\overline{f}(\mathbf{x}) = \int_D f(\mathbf{x} - \mathbf{r})G(\mathbf{r}, \mathbf{x}; \Delta)d\mathbf{r}$ is applied to the Navier–Stokes equations. Here, G and Δ are the filter kernel and filter width, respectively.[86,89] f denotes a variable in physical space, and D is the entire domain. For any variable f in Fourier space, a filtered variable is given by $\overline{f}(\boldsymbol{k}) = \widehat{G}(\boldsymbol{k})f(\boldsymbol{k})$.[86] Here, $\widehat{G}(\boldsymbol{k})$ is the filter transfer function. Therefore, the filtered incompressible Navier–Stokes equations can be derived as follows:[86,89]

$$\frac{\partial \overline{u}_i}{\partial x_i} = 0, \quad (5)$$

$$\frac{\partial \overline{u}_i}{\partial t} + \frac{\partial(\overline{u}_i\overline{u}_j)}{\partial x_j} = -\frac{\partial \overline{p}}{\partial x_i} - \frac{\partial \tau_{ij}}{\partial x_j} + \nu\frac{\partial^2 \overline{u}_i}{\partial x_j \partial x_j} + \overline{\mathcal{F}}_i. \quad (6)$$

Here, τ_{ij} is the unclosed SGS stress defined by $\tau_{ij} = \overline{u_i u_j} - \overline{u}_i\overline{u}_j$. In order to ensure the solvability of the LES equations, it is crucial to model the SGS stress as a function of the filtered variables.

B. The sub-grid models for LES

The sub-grid scale (SGS) models aim to construct an approximate constitutive equation for the unclosed SGS terms based on the resolved variables and accurately capture the intricate nonlinear interactions between the resolved large scales and unresolved small scales.[90,91] One of the most commonly used SGS models is the Smagorinsky model, given by[1–3]

$$\tau_{ij}^A = \tau_{ij} - \frac{\delta_{ij}}{3}\tau_{kk} = -2C_s^2\Delta^2|\overline{S}|\overline{S}_{ij}, \quad (7)$$

where δ_{ij} is the Kronecker delta operator, Δ denotes the filter width, $\overline{S}_{ij} = \frac{1}{2}(\partial\overline{u}_i/\partial x_j + \partial\overline{u}_j/\partial x_i)$ stands for the strain rate of the filtered velocity, and $|\overline{S}| = (2\overline{S}_{ij}\overline{S}_{ij})^{1/2}$ represents the characteristic filtered strain rate. The coefficient C_s^2 can be determined empirically or through theoretical analysis.[2] This coefficient can also be dynamically established using the Germano identity and a least squares algorithm, yielding the dynamic Smagorinsky model (DSM) with the coefficient given by[15,92]

$$C_s^2 = \frac{\langle \mathcal{L}_{ij}\mathcal{M}_{ij}\rangle}{\langle \mathcal{M}_{kl}\mathcal{M}_{kl}\rangle}. \quad (8)$$

Here, $\mathcal{L}_{ij} = \widetilde{\overline{u}_i\overline{u}_j} - \widetilde{\overline{u}}_i\widetilde{\overline{u}}_j$, $\mathcal{M}_{ij} = \widetilde{\alpha}_{ij} - \beta_{ij}$, $\alpha_{ij} = 2\Delta^2|\overline{S}|\overline{S}_{ij}$, and $\beta_{ij} = 2\widetilde{\Delta}^2|\widetilde{\overline{S}}|\widetilde{\overline{S}}_{ij}$, where an overbar denotes the filtering at scale Δ, and a tilde represents coarser filtering $\widetilde{\Delta} = 2\Delta$.

Another commonly used SGS model, namely, the dynamic mixed model (DMM), combines the scale-similarity model with the dissipative Smagorinsky term and is given by[16,93]

$$\tau_{ij} = C_1\Delta^2|\overline{S}|\overline{S}_{ij} + C_2\left(\widetilde{\overline{u}_i\overline{u}_j} - \widetilde{\overline{u}}_i\widetilde{\overline{u}}_j\right). \quad (9)$$

Similar to the DSM model, the coefficients C_1 and C_2 of the DMM model are dynamically estimated using the Germano identity through the least squares algorithm. C_1 and C_2 are expressed as[94]

$$C_1 = \frac{\langle N_{ij}^2\rangle\langle L_{ij}M_{ij}\rangle - \langle M_{ij}N_{ij}\rangle\langle L_{ij}N_{ij}\rangle}{\langle N_{ij}^2\rangle\langle M_{ij}^2\rangle - \langle M_{ij}N_{ij}\rangle^2}, \quad (10)$$

$$C_2 = \frac{\langle M_{ij}^2\rangle\langle L_{ij}N_{ij}\rangle - \langle M_{ij}N_{ij}\rangle\langle L_{ij}M_{ij}\rangle}{\langle N_{ij}^2\rangle\langle M_{ij}^2\rangle - \langle M_{ij}N_{ij}\rangle^2}. \quad (11)$$

Here, $M_{ij} = H_{1,ij} - \widetilde{h}_{1,ij}$, $H_{1,ij} = -2\widetilde{\Delta}^2|\widetilde{\overline{S}}|\widetilde{\overline{S}}_{ij}$, $h_{1,ij} = -2\Delta^2|\overline{S}|\overline{S}_{ij}$, and $N_{ij} = H_{2,ij} - \widetilde{h}_{2,ij}$, $H_{2,ij} = \widetilde{\widetilde{\overline{u}}_i\widetilde{\overline{u}}_j} - \widehat{\widetilde{\overline{u}}}_i\widehat{\widetilde{\overline{u}}}_j$, $h_{2,ij} = \widetilde{\overline{u}_i\overline{u}_j} - \widetilde{\overline{u}}_i\widetilde{\overline{u}}_j$. The hat denotes the filter at scale $\widehat{\Delta} = 4\Delta$.

III. THE FOURIER NEURAL OPERATOR

The Fourier neural operators (FNO) train the model on a finite set of input–output pairs to establish a mapping between two infinite-dimensional spaces. The non-linear mapping is denoted as $G^\dagger : \mathcal{A} \to \mathcal{U}$, where $\mathcal{A} = \mathcal{A}(D; \mathbb{R}^{d_a})$ and $\mathcal{U} = \mathcal{U}(D; \mathbb{R}^{d_u})$ are separable Banach spaces of function taking values in $\mathbb{R}^{d_a}$ and $\mathbb{R}^{d_u}$, respectively.[95] Here, $D \subset \mathbb{R}^d$ is a bounded open set, $\mathbb{R}$ denotes real number space, $\mathbb{R}^{d_a}$ is the value sets of input $a(x)$, and $\mathbb{R}^{d_u}$ represents the value sets of output $u(x)$. The Fourier neural operators learn an approximation of $G^\dagger$ by constructing a mapping parameterized by $\theta \in \Theta$. The optimal parameters $\theta^\dagger \in \Theta$ are determined through data-driven methods.[96] The architecture of FNO is shown in Fig. 1, which consists of three main steps.

First, the input $a(x)$ is transformed to a higher-dimensional representation $v_0(x) = P(a(x))$ through a local transformation P, which is commonly parameterized by a shallow fully connected neural network.

Then, the higher-dimensional representation $v_0(x)$ is updated iteratively by

$$v_{t+1}(x) = \sigma(Wv_t(x) + (\mathcal{K}(a; \phi)v_t)(x)), \quad \forall x \in D. \quad (12)$$

Here, $\mathcal{K}$ is a non-local integral operator, $\mathcal{K} : \mathcal{A} \times \Theta_\mathcal{K} \to \mathcal{L}(\mathcal{U}(D; \mathbb{R}^{d_v}), \mathcal{U}(D; \mathbb{R}^{d_v}))$ maps to bounded linear operators on $\mathcal{U}(D; \mathbb{R}^{d_v})$ and is parameterized by $\phi \in \Theta_\mathcal{K}$, $\sigma : \mathbb{R} \to \mathbb{R}$ is non-linear activation function, and $W : \mathbb{R}^{d_v} \to \mathbb{R}^{d_v}$ is a linear transformation. The neural operators employ iterative architectures $v_0 \mapsto v_1 \mapsto \cdots \mapsto v_T$ where v_i for $i = 0, 1, \ldots, T - 1$ is a sequence of function.[97]

Finally, the output $u(x) = Q(v_T(x))$ is generated by applying a projection Q to the higher-dimensional representation $v_T(x)$, where Q is a fully connected layer that maps from $\mathbb{R}^{d_v}$ to $\mathbb{R}^{d_u}$.[48]

Let $\mathcal{F}$ and $\mathcal{F}^{-1}$ denote the Fourier transform and its inverse transform of a function $f : D \to \mathbb{R}^{d_v}$, respectively. By replacing the kernel integral operator in Eq. (12) with a convolution operator defined in Fourier space, the Fourier integral operator can be rewritten as
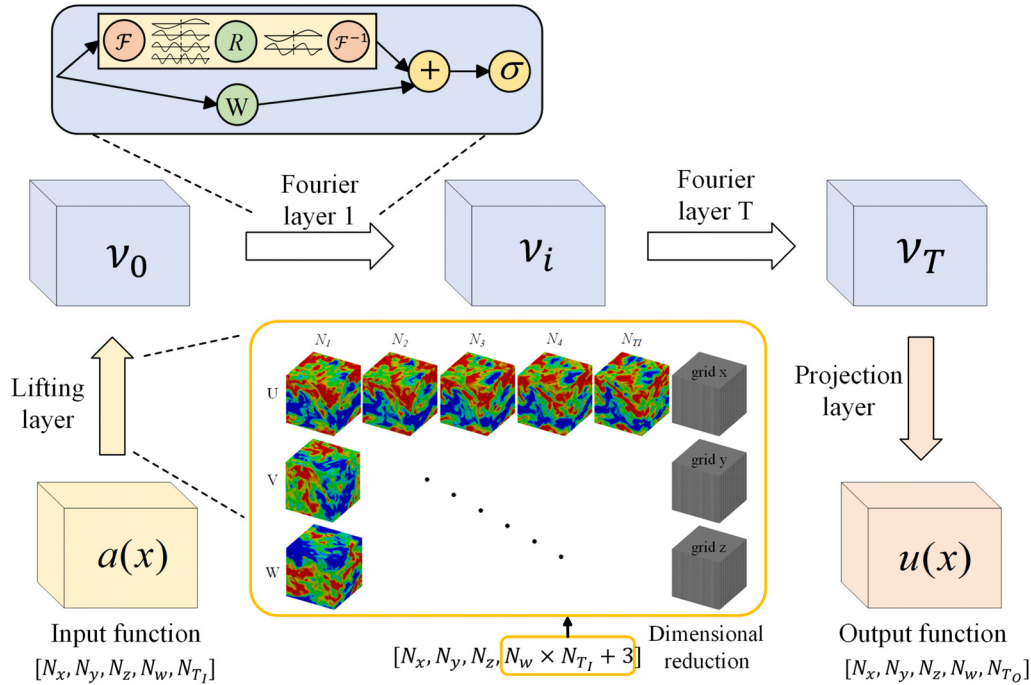
FIG. 1. The Fourier neural operator (FNO) architecture.

$$(\mathcal{K}(\phi)v_t)(x) = \mathcal{F}^{-1}\big(R_\phi \cdot (\mathcal{F}v_t)\big)(x), \quad \forall x \in D. \qquad (13)$$

Here, R_ϕ is the Fourier transform of a periodic function $\mathcal{K} : \overline{D} \to \mathbb{R}^{d_v \times d_v}$ parameterized by $\phi \in \Theta_{\mathcal{K}}$. Assuming that the frequency mode $k \in D$ is periodic, it can be represented using a Fourier series expansion that consists of discrete modes denoted as $k \in \mathbb{Z}^d$. Finite-dimensional parameterization can be obtained by truncating the Fourier series at a maximal mode $k_{\max} = |Z_{k_{\max}}| = |\{k \in \mathbb{Z}^d : |k_j| \le k_{\max,j}, \text{ for } j = 1, \ldots, d\}|$. $\mathcal{F}(v_t) \in \mathbb{C}^{n \times d_v}$ can be obtained by discretizing domain D with $n \in \mathbb{N}$ points, where $v_t \in \mathbb{R}^{n \times d_v}$.[48] $\mathcal{F}(v_t) \in \mathbb{C}^{k_{\max} \times d_v}$ can be obtained by truncating the higher modes, where $\mathbb{C}$ is the complex space. R_ϕ is parameterized as complex-valued weight tensor containing a collection of truncated Fourier modes $R_\phi \in \mathbb{C}^{k_{\max} \times d_v \times d_v}$. Therefore, the following equation can be derived:

$$\big(R_\phi \cdot (\mathcal{F}v_t)\big)_{k,l} = \sum_{j=1}^{d_v} R_{\phi k,l,j}(\mathcal{F}v_t)_{k,j},$$
$$k = 1, \ldots, k_{\max}, \quad j = 1, \ldots, d_v. \qquad (14)$$

For a flow that is uniformly discretized with resolution $s_1 \times \cdots \times s_d = n$, the fast Fourier transform (FFT) can be applied to $\mathcal{F}$. For $f \in \mathbb{R}^{n \times d_v}, k = (k_1, \ldots, k_d) \in \mathbb{Z}_{s_1} \times \cdots \times \mathbb{Z}_{s_d}$, and $x = (x_1, \ldots, x_d) \in D$, the FFT $\widehat{\mathcal{F}}$ and its inverse $\widehat{\mathcal{F}}^{-1}$ are given by

$$(\widehat{\mathcal{F}}f)_l(k) = \sum_{x_1=0}^{s_1-1} \cdots \sum_{x_d=0}^{s_d-1} f_l(x_1, \ldots, x_d) e^{-2i\pi \sum_{j=1}^{d} \frac{x_j k_j}{s_j}},$$

$$\left(\widehat{\mathcal{F}}^{-1}f\right)_l(x) = \sum_{k_1=0}^{s_1-1} \cdots \sum_{k_d=0}^{s_d-1} f_l(k_1, \ldots, k_d) e^{2i\pi \sum_{j=1}^{d} \frac{x_j k_j}{s_j}}. \qquad (15)$$

The FNO model stands out from traditional numerical methods and other neural operator approaches by showcasing remarkable adaptability and generalization in effectively handling high-dimensional and large-scale data.[55,60,68,98,99] The main idea of FNO is to utilize Fourier transform to map high-dimensional data to the frequency domain and employ neural network to learn the relationships between Fourier coefficients, enabling the approximation of nonlinear operators. This approach makes FNO to efficiently learn the governing rules of entire families of partial differential equations (PDEs).[68]

Unlike the approach employed in our previous study,[68] where direct FFT was applied to the four-dimensional data, we can improve accuracy and reduce computational cost by first reducing the dimensionality of the input and then applying FFT to the three-dimensional data. More specifically, Fig. 1 depicts the tensor input with the following dimensions: N_x, N_y, and N_z represent the number of grid points along the three respective directions, N_w indicates the velocity components in the three directions, and N_{T_I} is a time dimension of five in the input. We combine the three velocity components N_w with the five input time steps N_{T_I}, incorporating the spatial grid position information along the three directions. This integration allows us to map the data using lifting layer P into a higher-dimensional space for learning purposes. By employing this dimensionality reduction method through dimension merging, we effectively reduce the number of network parameters and address the issue of limited channel width due to memory constraints. This further enhances accuracy and computational efficiency. Therefore, all subsequent discussions of the FNO model refer to the modified version of the FNO model.

IV. A TRANSFORMER-BASED NEURAL OPERATOR

We develop a transformer-based neural operator (TNO) as a surrogate model for LES to achieve stable, efficient, and accurate

predictions on large-scale dynamics of turbulence. The TNO model aims to learn the mapping relationship between input $a(x)$ and output $\widehat{u}(x)$ through a data-driven approach. This mapping relationship can be expressed as

$$\widehat{u}(x) = \mathcal{G}(\mathbf{x}, a(x)). \tag{16}$$

Here, $\mathbf{x}$ represents the spatial coordinate encoding, $a(x)$ denotes the input function, which is typically the acquired velocity field at multiple time steps. $\mathcal{G}$ refers to the TNO network, and $\widehat{u}(x)$ represents the predicted velocity field for the subsequent time step.

Similar to FNO model, the flow field information of the first N_{T_I} time steps for the original input tensor is defined as $a(x)$, with a size of $a(x) = [N_x, N_y, N_z, N_w, N_{T_I}]$. Here, to simplify notation, we have omitted the batch size dimension and utilized a broadcasting mechanism for parallel computation along this dimension. Thus, only the remaining dimensions need to be considered. The transformer-based model is shown in Fig. 2.

The sequence reshape block $\mathcal{R}(a)$ transforms a three-dimensional tensor of spatial data into a sequence, converting the temporal and physical field components into features. It can be represented as $\mathcal{R}(a) := [N_x, N_y, N_z, N_w, N_{T_I}] \rightarrow [N_s, N_c]$. Here, the sequence $N_s = N_x \cdot N_y \cdot N_z$ and the features $N_c = N_w \cdot N_{T_I}$. The grid reshape block can be defined as $\mathcal{R}^{-1}(a) := [N_x \cdot N_y \cdot N_z, N_w] \rightarrow [N_x, N_y, N_z, N_w]$.

Figure 3 illustrates a schematic diagram of the multi-head attention mechanism. For the input $\mathbf{a}$, it is initially divided into N_h submatrices, corresponding to N_h heads. It can be represented as $\mathbf{a} = [\mathbf{a}^1, \mathbf{a}^2, \dots, \mathbf{a}^{N_h}]$. The embedding of the query, key, and value vectors $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ for each head can be expressed as

$$\mathbf{Q}^i = \mathbf{W_q}\mathbf{a}^i, \mathbf{K}^i = \mathbf{W_k}\mathbf{a}^i, \mathbf{V}^i = \mathbf{W_v}\mathbf{a}^i. \tag{17}$$

Here, the matrices $\mathbf{W_q}, \mathbf{W_k}, \mathbf{W_v}$ are learnable and shared across all heads. The dimensions of $\mathbf{Q}, \mathbf{K}$, and $\mathbf{V}$ are $[N_s, N_d]$, where N_s represents the number of samples and N_d denotes the dimensionality. Each $\mathbf{Q}^i$ for each head is of size N_d/N_h. It can be written as

$$\mathbf{Q} = [\mathbf{Q}^1, \dots, \mathbf{Q}^{N_h}]; \mathbf{K} = [\mathbf{K}^1, \dots, \mathbf{K}^{N_h}]; \mathbf{V} = [\mathbf{V}^1, \dots, \mathbf{V}^{N_h}]. \tag{18}$$

By utilizing Attention($\cdot$) to denote the general notion of attention calculation, it can be defined as

$$\mathbf{z} = \text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{sim}(\mathbf{Q}\mathbf{K}^T)\mathbf{V}. \tag{19}$$

Here, the function $sim(\cdot)$ represents the similarity calculation function (including the normalization process), and $(\cdot)^T$ denotes matrix transpose. The computational complexity defined by Eq. (19) is $\mathcal{O}(N_s^2 N_d)$, where N_s represents the length of the grid node sequence and N_d denotes the dimensionality of the embedding. This method poses significant computational challenges, particularly in the case of predicting physical fields where N_s is typically proportional to the number of grid nodes. In the case of predicting three-dimensional turbulence, the scale of nodes can reach magnitudes of 10^6 to 10^7, imposing a substantial computational burden.

To maintain consistency with the notation commonly used in attention literature, we represent the ith row in the sequence as q_i, k_i, v_i, where $0 < i < d$. Similarly, we denote the vector formed by the jth column in the sequence as q^j, k^j, v^j, where $0 < j < N_s$. It can be rewritten as

$$(\mathbf{z}_i)^j = \sum_{j=1}^{N_s} w_s (q_i \cdot k_l) v_l \approx \int_\Omega \kappa(x_i, \xi) v^j(\xi) d\xi. \tag{20}$$

Here, the operation between the attention matrix and the values can be considered as an integration operation of a kernel function. By considering the values as the basis functions, selecting the Fourier basis functions as the set of kernel functions, utilizing the inverse Fourier transform basis functions for the values, and incorporating learnable parameters and mode truncation method, the aforementioned operation is equivalent to the Fourier neural operator.

By employing linear integral attention to approximate Eq. (20), we can derive the following equation:

$$(\mathbf{z}^j)_i = \sum_{l=1}^{d} \frac{(\mathbf{k}^l \cdot \mathbf{v}^j)}{n} (\mathbf{q}^l)_i \approx \sum_{l=1}^{d} \left(\int_\Omega (k^l(\xi) v^j(\xi)) d\xi \right) q_l(x_i). \tag{21}$$

By adopting this linear attention mechanism, the computational complexity can be reduced to $\mathcal{O}(N_s N_d^2)$, leading to a significant reduction in computational workload when dealing with millions of grid nodes.[69]

The Relative Positional Encoding (RoPE) is a relative position encoding method with promising theoretical properties. For a given
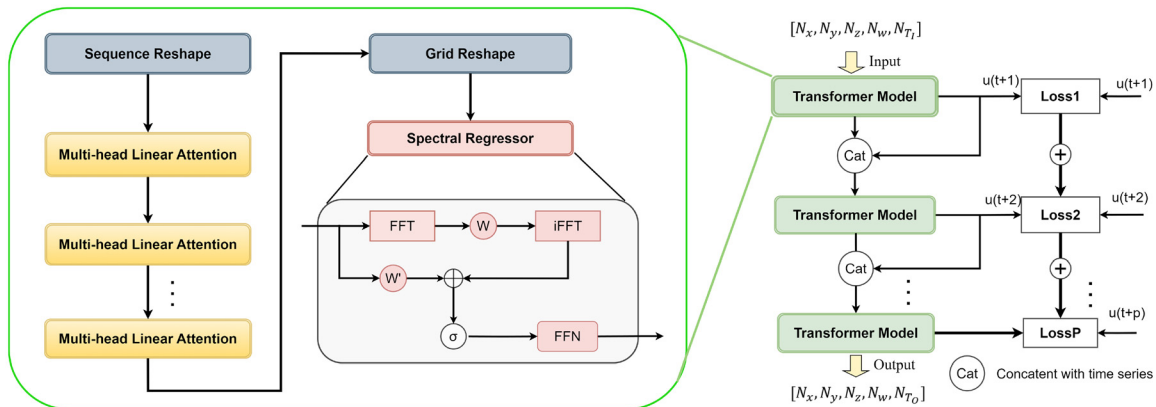

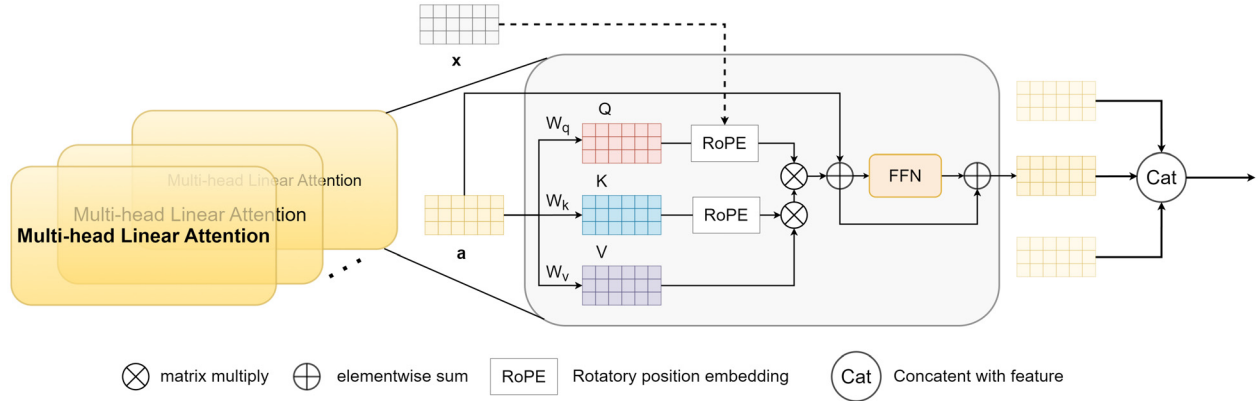**FIG. 2.** The transformer-based neural operator (TNO) architecture.

FIG. 3. The architecture of multi-head linear attention.

three-dimensional spatial coordinate $\mathbf{x}$, it can be represented as $\mathbf{x}_i = (\alpha_i, \beta_i, \gamma_i), 0 \leq i \leq N_s$. Correspondingly, $\mathbf{q}_i$ can be split along the feature dimension into three sub-matrices $[q_i^{\alpha}, q_i^{\beta}, q_i^{\gamma}]$, where q_i^{α} represents the q-matrix for the first coordinate, with a length of $[N_s, N_d/3]$. For one of the coordinates, taking α_i as an example, the RoPE can be expressed as[100]

$$\psi(\mathbf{q}_i, \mathbf{x}_i) = \mathbf{\Theta}(\mathbf{x}_i)\mathbf{q}_i,$$
$$\mathbf{\Theta}(\mathbf{x}_i) = \mathrm{Diag}\big(\mathbf{\Theta}(\alpha_i), \mathbf{\Theta}(\beta_i), \mathbf{\Theta}(\gamma_i)\big),$$
$$\mathbf{\Theta}(\alpha_i) = \mathrm{Diag}\big(\mathbf{R}_1^{\alpha_i}, \mathbf{R}_2^{\alpha_i}, ..., \mathbf{R}_{d/6}^{\alpha_i}\big), \quad (22)$$
$$\mathbf{R}_l^{\alpha_i} = \begin{bmatrix} \cos(\lambda\alpha_i\theta_l) & -\sin(\lambda\alpha_i\theta_l) \\ \sin(\lambda\alpha_i\theta_l) & \cos(\lambda\alpha_i\theta_l) \end{bmatrix}.$$

Here, $\mathrm{Diag}(\cdot)$ denotes stacking sub-matrices along the diagonal, and λ is wavelength of the spatial domain (e.g., $\lambda = 128$ for a spatial dimension discretized by 128 equidistant points in one dimension). $\theta_l = 10\,000^{-2(l-1)/d}, l \in \{1, 2, ..., d/2\}$.

It can be proven that the RoPE defined for the 3D sequence satisfies the following properties:

$$\psi(\mathbf{q}_i, \mathbf{x}_i) \cdot \psi(\mathbf{k}_j, \mathbf{x}_j) = \big(\mathbf{\Theta}(x_i)\mathbf{q}_i\big) \cdot \big(\mathbf{\Theta}(x_j)\mathbf{k}_j\big) = \mathbf{\Theta}(\mathbf{x}_i - \mathbf{x}_j)\big(\mathbf{q}_i \cdot \mathbf{k}_j\big). \quad (23)$$

Here, "$\cdot$" represents the dot product of vectors.

The feed forward neural network (FFN) in the spectral regressor consists of two multilayer perceptron (MLP) layers and is mainly used to adjust the number of feature channels. The FFN is defined as

$$\mathrm{FFN}(a) := W_2\sigma(W_1 a + b_1), \quad (24)$$

where W_1, b_1, and W_2 are the learnable parameters of fully connected networks. σ is nonlinear activation functions.

For the spectral regressor module, the regressor can be expressed as a function of the given input function $v(x)$ and can be defined as

$$\mathrm{FNO}(v) := \sigma\Big(\mathcal{F}^{-1}\big(W(k) * \mathcal{F}(v)(k)\big) + W'v + b\Big). \quad (25)$$

Here, $*$ denotes convolution operation, and $\mathcal{F}$ and $\mathcal{F}^{-1}$ represent the fast Fourier transform and its inverse, respectively. The weight matrix

function $W(k)$, which is applied in the Fourier space, is a learnable function that depends on the wavenumber k. The weight matrix W' is a learnable matrix applied in the physical space. b and σ denote learnable bias terms and nonlinear activation functions, respectively. The spectral regressor module can be seen as an extension of the Fourier layer in the FNO model introduced in Sec. III, combined with an additional FFN module. For detailed information on the Fourier layer, refer to Sec. III.

When using the network for prediction, the input of the velocity field at the previous N_{T_I} time steps to predict the velocity field at the next time step can be represented as

$$\widehat{\mathbf{u}}^{N_{T_I}+1} = \mathcal{G}(\mathbf{x}, [\mathbf{u}^1, \mathbf{u}^2, ..., \mathbf{u}^{N_{T_I}-1}, \mathbf{u}^{N_{T_I}}]). \quad (26)$$

When making predictions for the second time step using the same approach, it can be represented as

$$\widehat{\mathbf{u}}^{N_{T_I}+2} = \mathcal{G}(\mathbf{x}, [\mathbf{u}^2, \mathbf{u}^3, ..., \mathbf{u}^{N_{T_I}}, \widehat{\mathbf{u}}^{N_{T_I}+1}]). \quad (27)$$

Here, we concatenate the results obtained from the first prediction $\widehat{\mathbf{u}}^{N_{T_I}+1}$ with the input tensor for the second prediction, while considering only the previous N_{T_I} steps that are relevant to the target prediction time step. This recursive prediction approach can be employed for long-term forecasting of physical fields. When the prediction time is greater than N_{T_I}, all input tensors have transformed into the predicted tensors of the model.

The general loss function for data-driven methods of velocity fields can be expressed as

$$\|\widehat{\mathbf{u}} - \mathbf{u}\|_{\mathbf{\Omega},\mathbf{u}} = \int_{\mathbf{u}}\int_{\mathbf{\Omega}} \sum_{m=1}^{N_w} \|\widehat{\mathbf{u}}^m(\mathbf{x}, t) - \mathbf{u}^m(\mathbf{x}, t)\| d\mathbf{\Omega}d\mathbf{T}. \quad (28)$$

Here, $\widehat{\mathbf{u}}$ is the predicted velocity field, and $\mathbf{u}$ is ground truth velocity field. $\mathbf{\Omega}$ and $\mathbf{T}$ are the spatial domain and temporal domain, respectively.

Given the iterative computation of velocity fields at different time steps in a recurrent manner, the temporal dimension can be discretized into N_t time steps. Consequently, the loss function can be discretized as

$$\int_T \int_\Omega \sum_{m=1}^{N_w} \|\widehat{\mathbf{u}}^m(\mathbf{x},t) - \mathbf{u}^m(\mathbf{x},t)\| d\Omega d\mathbf{T}$$

$$\approx \frac{1}{N_t} \sum_{i=1}^{N_p} \|\widehat{\mathbf{u}}^{im}(\mathbf{x}) - \mathbf{u}^{im}(\mathbf{x})\|_\Omega. \tag{29}$$

Here, $\widehat{\mathbf{u}}^i$ is the predicted velocity field, $\mathbf{u}^i$ denotes ground truth, and N_p represents the number of time steps that need to be iterated.

Considering the uniform spatial distribution of the points where the velocity field is evaluated, it can be further discretized into a high-dimensional tensor representation using the L2 norm,

$$\|\widehat{\mathbf{u}}^i(\mathbf{x}) - \mathbf{u}^i(\mathbf{x})\|_\Omega \approx \frac{1}{N_s} \sum_{j,k,l=1}^{N_x,N_y,N_z} \left(\widehat{\mathbf{u}}_{jkl}^{im} - \mathbf{u}_{jkl}^{im}\right)^2. \tag{30}$$

Therefore, the final loss function can be rewritten as

$$\mathcal{L}(\widehat{\mathbf{u}}, \mathbf{u}) = \frac{1}{N_p} \sum_{i,m=1}^{N_p,N_w} \frac{1}{N_s} \sum_{j,k,l=1}^{N_x,N_y,N_z} \left(\widehat{\mathbf{u}}_{jkl}^{im} - \mathbf{u}_{jkl}^{im}\right)^2. \tag{31}$$

V. NUMERICAL EXPERIMENTS

In this chapter, we will systematically analyze and compare the performance between the FNO model, commonly used traditional sub-grid models, such as the dynamic Smagorinsky model (DSM) and dynamic mixed model (DMM), and our proposed TNO model. The numerical experiments will be conducted in two different flow types: statistically steady forced homogeneous isotropic turbulence (HIT) and unsteady free-shear turbulence. By showcasing the data generation and training processes, comparing *a posteriori* results, and evaluating computational efficiency, we aim to compare the performance of different models comprehensively.

In different flow cases, all models underwent ten independent repeated numerical experiments, each evolving from a different random initial field. Then, the *a posteriori* statistical results from each model's ten runs were averaged separately to reduce the influence of experimental randomness and facilitate a more meaningful comparison.

A. Performance on forced homogeneous isotropic turbulence

1. Dataset description

The direct numerical simulation (DNS) of forced homogeneous isotropic turbulence is performed in a $(2\pi)^3$ cubic box using 256^3 uniform grid, with periodic boundary conditions and a second-order two-step Adams–Bashforth time scheme.[88,94] The pseudo-spectral method is adopted, and the aliasing error caused by nonlinear advection terms is eliminated by the two-thirds rule.[101] The kinematic viscosity is $\nu = 0.00625$, and the Taylor Reynolds number is $Re_\lambda \approx 100$.[68] To ensure the turbulence remains in a statistically steady state, we apply large-scale forcing and save the data after an extended period[94] (more than 10τ, here $\tau = L_I/u_{rms} \approx 1.0$ is large-eddy turnover times).

In the present study, the DNS data are filtered into large-scale flow fields at grid resolutions of 64^3 by the sharp spectral filter $\widehat{G}(\boldsymbol{k}) = H(k_c - |\boldsymbol{k}|)$ in Fourier space for homogeneous isotropic turbulence.[86] Here, the cutoff wavenumber is $k_c = \pi/\Delta = 21$, and Δ denotes the filter width. The Heaviside step function $H(x) = 1$ if $x \geq 0$; otherwise, $H(x) = 0$.[86,102]

The DNS data are recorded with a time step of 0.001, and snapshots of the numerical solution are captured at intervals of 200 steps as a time node. Therefore, each prediction step performed by the neural operator is equivalent to 200 steps of DNS computation, corresponding to 0.2τ. We employ 100 different random fields as initial conditions, and for each set of computations, we save 600 time nodes. Therefore, the DNS data with the tensor size of $[100 \times 600 \times 256 \times 256 \times 256 \times 3]$ can be obtained, and the filtered direct numerical simulation (fDNS) data with the tensor size of $[100 \times 600 \times 64 \times 64 \times 64 \times 3]$ will be served as a training and testing dataset.[68] Specifically, the dataset we use to train the neural operator model consists of 100 groups, each group has 600 time nodes, and each time node denotes a filtered velocity field of 64^3 with three directions. For these 100 groups of HIT DNS data, each group would take approximately 4.3 h to parallel computing on a 32-core central processing unit (CPU). Figure 4 demonstrates the velocity contours of three random initial fields, so that the differences between different initial fields can be seen more intuitively.
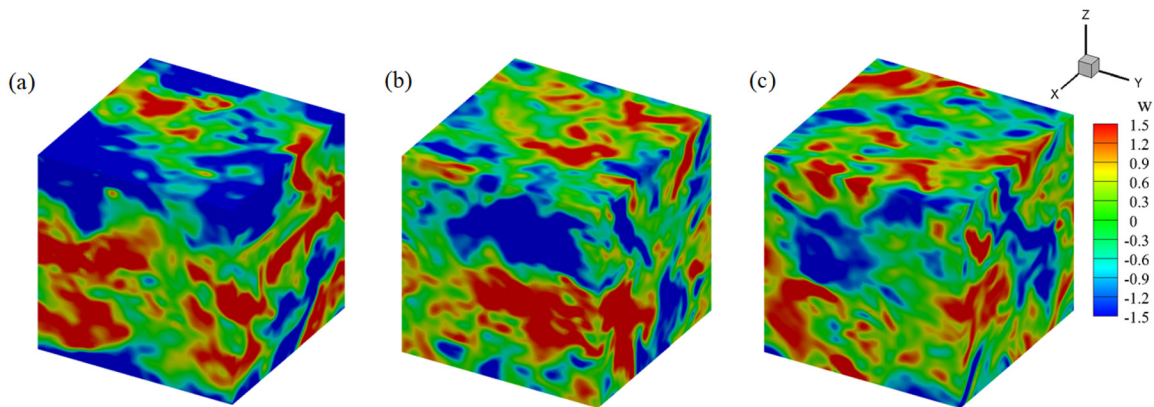

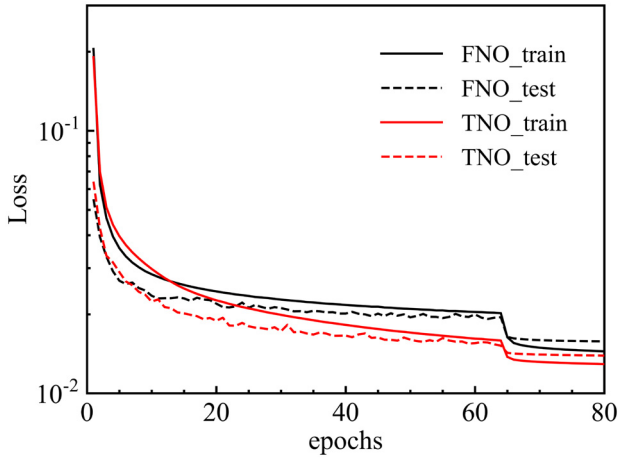**FIG. 4.** Velocity contours of different initial fields.

FIG. 5. The learning curves of the FNO and TNO models for LES of 3D HIT.

We employ a consistent training approach for both FNO and TNO models. We use the data from the previous five time steps as inputs to the neural operator models, aiming to learn the mapping between these inputs and the corresponding data at the sixth time step.[68] We denote the i-th time-node filtered velocity field as U_i. Therefore, the input tensor is $[U_i, U_{i+1}, U_{i+2}, U_{i+3}, U_{i+4}]$, and the output tensor is $[U_{i+5}]$, which is taken as input–output pairs.[68] By utilizing 100 different initial conditions and capturing 600 time nodes in each group, these samples are split into 80% for training and 20% for testing.[68] Once the model training is completed, it will be able to make fast predictions for the evolution of the filtered velocity field by directly mapping the relationship between inputs and outputs.

To ensure a fair comparison, the hyperparameters are set to be identical. The learning curves of the FNO and TNO models for 3D HIT is shown in Fig. 5. The FNO model and TNO model take 14.6 and 11.2 h to train the network, respectively. Both FNO and TNO models employed in this study utilize the same number of Fourier modes, specifically 16, and share a common initial learning rate of 10^{-3}.[69] The channel width of P is configured as 64, and the channel width of Q is 128. The AdamW optimizer is chosen as optimization with $\beta = (0.9, 0.999)$.[103,104] The GELU function is used for the activation function.[105]

To avoid overfitting in the models, we generated and used an additional ten independent groups of data from different initial fields for the *a posteriori* evaluation. In the *a posteriori* study, fDNS data are utilized as a benchmark to evaluate various models. Moreover, all the neural operator models and classical SGS models used for comparison are initialized with the same initial field. More specifically, while the DSM and DMM models utilize the flow field data of the fifth step as the initial conditions, the TNO and FNO models employ the first five steps of the flow field data of the new initial field for the inference process.

2. A posteriori study

The velocity spectrum predicted by various models at different time instants is shown in Fig. 6. It can be seen that the velocity spectrum, as predicted by both classical LES models and data-driven models, demonstrates persistent consistency with fDNS result in both short-term and long-term predictions. Here, the large-eddy turnover time τ is provided in Eq. (4). As observed in Fig. 6, the DMM model underestimates the velocity spectrum compared to the fDNS result at low wavenumbers and significantly overestimates the spectrum at high wavenumbers. The figure also reveals that the TNO and FNO models exhibit similar overall trends in predicting the velocity spectrum. However, the TNO model demonstrates a slightly more accuracy than the FNO model at both high and low wavenumbers regions, giving a closer alignment with the fDNS. Furthermore, it can be seen from Fig. 6(b) that the velocity spectrum predicted by the TNO model are more accurate than those of the DSM model within the wavenumber $3 \leq k \leq 17$, while the DSM model demonstrates greater accuracy when the wavenumber $k \geq 18$. Therefore, both the TNO and DSM models show similar performance in predicting the velocity spectrum in the HIT case, achieving good agreement with the fDNS.

In order to further compare the performance of different models in predicting multi-scale properties of turbulence, we calculate the longitudinal structure functions of the filtered velocity. These structure functions are defined by[106,107]

$$S_n(r) = \left\langle \left| \frac{\delta_r \overline{u}}{\overline{u}_{\mathrm{rms}}} \right|^n \right\rangle, \tag{32}$$

where n is the order of structure function, $\overline{u}_{\mathrm{rms}} = \sqrt{\langle \overline{u}_i \overline{u}_i \rangle}$, and $\delta_r \overline{u} = [\overline{\mathbf{u}}(\mathbf{x} + \mathbf{r}) - \overline{\mathbf{u}}(\mathbf{x})] \cdot \widehat{\mathbf{r}}$ represents the longitudinal increment of the velocity at the separation $\mathbf{r}$. Here, $\widehat{\mathbf{r}} = \mathbf{r}/|\mathbf{r}|$ denotes the unit distance vector.
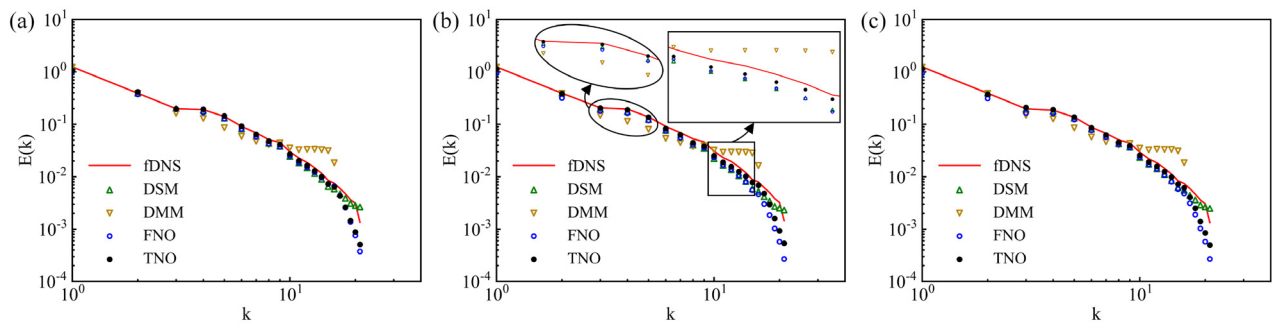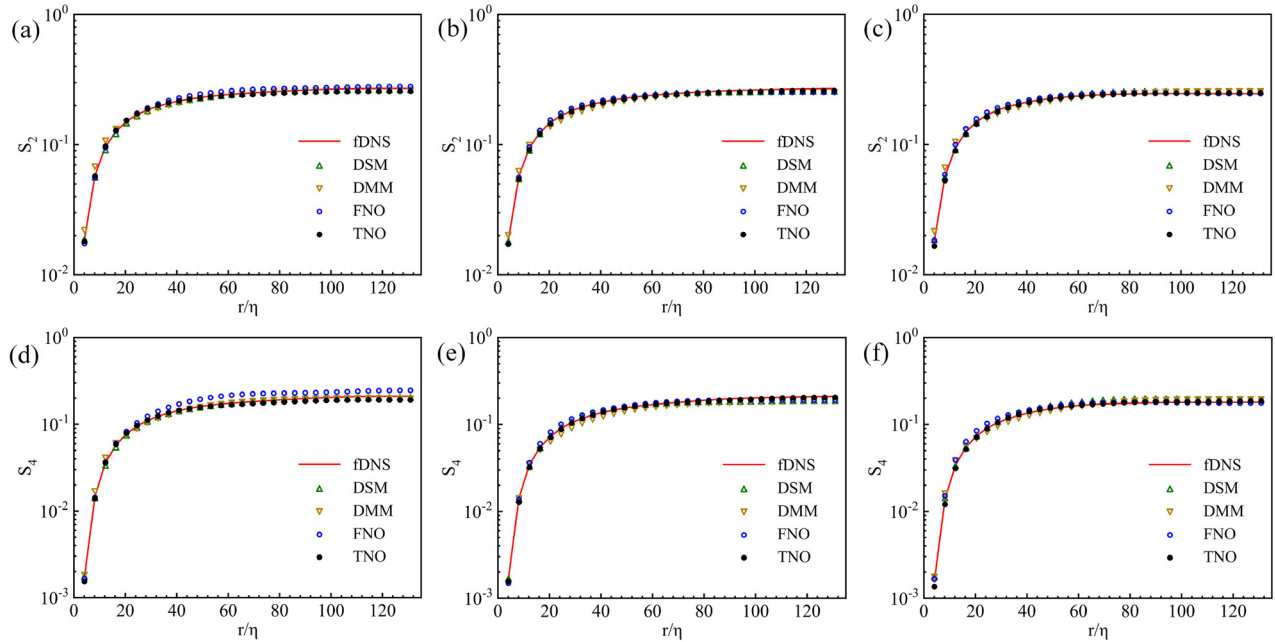

FIG. 6. Comparisons of velocity spectrum for LES in the forced HIT at different time instants: (a) $t/\tau \approx 6.0$; (b) $t/\tau \approx 18$; (c) $t/\tau \approx 40$.

FIG. 7. Structure functions of the velocity for LES in the forced HIT at different time instants: (a) S_2, $t/\tau \approx 6.0$; (b) S_2, $t/\tau \approx 18$; (c) S_2, $t/\tau \approx 40$; (d) S_4, $t/\tau \approx 6.0$; (e) S_4, $t/\tau \approx 18$; (f) S_4, $t/\tau \approx 40$.

Figure 7 compares the structure functions of the filtered velocity for different models with fDNS data at different time instants. Here, the Kolmogorov scale are used to normalize the distance. It can be seen that the DMM model exhibits an overestimation of the structure

functions at short distances in comparison to those of the fDNS data. As can be observed from Figs. 7(a) and 7(d), the FNO model exhibits an increasing deviation in predicting the structure functions at $t/\tau \approx 6.0$ as the distance r/η increases, with the predicted results
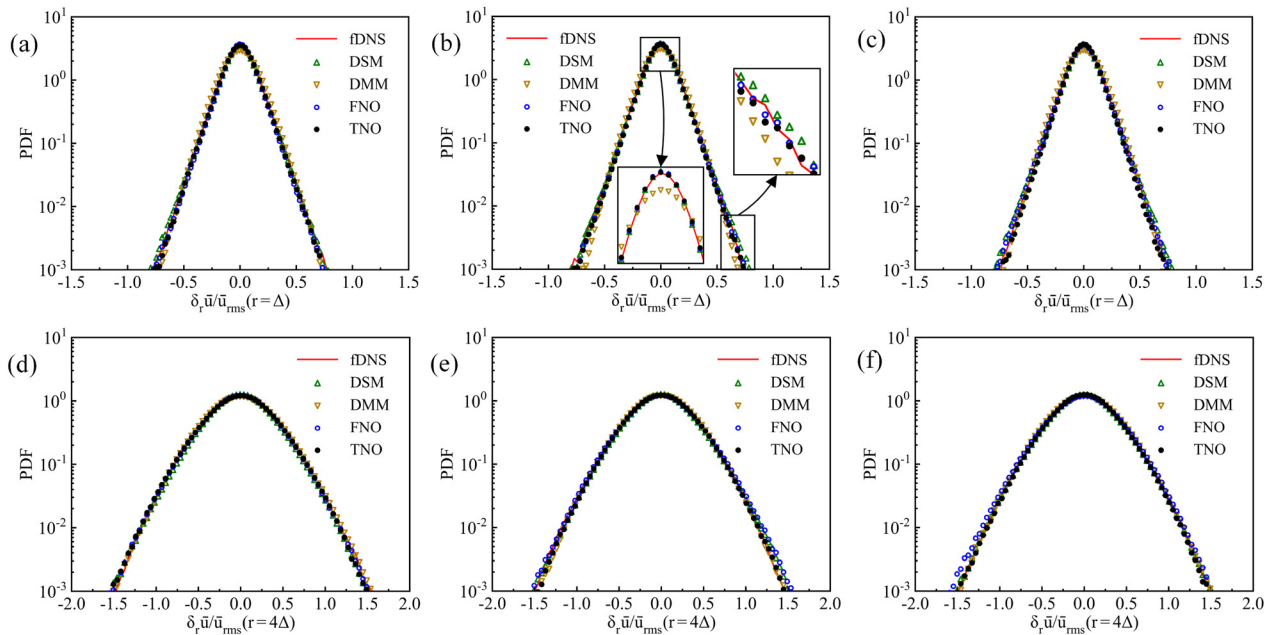
FIG. 8. The PDFs of the normalized velocity increments for LES in the forced HIT at different time instants: (a) $r/\Delta = 1$, $t/\tau \approx 6.0$; (b) $r/\Delta = 1$, $t/\tau \approx 18$; (c) $r/\Delta = 1$, $t/\tau \approx 40$; (d) $r/\Delta = 4$, $t/\tau \approx 6.0$; (e) $r/\Delta = 4$, $t/\tau \approx 18$; (f) $r/\Delta = 4$, $t/\tau \approx 40$.
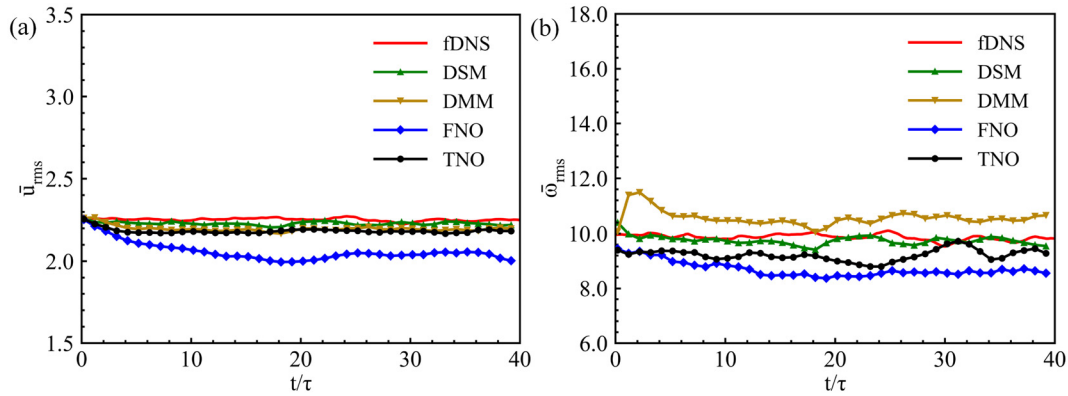
FIG. 9. Temporal evolutions of the velocity rms value and vorticity rms value for LES in the forced HIT.

overestimating those obtained from fDNS. In contrast, the TNO and DSM models can always accurately predict the structure functions at both short-term and long-term predictions.

Furthermore, we compare probability density functions (PDFs) of the normalized velocity increments $\delta_r \overline{u}/\overline{u}^{rms}$ with distance $r = \Delta$ and $r = 4\Delta$ at different time instants in Fig. 8. It can be seen that the PDFs of the normalized velocity increments predicted by all models show a good agreement with the fDNS data. From Fig. 8(b), it is seen that the predictions of the velocity increments from the TNO model and DSM model are very close to fDNS in the region near zero velocity increment. However, the predictions from the DSM model and the DMM model deviate after $\delta_r \overline{u}/\overline{u}_{rms} = 0.5$, while the TNO model maintains a good consistency with fDNS. This further demonstrates the TNO model's superior accuracy in comparison to the other models.

In order to demonstrate the stability of the different models, we depict the temporal evolution of the root mean square (rms) values of velocity and vorticity in Fig. 9. It can be observed that both DSM and DMM models, as well as the enhanced FNO and TNO models, exhibit relatively stable predictive performance. The stability of classical LES models can be attributed to their inherent dissipative characteristics.[1,108] The improved data-driven models, FNO and TNO models,

can provide stable predictions for simple flow cases such as HIT by effectively learning from vast of data. It can be seen that the rms values of velocity and vorticity predicted by DSM model are closer to those of fDNS when comparing with other models. This is also due to the fact that the DSM model applies large-scale forces identical to fDNS at the two largest wavenumbers during numerical simulation. Among the two data-driven models, the TNO model exhibits a more precise performance than the FNO model.

Furthermore, the PDFs of the normalized vorticity magnitude at different time instants are shown in Fig. 10. Here, the vorticity is normalized by the root mean square values of the vorticity obtained from the fDNS data. It is observed that the PDFs predicted by DSM and TNO models have a good agreement with those of fDNS in both short-term and long-term predictions. However, both FNO and DMM models exhibit a significant deviation from fDNS.

We demonstrate contours of the predicted velocity fields in the z-direction for LES in the forced HIT at several time instants in Fig. 11. Here, due to the characteristics of homogeneous isotropic turbulence, we have arbitrarily chosen to present the velocity in the z-direction. Additionally, we randomly selected one case from the 10 different cases with different initial fields to generate the contours. It is observed that the velocity fields predicted by TNO and DSM models exhibit a
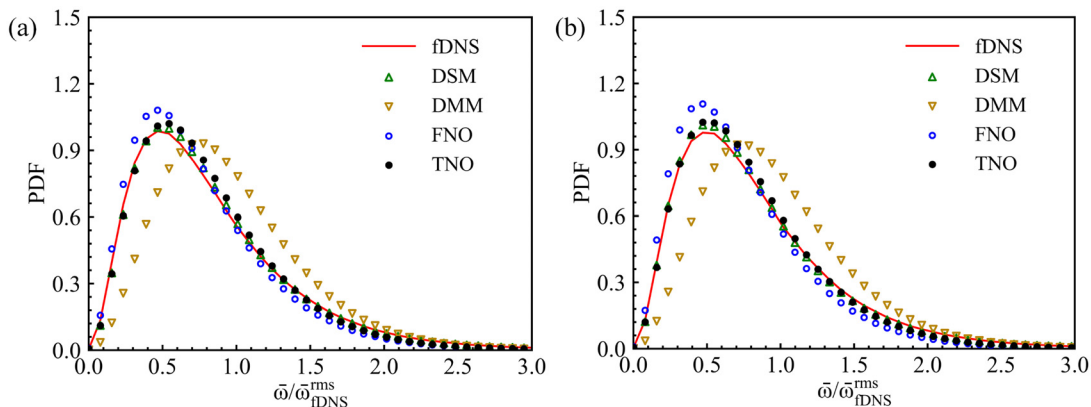

FIG. 10. The PDFs of the normalized vorticity for LES in the forced HIT at different time instants: (a) $t/\tau \approx 6.0$; (b) $t/\tau \approx 40$.
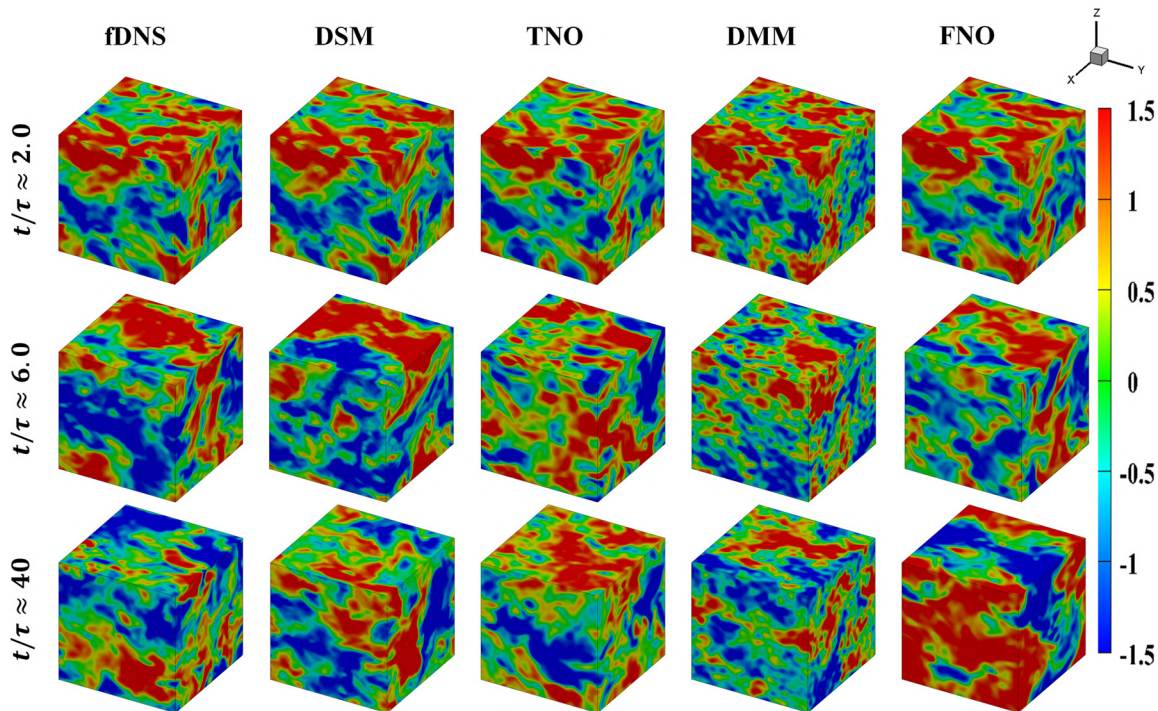
FIG. 11. Contours of the predicted velocity field in the z-direction for LES in the forced HIT at different time instants.

closer tendency to the fDNS results within a short-term forecasting $t/\tau \approx 2.0$. It becomes challenging to maintain complete similarity in the flow field after a long-term evolution $t/\tau \approx 40$. Therefore, for long-term predictions, our primary focus is on observing the overall statistical trends of the flow field. We can observe that the LES with DSM and TNO models exhibit a closer tendency to the fDNS results.

3. Computational efficiency

Here, we compare the computational efficiency among different models in Table I. The recorded time represents the time required by different models to predict each time node in the forced HIT. We conduct the neural network models using Pytorch and train the models on Nvidia Tesla V100 graphics processing unit (GPU), where the CPU type is Intel(R) Xeon(R) Gold 6240 CPU @2.60 GHz. The DSM simulations are implemented on a computing cluster, where the type of CPU is Intel Xeon Gold 6148 with 16 cores each @2.40 GHz. It can be observed that the TNO model exhibits a prediction speed that is three

TABLE I. Computational efficiency of different approaches on forced HIT.

Method	Total parameters ($\times 10^6$)	GPU (s)	CPU (s)
DSM	N/A	N/A	105.6
DMM	N/A	N/A	166.4
FNO	563.9	0.206	4.366
TNO	268.6	1.021	12.48

times slower than the FNO model. However, it achieves a notable reduction in the number of network parameters when compared to FNO. It can be seen that the TNO model requires only 12.48 s of CPU time to predict a single time step, while the LES with DSM and DMM models require 105.6 and 166.4 s, respectively. Furthermore, if the TNO model is run on a GPU, the time can be further reduced to 1 s. Compared to LES with DSM and DMM models, the trained TNO model demonstrates a significant advantage in computational efficiency. Although it is important to consider the longer training time required for the TNO model, its ability to achieve faster flow field predictions is a notable advantage.

4. Generalization on higher Taylor Reynolds numbers

Moreover, the generalization ability of the 3D TNO model is tested. The well-trained TNO model can be directly used to predict the fDNS data at higher Taylor Reynolds number cases without additional training or modifications. In this study, we generated four sets of forced HIT data with different initial fields at higher Taylor Reynolds numbers $Re_\lambda \approx 160$ and $Re_\lambda \approx 250$, respectively, to assess the model's generalization performance during the inference process. By employing the same prediction methodology as in the forced HIT with low Taylor Reynolds number, we input the first five time steps of $Re_\lambda \approx 160$ forced HIT velocity field data into the well-trained data-driven models to forecast the further evolution of the flow field. Here, we perform individual predictions for four sets of data with distinct initial conditions and subsequently average the statistical quantities in the *a posteriori* analysis.
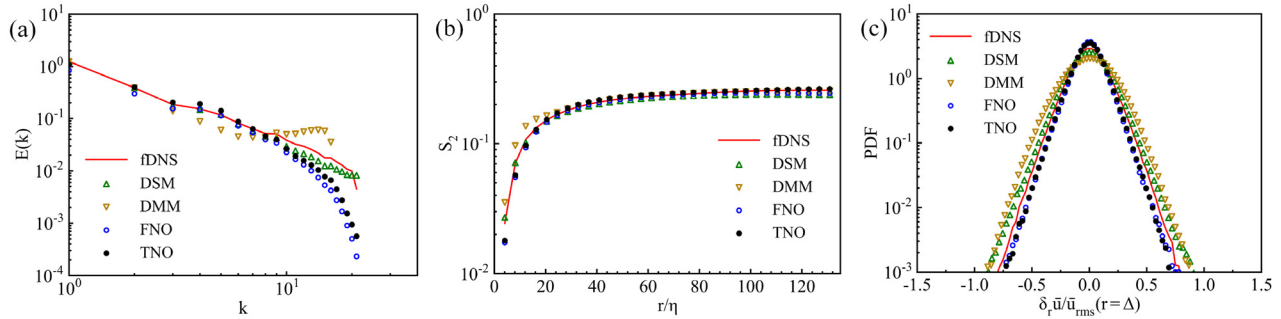
FIG. 12. Statistical analysis of prediction when Taylor Reynolds number $Re_\lambda \approx 160$: (a)Velocity spectrum; (b) second-order structure functions; (c) PDFs of the normalized velocity increments.

Figure 12 presents the statistical quantities, including the velocity spectrum, second-order structure functions, and PDFs of the normalized velocity increments, predicted by different models in forced HIT with Taylor Reynolds numbers $Re_\lambda \approx 160$ at $t/\tau_m \approx 28.6$. Here, the large-eddy turnover time is $\tau_m \approx 0.7$. It can be seen from Fig. 12(a) that the velocity spectrum predicted by the DSM model demonstrates higher accuracy and closer agreement with those of fDNS compared to other models. However, Figs. 12(b) and 12(c) reveal that the TNO model exhibits superior accuracy in predicting the second-order structure functions and PDFs of velocity increments compared to LES with DSM and DMM models.

Moreover, we further test different models at a higher Taylor Reynolds number $Re_\lambda \approx 250$. The average statistical quantities predicted by different models in forced HIT with Taylor Reynolds numbers $Re_\lambda \approx 250$ at $t/\tau_m \approx 26.7$ are illustrated in Fig. 13. Here, the large-eddy turnover time is $\tau_h \approx 0.6$. It can be observed that the velocity spectrum predicted by the DSM model demonstrates higher accuracy compared to other models in general. However, in the prediction of second-order structure functions and PDFs of velocity increments, the TNO model exhibits superior precision compared to LES with DSM and DMM models.

Furthermore, it can be observed from Figs. 12(c) and 13(c) that as the Taylor Reynolds number increases, the DSM and DMM models exhibit deviations in predicting the PDFs of velocity increments. In contrast, the PDFs of velocity increments predicted by the TNO model demonstrates better consistency with those of the fDNS data. Overall, both the TNO and FNO models exhibit generalization capabilities to higher

Taylor Reynolds numbers, producing predictions that are comparable to those of the DSM model and closely consistent with the fDNS data.

When dealing with higher Taylor Reynolds number cases, both the DSM and DMM models require higher temporal resolution, resulting in an increased computational time. However, the TNO model can be directly applied to higher Taylor Reynolds number cases and accurate predictions can be achieved while maintaining the computational efficiency. Since turbulent flow data for high Taylor Reynolds numbers is scarce, it is important to train the TNO model using data with lower Reynolds numbers and apply the well-trained model to predict turbulence with higher Taylor Reynolds numbers. Considering the self-similarity of turbulence, the large-scale statistical features and flow structures in some regions are insensitive to the Reynolds numbers, which provides a theoretical foundation for developing such a model.

B. Performance on free-shear turbulence

Apart from assessing the performance of data-driven models and classical sub-grid scale (SGS) models on a three-dimensional (3D) forced homogeneous isotropic turbulence (HIT), we also evaluate their capabilities in a more challenging simulation task: a 3D turbulent mixing layer with free-shear.

1. Dataset description

The dynamics of the free-shear turbulent mixing layer are described by the same Navier–Stokes equations [Eqs. (1) and (2)] but
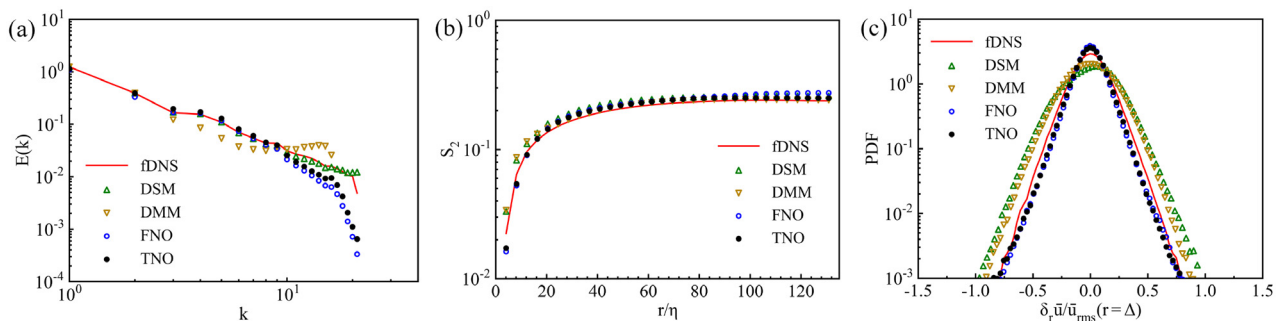
FIG. 13. Statistical analysis of prediction when Taylor Reynolds number $Re_\lambda \approx 250$: (a) Velocity spectrum; (b) second-order structure functions; (c) PDFs of the normalized velocity increments.
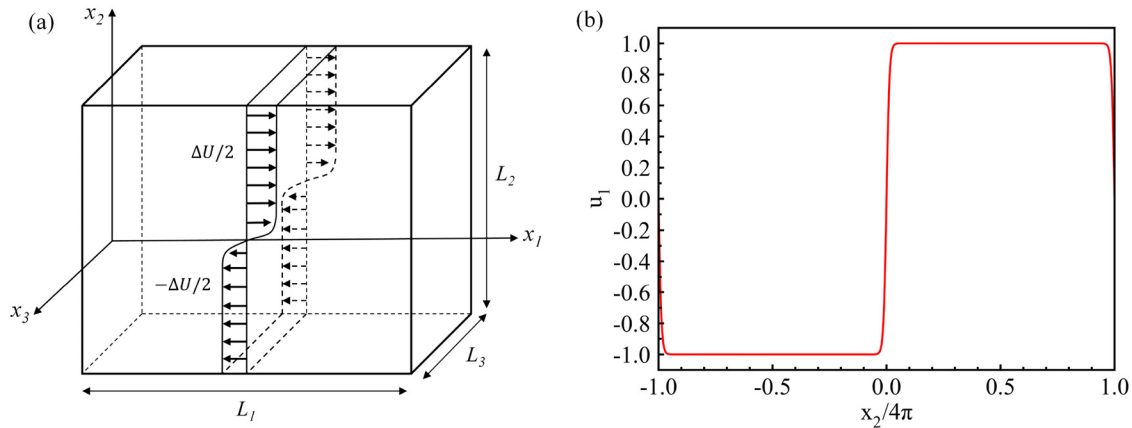
FIG. 14. Diagram of the temporally evolving mixing layer with the velocity profile: (a) schematic of the mixing layer; (b) streamwise velocity profile along the normal direction.

without a forcing term. Figure 14 illustrates the diagram of the flow configuration for the temporally evolving turbulent mixing layer with the initial hyperbolic tangent streamwise velocity profile. The mixing layer is performed in a cuboid domain with lengths $L_1 \times L_2 \times L_3 = 8\pi \times 8\pi \times 4\pi$ using a uniform grid resolution of $N_1 \times N_2 \times N_3 = 256 \times 256 \times 128$, where $x_1 \in [-L_1/2, L_1/2]$, $x_2 \in [-L_2/2, L_2/2]$, and $x_3 \in [-L_3/2, L_3/2]$ denote the streamwise, normal, and spanwise directions, respectively. The initial streamwise velocity is given by[88,109,110]

$$u_1 = \frac{\Delta U}{2}\left[\tanh\left(\frac{x_2}{2\delta_\theta^0}\right) - \tanh\left(\frac{x_2 + L_2/2}{2\delta_\theta^0}\right)\right.$$
$$\left. - \tanh\left(\frac{x_2 - L_2/2}{2\delta_\theta^0}\right)\right] + \lambda_1, \quad (33)$$

where $\delta_\theta^0 = 0.08$ is the initial momentum thickness and $\Delta U = U_2 - U_1 = 2$ represents the velocity contrast between two identical and opposing free streams across the shear layer.[88,111] The momentum thickness measures the extent of the turbulent region within the mixing layer, which is given by[110–112]

$$\delta_\theta = \int_{-L_2/4}^{L_2/4}\left[\frac{1}{4} - \left(\frac{\langle \overline{u}_1 \rangle}{\Delta U}\right)^2\right]dx_2. \quad (34)$$

The initial normal and spanwise velocities are given as $u_2 = \lambda_2$ and $u_3 = \lambda_3$, respectively. Here, $\lambda_1, \lambda_2, \lambda_3 \sim \mathcal{N}(\mu, \sigma^2)$, i.e., $\lambda_1, \lambda_2, \lambda_3$ satisfy the Gaussian random distribution.[109] The expectation of the distribution is $\mu = 0$ and the variance of the distribution is $\sigma^2 = 0.01$. $\tau_\theta = \delta_\theta^0/\Delta U = 20dt$, and $dt = 0.002$ represents the time step used in the DNS simulation. In order to minimize the influence of the upper and lower boundaries on the central mixing layer, two numerical diffusion buffer zones are introduced at the vertical edges of the computational domain.[88,109,111] Periodic boundary conditions are employed in all three directions. The spatial discretization is performed using the pseudo-spectral method with the two-thirds dealiasing rule, while the time-advancing scheme is implemented using an explicit two-step Adam–Bashforth scheme.

The DNS data are subjected to explicit filtering using a widely used Gaussian filter, which is defined by[86,89]

$$G(\mathbf{r}; \overline{\Delta}) = \left(\frac{6}{\pi\overline{\Delta}^2}\right)^{1/2} \exp\left(-\frac{6\mathbf{r}^2}{\overline{\Delta}^2}\right). \quad (35)$$

Here, the filter scale $\overline{\Delta} = 4h_{DNS}$ is chosen for the free-shear turbulent mixing layer, where h_{DNS} is the grid spacing of DNS. The filter-to-grid ratio FGR $= \overline{\Delta}/h_{LES} = 1$ is utilized and the corresponding grid resolution of LES $64 \times 64 \times 32$ can be obtained.[88,102]

With the same method, we conduct numerical simulations for 200 different sets of initial fields and save the data for 90 time snapshots for each set of initial fields. Each snapshot is taken at a time interval of $200dt$. Therefore, the data of size $[200 \times 90 \times 64 \times 64 \times 32 \times 3]$ can be obtained as training and testing sets. For these 200 groups of turbulent mixing layer DNS data, each group would take approximately 1.1 h to parallel computing on a 32-core CPU. Similar to Sec. V A, 80% of data is used for training and 20% is used for testing. In this case, employing the same network parameters, the study sets the Fourier modes to 16, the initial learning rate to 10^{-3}, the channel width of P to 64, and configures the channel width of Q as 128. The AdamW optimizer is used, and the GELU function is employed as the activation function. The learning curves of the FNO and TNO models for 3D free-shear turbulent mixing layer is shown in Fig. 15. The FNO model and TNO model take 15.2 and 10.6 h to train the network, respectively. It can be seen that the training and testing loss of the TNO model are lower than those of the FNO model.

In order to mitigate overfitting in the models, we generated and employed ten extra independent datasets originating from diverse initial fields to conduct the *a posteriori* evaluation. In the *a posteriori* study, fDNS data are utilized as a baseline to evaluate various models. Moreover, all neural operators and classical models used for comparison are initialized with the same initial field.

2. A posteriori study

The temporal evolutions of the momentum thickness δ_θ for LES in the free-shear turbulent mixing layer are shown in Fig. 16. It can be
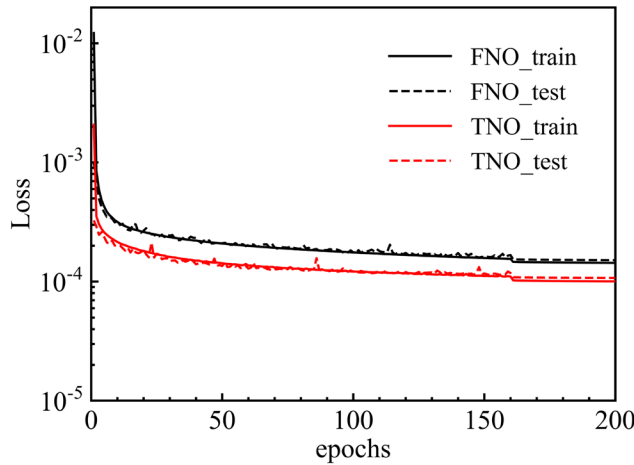
FIG. 15. The learning curves of the FNO and TNO models for 3D free-shear turbulent mixing layer.

FIG. 16. Temporal evolutions of the momentum thickness δ_θ for LES in the free-shear turbulent mixing layer.

seen that the DMM model tends to overestimate the momentum thickness in comparison to the fDNS model. The DSM model tends to underestimate the momentum thickness during the initial stages of the transition region. However, it tends to overestimate the momentum thickness in the linear growth region. The FNO model demonstrates a strong capability to accurately capture the growth rate of momentum thickness in the early stages of temporal development. Nevertheless, its predictive ability becomes unreliable after reaching a period of 400 units ($t/\tau_\theta \geq 400$). In contrast, the predictions of TNO model show a good agreement with fDNS in the transition region and maintain relatively stable results in the linear growth regions.

Moreover, the temporal evolutions of the streamwise turbulent kinetic energy $E_{k1} = \frac{1}{2}\left(\sqrt{\langle u_1 u_1 \rangle}\right)^2$, normal turbulent kinetic energy $E_{k2} = \frac{1}{2}\left(\sqrt{\langle u_2 u_2 \rangle}\right)^2$, and spanwise turbulent kinetic energy $E_{k3} = \frac{1}{2}\left(\sqrt{\langle u_3 u_3 \rangle}\right)^2$ are displayed in Fig. 17. Here, $\langle \cdot \rangle$ is a spatial average over the whole computational domain. During the development of the shear layer in fDNS, there is a gradual increase in the turbulent kinetic energy across various directions. The DSM and DMM models underestimate the turbulent kinetic energy at the beginning and overestimate the result in the linear growth region. The FNO model predicts reasonable results during the first 200 time units ($t/\tau_\theta \leq 200$), after that the
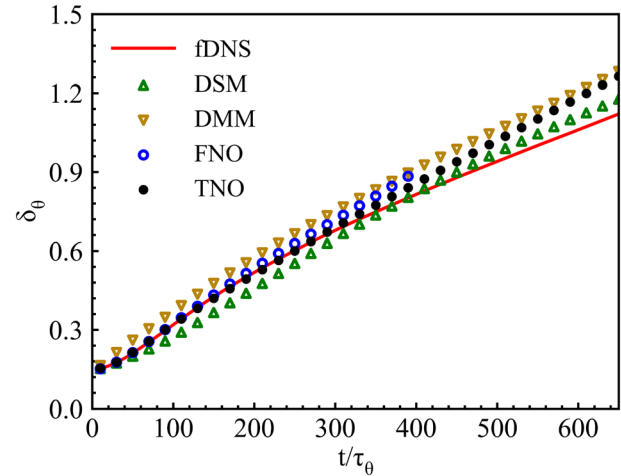
results diverge quickly. By contrast, the TNO model closely matches the fDNS data by accurately predicting the kinetic energy in all three directions throughout the development of the shear layer.

The velocity spectrum of different models at time instants $t/\tau_\theta \approx 100$, $t/\tau_\theta \approx 300$, and $t/\tau_\theta \approx 650$ are shown in Fig. 18. The velocity spectrum predicted by the DSM model is observed to be underestimated at $t/\tau_\theta \approx 100$, whereas it is overestimated for the subsequent time period starting from $t/\tau_\theta \approx 300$. The DMM model overestimates the velocity spectrum at low wavenumbers and underestimates the velocity spectrum at high wavenumbers when compared to those of the fDNS. The FNO model is capable of providing reasonable predictions in the short-term, but the deviation from the fDNS results becomes more significant for longer time. In comparison, the TNO model demonstrates superior performance compared to other models at $t/\tau_\theta \approx 100$ and $t/\tau_\theta \approx 300$, with predictions closely matching the results of fDNS. The velocity spectrum predicted by all models exhibit a certain degree of deviation from fDNS at $t/\tau_\theta \approx 650$, while the TNO model slightly outperforms the other models in terms of prediction accuracy.

We then illustrate the PDFs of velocity increment in the spanwise direction, as shown in Fig. 19. The spanwise velocity increment is defined by $\delta_{r_3}\overline{u} = [\overline{\mathbf{u}}(\mathbf{x} + \mathbf{r}) - \overline{\mathbf{u}}(\mathbf{x})] \cdot \widehat{\mathbf{e}_3}$, where $\widehat{\mathbf{e}_3}$ is the unit vector
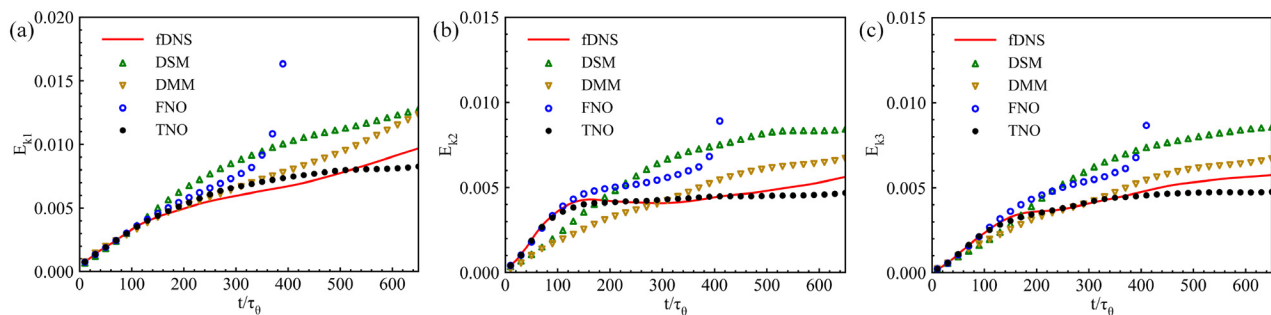

FIG. 17. Temporal evolutions of the turbulent kinetic energy for LES in the free-shear turbulent mixing layer: (a) Streamwise; (b) normal; (c) spanwise.
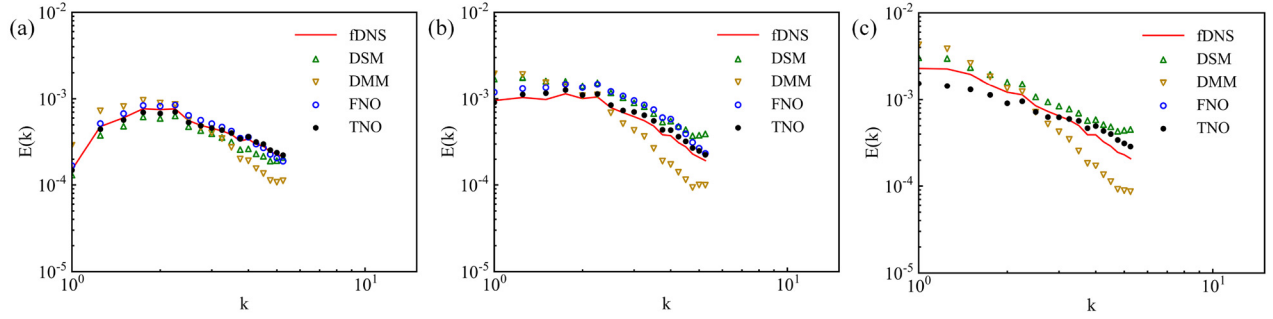
FIG. 18. Velocity spectrum for LES in the free-shear turbulent mixing layer at different time instants: (a) $t/\tau_\theta \approx 100$; (b) $t/\tau_\theta \approx 300$; (c) $t/\tau_\theta \approx 650$.
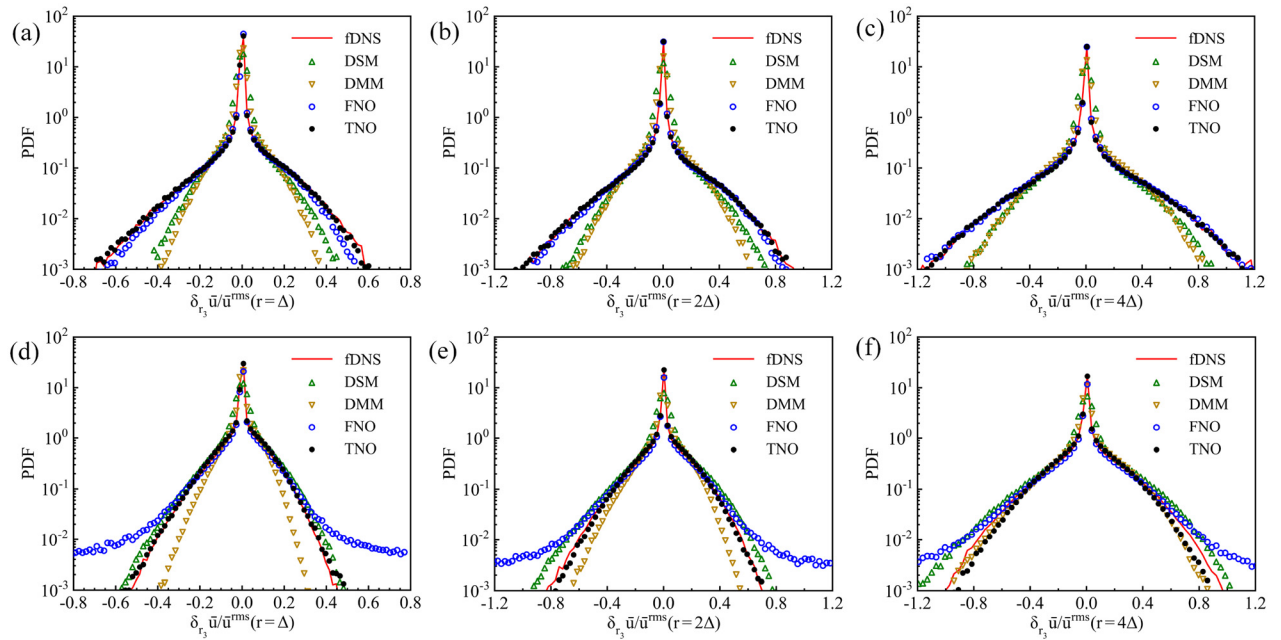

FIG. 19. The PDFs of the spanwise velocity increment for LES in the free-shear turbulent mixing layer at different time instants: (a) $r = \Delta$, $t/\tau_\theta \approx 100$; (b) $r = 2\Delta$, $t/\tau_\theta \approx 100$; (c) $r = 4\Delta$, $t/\tau_\theta \approx 100$; (d) $r = \Delta$, $t/\tau_\theta \approx 650$; (e) $r = 2\Delta$, $t/\tau_\theta \approx 650$ (f) $r = 4\Delta$, $t/\tau_\theta \approx 650$.

in the spanwise direction, and the velocity increments are normalized by the rms values of velocity $\overline{u}^{rms}$. The sharp peak of PDF can be attributed to the non-turbulent regions where the velocity increment is close to zero in the spanwise direction.[88] Conversely, the regions characterized by non-zero velocity increments are predominantly influenced by turbulence. It is observed that both DSM and DMM models exhibit deviations from the fDNS results in both short-term and long-term predictions. The FNO model provides reasonably accurate predictions at $t/\tau_\theta \approx 100$, but the deviations become unacceptable as the prediction time increases. In contrast, the velocity increment given by the TNO model shows an excellent agreement with those of the fDNS for both short-term and long-term predictions.

The instantaneous isosurfaces of $Q = 0.2$ at $t/\tau_\theta \approx 100$ and $t/\tau_\theta \approx 650$ colored by the streamwise velocity in the free-shear turbulent mixing layer are shown in Fig. 20. The Q-criterion has been widely

used for visualizing vortex structures in turbulent flows and is defined by[113,114]

$$Q = \frac{1}{2}\left(\overline{\Omega}_{ij}\overline{\Omega}_{ij} - \overline{S}_{ij}\overline{S}_{ij}\right), \qquad (36)$$

where $\overline{\Omega}_{ij} = (\partial \overline{u}_i/\partial x_j - \partial \overline{u}_j/\partial x_i)/2$ denotes the filtered rotation-rate tensor. It is observed that the LES with DSM and DMM models predict relatively larger vortex structures compared to the fDNS result. The vortex structures predicted by the FNO model become unreliable after $t/\tau_\theta \approx 650$. On the contrary, the TNO model demonstrates a higher level of agreement with fDNS results, particularly in terms of reconstructing the small vortex structures, highlighting its advantage in improving the accuracy of LES.

Finally, we present the velocity profiles and absolute errors of the predicted streamwise velocities for both FNO and TNO models at
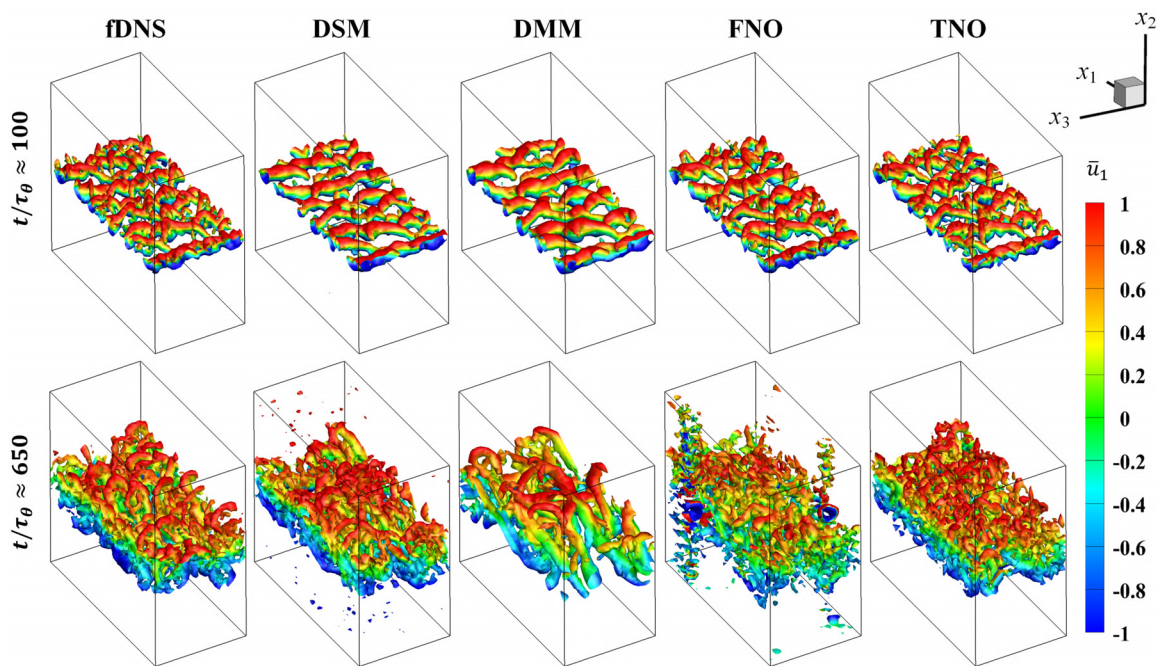
FIG. 20. The iso-surface of the Q-criterion at $Q = 0.2$ colored by the streamwise velocity at $t/\tau_\theta \approx 100$ and $t/\tau_\theta \approx 650$ in the free-shear turbulent mixing layer.

different time instants in Fig. 21. It can be observed that the velocity profiles predicted by both the FNO and TNO models demonstrate a good agreement with those obtained from the fDNS results at $t/\tau_\theta \approx 100$. However, as time advances, a significant increase in prediction error is observed for the FNO model at $t/\tau_\theta \approx 650$. The TNO model demonstrates smaller errors in predicting the streamwise velocity compared to the fDNS results, highlighting its superior accuracy over the FNO model.

3. Computational efficiency

Here, we compare the computational efficiency among different models in Table II. The recorded time represents the time required by different models to predict each time node in free-shear turbulent mixing layer. We train the FNO and TNO models on Nvidia Tesla V100 GPU. The LES with DSM and DMM models are implemented on a computing cluster, where the type of CPU is Intel Xeon Gold 6148
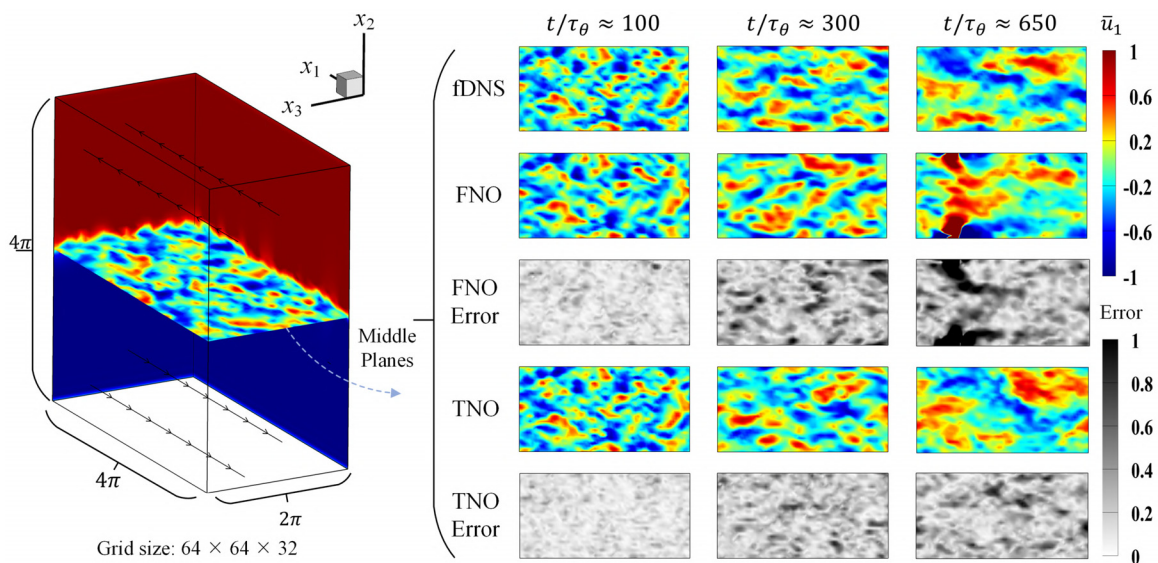

FIG. 21. Velocity profile in the free-shear turbulent mixing layer at different time instants.

TABLE II. Computational efficiency of different approaches on free-shear turbulent mixing layer.

Method	Total parameters ($\times 10^6$)	GPU (s)	CPU (s)
DSM	N/A	N/A	83.55
DMM	N/A	N/A	97.86
FNO	563.9	0.426	7.912
TNO	268.6	1.691	20.18

with 16 cores each @2.40 GHz. It can be observed that the TNO model achieves a notable reduction in the number of network parameters when compared to FNO model. It can be seen that the TNO model requires only 20.18 s of CPU time to predict a single time step, while the LES with DSM and DMM models require 83.55 and 97.86 s, respectively. Furthermore, if the TNO model is run on a GPU, the time can be further reduced to 1.691 s. This also demonstrates the potential of the TNO model in terms of fast predictions.

VI. CONCLUSION

Simulating 3D nonlinear PDEs is crucial in engineering applications, but data-driven approaches for fast 3D PDEs simulations are scarce compared to their success in 1D and 2D PDEs. Accurately modeling the complex non-linear interactions in 3D PDEs requires significant model complexity and a large number of parameters, posing a major challenge. Compared to the FNO model, transformer-based models are more suitable for developing accurate large-scale models due to their adaptability to larger datasets. Recently, the transformer-based models have demonstrated remarkable effectiveness as surrogate models for solving PDEs, highlighting their considerable potential in addressing 3D nonlinear problems.

In this work, we developed a transformer-based neural operator (TNO) model to effectively predict the large-scale dynamics of 3D turbulence. The velocity fields of the previous time steps are taken as inputs of the model, aiming to directly map them to the velocity field of the next time step. By training the model with a large amount of data, the TNO learns the evolution rules of the fluid dynamics from the data, enabling faster and more accurate predictions of the flow field evolution. The proposed model is tested in the LES of two types of 3D turbulence: forced homogeneous isotropic turbulence and free-shear turbulent mixing layer. The classical SGS models, including the DSM and DMM models, are utilized as comparison models in LES, with fDNS results serving as the benchmark.

A more systematic evaluation method is employed to validate the performance of different models. The numerical simulations comprehensively evaluate the performance of these models on a variety of flow statistics, including the velocity spectrum, structure functions, the PDFs of vorticity, the PDFs of velocity increments, momentum thickness, turbulent kinetic energy, and the iso-surface of the Q-criterion. The results demonstrate that the TNO model performs similarly to the LES with DSM model and outperforms the FNO model and the LES using DMM model in HIT. Moreover, in the turbulent mixing layer, the TNO model exhibits superior accuracy compared to the other models. Moreover, compared to the FNO model, the TNO model not only achieves higher prediction accuracy while significantly reducing the number of network parameters, but it also demonstrates the ability to provide long-term stable predictions in the turbulent mixing layer

where the FNO model fails. Meanwhile, the well-trained TNO model is much more efficient than traditional LES with DSM and DMM models. Moreover, the trained TNO model can be well generalized to the situations of higher Taylor–Reynolds numbers. Therefore, TNO has great potential in developing advanced neural network models to solve 3D nonlinear problems in engineering applications.

One limitation of the proposed model is that it has only been evaluated on simple flow cases, while real-world engineering applications typically involve much more complex flows. Due to the influence of boundary conditions, further improvements and modifications to the model are required when extending its application to more complex flow cases. Recently, more advanced FNO-based and transformer-based model have been proposed,[60,115–117] but their evaluation has been primarily limited to 2D problems. Li *et al.* proposed a geo-FNO model to handle PDEs on irregular geometries by mapping the input physical domain into a uniform latent space using a deformation function.[55] This approach provides valuable insights for dealing with 3D turbulence on non-uniform grids and non-periodic boundary conditions. In future work, we will also consider the attention mechanism in the temporal dimension to enhance prediction accuracy and generalization abilities. The data-driven TNO model requires separate training for each distinct flow configuration. In the follow-up study, we will try to develop large models so that they can be better generalized to different flow types. Moreover, we will integrate various improvement approaches to explore the application of the proposed model in more complex engineering flow problems.

AUTHOR DECLARATIONS
Conflict of Interest

The authors have no conflicts to disclose.

Author Contributions

Zhijie Li: Conceptualization (equal); Data curation (equal); Formal analysis (equal); Investigation (equal); Methodology (equal); Software (equal); Validation (equal); Visualization (equal); Writing – original draft (equal). **Tianyuan Liu:** Conceptualization (equal); Data curation (equal); Formal analysis (equal); Investigation (equal); Methodology (equal);

Resources (equal); Software (equal); Validation (equal); Writing – original draft (equal). **Wenhui Peng:** Conceptualization (equal); Data curation (equal); Methodology (equal); Software (equal); Supervision (equal). **Zelong Yuan:** Conceptualization (equal); Methodology (equal); Software (equal); Writing – review & editing (equal). **Jianchun Wang:** Conceptualization (equal); Funding acquisition (equal); Investigation (equal); Methodology (equal); Project administration (equal); Resources (equal); Supervision (equal); Writing – review & editing (equal).

DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding authors upon reasonable request.

REFERENCES

[1] J. Smagorinsky, "General circulation experiments with the primitive equations: I. The basic experiment," Mon. Weather Rev. **91**, 99–164 (1963).

[2] D. K. Lilly, "The representation of small-scale turbulence in numerical simulation experiments," in *Proceedings of the IBM Scientific Computing Symposium on Environmental Sciences* (IBM Data Processing Division, 1967), pp. 195–210.

[3] M. Germano, "Turbulence: The filtering approach," J. Fluid Mech. **238**, 325–336 (1992).

[4] E. Garnier, N. Adams, and P. Sagaut, *Large Eddy Simulation for Compressible Flows* (Springer Science & Business Media, 2009).

[5] J. W. Deardorff, "A numerical study of three-dimensional turbulent channel flow at large Reynolds numbers," J. Fluid Mech. **41**, 453–480 (1970).

[6] P. A. Durbin, "Some recent developments in turbulence closure modeling," Annu. Rev. Fluid Mech. **50**, 77–103 (2018).

[7] S. B. Pope, "A more general effective-viscosity hypothesis," J. Fluid Mech. **72**, 331–340 (1975).

[8] H. Choi and P. Moin, "Grid-point requirements for large eddy simulation: Chapman's estimates revisited," Phys. Fluids **24**, 011702 (2012).

[9] X. I. A. Yang and K. P. Griffin, "Grid-point and time-step requirements for direct numerical simulation and large-eddy simulation," Phys. Fluids **33**, 015108 (2021).

[10] Y. Bin, G. Huang, R. Kunz, and X. I. A. Yang, "Constrained recalibration of Reynolds-averaged Navier–Stokes models," AIAA J. **62**, 1434 (2023).

[11] C. Meneveau and J. Katz, "Scale-invariance and turbulence models for large-eddy simulation," Annu. Rev. Fluid Mech. **32**, 1–32 (2000).

[12] B. Fan, Z. Yuan, Y. Wang, and J. Wang, "Eddy viscosity enhanced temporal direct deconvolution models for temporal large-eddy simulation of turbulence," Phys. Fluids **35**, 065126 (2023).

[13] S. Chen, Z. Xia, S. Pei, J. Wang, Y. Yang, Z. Xiao, and Y. Shi, "Reynolds-stress-constrained large-eddy simulation of wall-bounded turbulent flows," J. Fluid Mech. **703**, 1–28 (2012).

[14] P. Moin, K. Squires, W. Cabot, and S. Lee, "A dynamic subgrid-scale model for compressible turbulence and scalar transport," Phys. Fluids **3**, 2746–2757 (1991).

[15] M. Germano, U. Piomelli, P. Moin, and W. H. Cabot, "A dynamic subgrid-scale eddy viscosity model," Phys. Fluids **3**, 1760–1765 (1991).

[16] Y. Shi, Z. Xiao, and S. Chen, "Constrained subgrid-scale stress model for large eddy simulation," Phys. Fluids **20**, 011701 (2008).

[17] G. Erlebacher, M. Y. Hussaini, C. G. Speziale, and T. A. Zang, "Toward the large-eddy simulation of compressible turbulent flows," J. Fluid Mech. **238**, 155–185 (1992).

[18] R. A. Clark, J. H. Ferziger, and W. C. Reynolds, "Evaluation of subgrid-scale models using an accurately simulated turbulent flow," J. Fluid Mech. **91**, 1–16 (1979).

[19] R. Maulik, O. San, A. Rasheed, and P. Vedula, "Data-driven deconvolution for large eddy simulations of Kraichnan turbulence," Phys. Fluids **30**, 125109 (2018).

[20] Z. Yuan, Y. Wang, C. Xie, and J. Wang, "Dynamic iterative approximate deconvolution models for large-eddy simulation of turbulence," Phys. Fluids **33**, 085125 (2021).

[21] S. L. Brunton, B. R. Noack, and P. Koumoutsakos, "Machine learning for fluid mechanics," Annu. Rev. Fluid Mech. **52**, 477–508 (2020).

[22] K. Duraisamy, G. Iaccarino, and H. Xiao, "Turbulence modeling in the age of data," Annu. Rev. Fluid Mech. **51**, 357–377 (2019).

[23] M. Lienen, J. Hansen-Palmus, D. Lüdke, and S. Günnemann, "Generative diffusion for 3D turbulent flows," arXiv:2306.01776 (2023).

[24] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature **521**, 436–444 (2015).

[25] Y. Wang, Z. Yuan, C. Xie, and J. Wang, "Artificial neural network-based spatial gradient models for large-eddy simulation of turbulence," AIP Adv. **11**, 055216 (2021).

[26] A. Beck, D. Flad, and C.-D. Munz, "Deep neural networks for data-driven turbulence models," in *Proceedings of the APS Division of Fluid Dynamics Meeting Abstracts* (American Physical Society, 2019), p. G16.

[27] C. Xie, J. Wang, K. Li, and C. Ma, "Artificial neural network approach to large-eddy simulation of compressible isotropic turbulence," Phys. Rev. E **99**, 053113 (2019).

[28] Z. Wang, K. Luo, D. Li, J. Tan, and J. Fan, "Investigations of data-driven closure for subgrid-scale stress in large-eddy simulation," Phys. Fluids **30**, 125101 (2018).

[29] M. Gamahara and Y. Hattori, "Searching for turbulence models by artificial neural network," Phys. Rev. Fluids **2**, 054604 (2017).

[30] Z. Zhou, G. He, S. Wang, and G. Jin, "Subgrid-scale model for large-eddy simulation of isotropic turbulent flows using an artificial neural network," Comput. Fluids **195**, 104319 (2019).

[31] H. Li, Y. Zhao, J. Wang, and R. D. Sandberg, "Data-driven model development for large-eddy simulation of turbulence using gene-expression programing," Phys. Fluids **33**, 125127 (2021).

[32] J. Ling, A. Kurzawski, and J. Templeton, "Reynolds averaged turbulence modelling using deep neural networks with embedded invariance," J. Fluid Mech. **807**, 155–166 (2016).

[33] Y. Guan, A. Chattopadhyay, A. Subel, and 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**, 111090 (2022).

[34] R. Han, Y. Wang, Y. Zhang, and G. Chen, "A novel spatial-temporal prediction method for unsteady wake flows based on hybrid deep neural network," Phys. Fluids **31**, 127101 (2019).

[35] S. Cai, Z. Mao, Z. Wang, M. Yin, and G. E. Karniadakis, "Physics-informed neural networks (PINNs) for fluid mechanics: A review," Acta Mech. Sin. **37**, 1727–1738 (2021).

[36] S. Lanthaler, S. Mishra, and G. E. Karniadakis, "Error estimates for DeepONets: A deep learning framework in infinite dimensions," Trans. Math. Appl. **6**, tnac001 (2022).

[37] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, "Physics-informed machine learning," Nat. Rev. Phys. **3**, 422–440 (2021).

[38] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics informed deep learning (Part I): Data-driven solutions of nonlinear partial differential equations," arXiv:1711.10561 (2017).

[39] X. I. A. Yang, S. Zafar, J.-X. Wang, and H. Xiao, "Predictive large-eddy-simulation wall modeling via physics-informed neural networks," Phys. Rev. Fluids **4**, 034602 (2019).

[40] M. Raissi, P. Perdikaris, and 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**, 686–707 (2019).

[41] Y. Chen, D. Huang, D. Zhang, J. Zeng, N. Wang, H. Zhang, and J. Yan, "Theory-guided hard constraint projection (HCP): A knowledge-based data-driven scientific machine learning method," J. Comput. Phys. **445**, 110624 (2021).

[42] X. Jin, S. Cai, H. Li, and G. E. Karniadakis, "NSFnets (Navier–Stokes flow nets): Physics-informed neural networks for the incompressible Navier–Stokes equations," J. Comput. Phys. **426**, 109951 (2021).

[43] K. Wu and D. Xiu, "Data-driven deep learning of partial differential equations in modal space," J. Comput. Phys. **408**, 109307 (2020).

[44] H. Xu, D. Zhang, and J. Zeng, "Deep-learning of parametric partial differential equations from sparse and noisy data," Phys. Fluids **33**, 037132 (2021).

[45]L. Lu, X. Meng, S. Cai, Z. Mao, S. Goswami, Z. Zhang, and G. E. Karniadakis, "A comprehensive and fair comparison of two neural operators (with practical extensions) based on fair data," Comput. Methods Appl. Mech. Eng. **393**, 114778 (2022).

[46]N. Kovachki, Z. Li, B. Liu, K. Azizzadenesheli, K. Bhattacharya, A. Stuart, and A. Anandkumar, "Neural operator: Learning maps between function spaces with applications to PDEs," J. Mach. Learn. Res. **24**, 1–97 (2023).

[47]S. Goswami, K. Kontolati, M. D. Shields, and G. E. Karniadakis, "Deep transfer learning for partial differential equations under conditional shift with DeepONet," arXiv:2204.09810 (2022).

[48]Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar, "Fourier neural operator for parametric partial differential equations," arXiv:2010.08895 (2020).

[49]J. Chen, J. Viquerat, and E. Hachem, "U-net architectures for fast prediction of incompressible laminar flows," arXiv:1910.13532 (2019).

[50]R. Wang, K. Kashinath, M. Mustafa, A. Albert, and R. Yu, "Towards physics-informed deep learning for turbulent flow prediction," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Association for Computing Machinery, 2020), pp. 1457–1466.

[51]K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2016), pp. 770–778.

[52]G. Wen, Z. Li, K. Azizzadenesheli, A. Anandkumar, and S. M. Benson, "U-FNO—An enhanced Fourier neural operator-based deep-learning model for multiphase flow," Adv. Water Resour. **163**, 104180 (2022).

[53]A. Choubineh, J. Chen, D. A. Wood, F. Coenen, and F. Ma, "Fourier neural operator for fluid flow in small-shape 2D simulated porous media dataset," Algorithms **16**, 24 (2023).

[54]W. Peng, Z. Yuan, and J. Wang, "Attention-enhanced neural network models for turbulence simulation," Phys. Fluids **34**, 025111 (2022).

[55]Z. Li, D. Z. Huang, B. Liu, and A. Anandkumar, "Fourier neural operator with learned deformations for PDEs on general geometries," arXiv:2207.05209 (2022).

[56]Z. Jiang, M. Zhu, D. Li, Q. Li, Y. O. Yuan, and L. Lu, "Fourier-MIONet: Fourier-enhanced multiple-input neural operators for multiphase modeling of geological carbon sequestration," arXiv:2303.04778 (2023).

[57]A. Tran, A. Mathews, L. Xie, and C. S. Ong, "Factorized Fourier neural operators," arXiv:2111.13802 (2021).

[58]P. I. Renn, C. Wang, S. Lale, Z. Li, A. Anandkumar, and M. Gharib, "Forecasting subcritical cylinder wakes with Fourier Neural Operators," arXiv:2301.08290 (2023).

[59]Z. Li, H. Zheng, N. Kovachki, D. Jin, H. Chen, B. Liu, K. Azizzadenesheli, and A. Anandkumar, "Physics-informed neural operator for learning partial differential equations," arXiv:2111.03794 (2021).

[60]J. Guibas, M. Mardani, Z. Li, A. Tao, A. Anandkumar, and B. Catanzaro, "Adaptive Fourier neural operators: Efficient token mixers for transformers," arXiv:2111.13587 (2021).

[61]Z. Hao, C. Ying, Z. Wang, H. Su, Y. Dong, S. Liu, Z. Cheng, J. Zhu, and J. Song, "GNOT: A general neural operator transformer for operator learning," arXiv:2302.14376 (2023).

[62]J. A. L. Benitez, T. Furuya, F. Faucher, X. Tricoche, and M. V. de Hoop, "Fine-tuning neural-operator architectures for training and generalization," arXiv:2301.11509 (2023).

[63]D. Meng, Y. Zhu, J. Wang, and Y. Shi, "Fast flow prediction of airfoil dynamic stall based on Fourier neural operator," Phys. Fluids **35**, 115126 (2023).

[64]Z. Deng, H. Liu, B. Shi, Z. Wang, F. Yu, Z. Liu, and G. Chen, "Temporal predictions of periodic flows using a mesh transformation and deep learning-based strategy," Aerosp. Sci. Technol. **134**, 108081 (2023).

[65]M. Momenifar, E. Diao, V. Tarokh, and A. D. Bragg, "Dimension reduced turbulent flow data from deep vector quantisers," J. Turbul. **23**, 232–264 (2022).

[66]A. T. Mohan, D. Tretiak, M. Chertkov, and D. Livescu, "Spatio-temporal deep learning models of 3D turbulence with physics informed diagnostics," J. Turbul. **21**, 484–524 (2020).

[67]T. Nakamura, K. Fukami, K. Hasegawa, Y. Nabae, and K. Fukagata, "Convolutional neural network and long short-term memory based reduced order surrogate for minimal turbulent channel flow," Phys. Fluids **33**, 025116 (2021).

[68]Z. Li, W. Peng, Z. Yuan, and J. Wang, "Fourier neural operator approach to large eddy simulation of three-dimensional turbulence," Theor. Appl. Mech. Lett. **12**, 100389 (2022).

[69]W. Peng, Z. Yuan, Z. Li, and J. Wang, "Linear attention coupled Fourier neural operator for simulation of three-dimensional turbulence," Phys. Fluids **35**, 015106 (2023).

[70]Z. Li, W. Peng, Z. Yuan, and J. Wang, "Long-term predictions of turbulence by implicit U-Net enhanced Fourier neural operator," Phys. Fluids **35**, 075145 (2023).

[71]Z. Li, N. B. Kovachki, C. Choy, B. Li, J. Kossaifi, S. P. Otta, M. A. Nabian, M. Stadler, C. Hundt, K. Azizzadenesheli *et al.*, "Geometry-informed neural operator for large-scale 3D PDEs," arXiv:2309.00583 (2023).

[72]A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," Proc. Adv. Neural Inf. Process. Syst. **30**, 6000 (2017), https://proceedings.neurips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

[73]N. S. Keskar, B. McCann, L. R. Varshney, C. Xiong, and R. Socher, "CTRL: A conditional transformer language model for controllable generation," arXiv:1909.05858 (2019).

[74]Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov, "Transformer-XL: Attentive language models beyond a fixed-length context," arXiv:1901.02860 (2019).

[75]Z. Li, K. Meidani, and A. B. Farimani, "Transformer for partial differential equations' operator learning," arXiv:2205.13671 (2022).

[76]D. Drikakis, I. W. Kokkinakis, D. Fung, and S. M. Spottswood, "Generalizability of transformer-based deep learning for multidimensional turbulent flow data," Phys. Fluids **36**, 026102 (2024).

[77]Q. Xu, Z. Zhuang, Y. Pan, and B. Wen, "Super-resolution reconstruction of turbulent flows with a transformer-based deep learning framework," Phys. Fluids **35**, 055130 (2023).

[78]M. Momenifar, E. Diao, V. Tarokh, and A. D. Bragg, "Emulating spatio-temporal realizations of three-dimensional isotropic turbulence via deep sequence learning models," arXiv:2112.03469 (2021).

[79]K. Bi, L. Xie, H. Zhang, X. Chen, X. Gu, and Q. Tian, "Pangu-weather: A 3D high-resolution model for fast and accurate global weather forecast," arXiv:2211.02556 (2022).

[80]Y. Wang, A. Solera-Rico, C. S. Vila, and R. Vinuesa, "Towards optimal β-variational autoencoders combined with transformers for reduced-order modelling of turbulent flows," Int. J. Heat Fluid Flow **105**, 109254 (2024).

[81]S. Janny, A. Beneteau, N. Thome, M. Nadri, J. Digne, and C. Wolf, "Eagle: Large-scale learning of turbulent fluid dynamics with mesh transformers," arXiv:2302.10803 (2023).

[82]A. Patil, J. Viquerat, and E. Hachem, "Autoregressive transformers for data-driven spatio-temporal learning of turbulent flows," arXiv:2209.08052 (2022).

[83]Y. Dang, Z. Hu, M. Cranmer, M. Eickenberg, and S. Ho, "TNT: Vision transformer for turbulence simulations," arXiv:2207.04616 (2022).

[84]Z. Li, D. Shu, and A. B. Farimani, "Scalable transformer for PDE surrogate modeling," arXiv:2305.17560 (2023).

[85]J. Jiang, G. Li, Y. Jiang, L. Zhang, and X. Deng, "TransCFD: A transformer-based decoder for flow field prediction," Eng. Appl. Artif. Intell. **123**, 106340 (2023).

[86]S. B. Pope and S. B. Pope, *Turbulent Flows* (Cambridge University Press, 2000).

[87]T. Ishihara, T. Gotoh, and Y. Kaneda, "Study of high–Reynolds number isotropic turbulence by direct numerical simulation," Annu. Rev. Fluid Mech. **41**, 165–180 (2009).

[88]Y. Wang, Z. Yuan, X. Wang, and J. Wang, "Constant-coefficient spatial gradient models for the sub-grid scale closure in large-eddy simulation of turbulence," Phys. Fluids **34**, 095108 (2022).

[89]P. Sagaut, *Large Eddy Simulation for Incompressible Flows: An Introduction* (Springer Science & Business Media, 2006).

[90]R. D. Moser, S. W. Haering, and G. R. Yalla, "Statistical properties of subgrid-scale turbulence models," Annu. Rev. Fluid Mech. **53**, 255–286 (2021).

[91]P. L. Johnson, "A physics-inspired alternative to spatial filtering for large-eddy simulations of turbulent flows," J. Fluid Mech. **934**, A30 (2022).

[92]D. K. Lilly, "A proposed modification of the Germano subgrid-scale closure method," Phys. Fluids A **4**, 633–635 (1992).

28 November 2025 06:44:33

[93]S. Liu, C. Meneveau, and J. Katz, "On the properties of similarity subgrid-scale models as deduced from measurements in a turbulent jet," J. Fluid Mech. **275**, 83–119 (1994).

[94]Z. Yuan, C. Xie, and J. Wang, "Deconvolutional artificial neural network models for large eddy simulation of turbulence," Phys. Fluids **32**, 115106 (2020).

[95]B. Beauzamy, *Introduction to Banach Spaces and Their Geometry* (Elsevier, 2011).

[96]V. N. Vapnik, "An overview of statistical learning theory," IEEE Trans. Neural Network **10**, 988–999 (1999).

[97]Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar, "Neural operator: Graph kernel network for partial differential equations," arXiv:2003.03485 (2020).

[98]M. M. Rashid, T. Pittie, S. Chakraborty, and N. A. Krishnan, "Learning the stress-strain fields in digital composites using Fourier neural operator," iScience **25**, 105452 (2022).

[99]J. Pathak, S. Subramanian, P. Harrington, S. Raja, A. Chattopadhyay, M. Mardani, T. Kurth, D. Hall, Z. Li, K. Azizzadenesheli *et al.*, "Fourcastnet: A global data-driven high-resolution weather model using adaptive Fourier neural operators," arXiv:2202.11214 (2022).

[100]J. Su, M. Ahmed, Y. Lu, S. Pan, W. Bo, and Y. Liu, "Roformer: Enhanced transformer with rotary position embedding," Neurocomputing **568**, 127063 (2024).

[101]M. Y. Hussaini and T. A. Zang, "Spectral methods in fluid dynamics," Annu. Rev. Fluid Mech. **19**, 339–367 (1987).

[102]N. Chang, Z. Yuan, and J. Wang, "The effect of sub-filter scale dynamics in large eddy simulation of turbulence," Phys. Fluids **34**, 095104 (2022).

[103]Z. Zhuang, M. Liu, A. Cutkosky, and F. Orabona, "Understanding AdamW through proximal methods and scale-freeness," Trans. Mach. Learn. Res. (published online 2022), https://openreview.net/forum?id=IKhEPWGdwK.

[104]I. Loshchilov and F. Hutter, "Fixing weight decay regularization in Adam," arXiv:1711.05101 (2018).

[105]D. Hendrycks and K. Gimpel, "Gaussian error linear units (GELUs)," arXiv:1606.08415 (2016).

[106]C. Xie, J. Wang, H. Li, M. Wan, and S. Chen, "A modified optimal LES model for highly compressible isotropic turbulence," Phys. Fluids **30**, 065108 (2018).

[107]C. Xie, J. Wang, H. Li, M. Wan, and S. Chen, "An approximate second-order closure model for large-eddy simulation of compressible isotropic turbulence," Commun. Comput. Phys. **27**, 775–808 (2020).

[108]J. Kleissl, V. Kumar, C. Meneveau, and M. B. Parlange, "Numerical study of dynamic Smagorinsky models in large-eddy simulation of the atmospheric boundary layer: Validation in stable and unstable conditions," Water Resour. Res. **42**, W06D10, https://doi.org/10.1029/2005WR004685 (2006).

[109]X. Wang, J. Wang, and S. Chen, "Compressibility effects on statistics and coherent structures of compressible turbulent mixing layers," J. Fluid Mech. **947**, A38 (2022).

[110]N. Sharan, G. Matheou, and P. E. Dimotakis, "Turbulent shear-layer mixing: Initial conditions, and direct-numerical and large-eddy simulations," J. Fluid Mech. **877**, 35–81 (2019).

[111]Z. Yuan, Y. Wang, X. Wang, and J. Wang, "Adjoint-based variational optimal mixed models for large-eddy simulation of turbulence," Phys. Fluids **35**, 075105 (2023).

[112]M. M. Rogers and R. D. Moser, "Direct simulation of a self-similar turbulent mixing layer," Phys. Fluids **6**, 903–923 (1994).

[113]Y. Dubief and F. Delcayre, "On coherent-vortex identification in turbulence," J. Turbul. **1**, 011 (2000).

[114]J. Zhan, Y. Li, W. O. Wai, and W. Hu, "Comparison between the Q criterion and Rortex in the application of an in-stream structure," Phys. Fluids **31**, 121701 (2019).

[115]T. Kurth, S. Subramanian, P. Harrington, J. Pathak, M. Mardani, D. Hall, A. Miele, K. Kashinath, and A. Anandkumar, "FourCastNet: Accelerating global high-resolution weather forecasting using adaptive Fourier neural operators," arXiv:2208.05419 (2022).

[116]B. Alkin, A. Fürst, S. Schmid, L. Gruber, M. Holzleitner, and J. Brandstetter, "Universal physics transformers," arXiv:2402.12365 (2024).

[117]L. Z. Zhao, X. Ding, and B. A. Prakash, "Pinnsformer: A transformer-based framework for physics-informed neural networks," arXiv:2307.11833 (2023).

28 November 2025 06:44:33