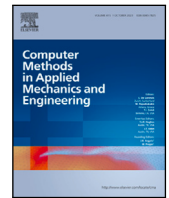




Contents lists available at ScienceDirect

## Comput. Methods Appl. Mech. Engrg.

journal homepage: [www.elsevier.com/locate/cma](http://www.elsevier.com/locate/cma)

# Bayesian conditional diffusion models for versatile spatiotemporal turbulence generation<sup>☆</sup>

Han Gao<sup>a,c,1</sup>, Xu Han<sup>d,1</sup>, Xiantao Fan<sup>a</sup>, Luning Sun<sup>a,e</sup>, Li-Ping Liu<sup>d</sup>, Lian Duan<sup>f</sup>, Jian-Xun Wang<sup>a,b,\*</sup>

<sup>a</sup> Aerospace and Mechanical Engineering Department, University of Notre Dame, Notre Dame, IN, USA

<sup>b</sup> Lucy Family Institute for Data & Society, University of Notre Dame, Notre Dame, IN, USA

<sup>c</sup> School of Engineering and Applied Science, Harvard University, Cambridge, MA, USA

<sup>d</sup> Department of Computer Science, Tufts University, Medford, MA, USA

<sup>e</sup> Lawrence Livermore National Laboratory, Livermore, CA, USA

<sup>f</sup> Department of Mechanical and Aerospace Engineering Department, The Ohio State University, Columbus, OH, USA

## ARTICLE INFO

### Keywords:

Turbulent flow  
Generative modeling  
Bayesian statistics  
Surrogate modeling  
Wall-bounded turbulence  
Chaotic dynamics

## ABSTRACT

Turbulent flows, characterized by their chaotic and stochastic nature, have historically presented formidable challenges to predictive computational modeling. Traditional eddy-resolved numerical simulations often require vast computational resources, making them impractical or infeasible for numerous engineering applications. As an alternative, deep learning-based surrogate models have emerged, offering data-drive solutions. However, these are typically constructed within deterministic settings, leading to shortfall in capturing the innate chaotic and stochastic behaviors of turbulent dynamics. In this study, we introduce a novel generative framework grounded in probabilistic diffusion models for versatile generation of spatiotemporal turbulence under various conditions. Our method unifies both unconditional and conditional sampling strategies within a Bayesian framework, which can accommodate diverse conditioning scenarios, including those with a direct differentiable link between specified conditions and generated unsteady flow outcomes, as well as scenarios lacking such explicit correlations. A notable feature of our approach is the method proposed for long-span flow sequence generation, which is based on autoregressive gradient-based conditional sampling, eliminating the need for cumbersome retraining processes. We evaluate and showcase the versatile turbulence generation capability of our framework through a suite of numerical experiments, including: (1) the synthesis of Large Eddy Simulations (LES) simulated instantaneous flow sequences from unsteady Reynolds-Averaged Navier–Stokes (URANS) inputs; (2) holistic generation of inhomogeneous, anisotropic wall-bounded turbulence, whether from given initial conditions, prescribed turbulence statistics, or entirely from scratch; (3) super-resolved generation of high-speed turbulent boundary layer flows from low-resolution data across a range of input resolutions. Collectively, our numerical experiments highlight the merit and transformative potential of the proposed methods, making a significant advance in the field of turbulence generation.

<sup>☆</sup> Videos of all numerical experiments can be found at <https://sites.nd.edu/jianxun-wang/animations>.

\* Corresponding author at: Aerospace and Mechanical Engineering Department, University of Notre Dame, Notre Dame, IN, USA.

E-mail address: [jwang33@nd.edu](mailto:jwang33@nd.edu) (J.-X. Wang).

<sup>1</sup> Contributed equally.

<https://doi.org/10.1016/j.cma.2024.117023>

Received 14 January 2024; Received in revised form 26 March 2024; Accepted 23 April 2024

0045-7825/© 2024 Elsevier B.V. All rights reserved.

## 1. Introduction

Turbulent flows, ubiquitous in diverse contexts such as high-speed aircraft operation, oceanic currents, and combustion processes, exhibit complex unsteady and chaotic behaviors, characterized by swirling vortices and eddies over a wide spectrum of scales. To investigate and simulate these intricate phenomena, researchers often resort to numerical solutions of the governing partial differential equations (PDEs) of fluid dynamics. Eddy-resolved numerical simulations, notably Direct Numerical Simulation (DNS) and Large Eddy Simulation (LES), aims to accurately capture the unsteady/chaotic dynamics inherent in turbulent structures across extensive spatiotemporal scales. However, the computational demands of such methods can easily escalate to levels that are prohibitively expensive or practically infeasible. This presents significant challenges, particularly for tasks requiring rapid turnarounds, like real-time forecasting, or those that necessitating repeated model evaluations, as observed in design optimization and uncertainty quantification.

Recent advances in deep learning (DL) offer new perspectives in addressing the intrinsic challenges of simulating unsteady turbulent flows. Over the past few years, there has been a surge of interest in harnessing the power of DL to enhance computational fluid dynamics (CFD) capabilities [1,2]. In this context, various DL-augmented CFD frameworks have emerged, wherein deep neural networks (DNNs) are integrated with conventional CFD algorithms, enabling functionalities such as discovering high-order discretizations [3,4], refining coarse-grid computations [5,6], learning turbulence closure models [7–9], and accelerating pressure projection solvers [10]. Beyond enhancing existing CFD techniques, DL plays a pivotal role in creating rapid surrogate or reduced-order models, which serve as substitutes for the computationally-intensive numerical solvers for emulating spatiotemporal flow physics [11]. These innovations underscore the potential of DL in enhancing the accuracy and computational efficiency of fluid simulations. Despite these advancements, several challenges persist when it comes to turbulence simulation and prediction. A major concern is the reliability and robustness of DL-based solution, especially over a long time span. The chaotic nature of turbulence means that even minor perturbations or modeling inaccuracies can lead to significant deviations in long-term predictions. Thus, the deterministic nature of many DL-based fluid models often falls short in addressing the stochasticity intrinsic to turbulent flows.

In response, generative modeling, grounded in probabilistic frameworks, emerges as a promising alternative. In the realm of data science, generative models encompass a set of algorithms designed to distill complex distributions of data. By leveraging latent representations [12], adversarial learning schemes [13], or sequential data generation techniques [14], these models provide a comprehensive toolkit for a variety of academic and industrial applications, including but not limited to image generation, language processing, and anomaly detection [15]. The core of generative models lies in the objective of learning the underlying probability distributions of the data, which allows the synthesis of new data samples that adhere to the statistical properties inherent in the training set. In the context of turbulent flows, this implies the ability to generate instantaneous flow field realizations that mirror the same statistical characteristics of the observed turbulence data, thereby bypassing the need for costly CFD simulations. In recent years, there has been a growing interest in developing DL-based generative models for turbulent flows, motivated by their potential in flow reconstruction and super-resolution, synthetic inflow generation, and surrogate turbulent flow emulation. These existing methods can be broadly classified into following categories,

**Autoregressive sequence models.** Existing DL-based methods for turbulence generation are mainly based on autoregressive learning architecture. These methods aim to learn the dynamic evolution of turbulent flow via neural networks based on labeled data. After training, these models can take flow fields from preceding time step as input, outputting flow predictions for subsequent steps. By rolling out the trained model given specified initial conditions, the sequence of turbulent flow fields can be synthesized in an autoregressive manner. These approaches mainly rely on the learned temporal correlations within the flow data, enabling the DL models to operate similarly to conventional explicit numerical solver. To address the challenge of handling high-dimensional turbulent flow data, these methods often incorporate dimensionality reduction techniques, such as Proper Orthogonal Decomposition (POD), Convolutional neural network (CNN) autoencoders, in conjunction with sequence networks. For instance, Fukami et al. [16–18] developed convolutional autoencoder-based autoregressive learning models for inflow turbulence synthesis, super-resolution or surrogate flow predictions. Yousif et al. [19] combined CNN autoencoder with Long Short-Term Memory (LSTM) to achieve the similar goals, and their models have been further extended by introducing adversarial training and attention mechanisms [20]. However, it is important to emphasize that these autoregressive neural forecasting models intrinsically operate deterministically as they do not learn the underlying distribution of turbulence data, rendering them incapable of arbitrarily generating instantaneous turbulent flow realizations as a stochastic process. Furthermore, such deterministic autoregressive models are susceptible to cumulative error propagation, compromising the robustness of their long-term rollouts, with predictions either escalating unpredictably or dampening important flow features converged to time-averaged values. The inherently chaotic nature of turbulence exacerbates this vulnerability, leading to failures for long-term predictions. Uncertainty propagation in these models have been recently explored [21,22].

**GAN-based generative models.** To actually learn the underlying probability distribution of turbulent flow data, allowing for the random sampling of new realizations, Generative Adversarial Networks (GANs) have emerged as an effective tool. GANs rely on a competitive dynamic between generator and discriminator: the former generates synthetic turbulent data, while the latter differentiates synthetic from labeled data. Through iterative adversarial training, the outputs of the generator are progressively refined, targeting convergence to the true distribution of the training data. Recent studies have demonstrated the applicability of GANs in turbulent flow generation, super-resolution and inpainting. Several GAN variants, including the Wasserstein GAN (WGAN), conditional GAN (cGAN), and deep convolutional GAN (DCGAN), have been adapted to synthesize individual snapshots of both homogeneous isotropic turbulence and wall-bounded turbulent flows [23,24]. In the super-resolution (SR) context, methods such as

super-resolution GAN (SRGAN), enhanced SRGAN, and cycle-consistent GAN (CycGAN) have been employed to enhance the low-resolution or low-fidelity 2D/3D turbulence snapshots [25–28]. Buzzicotti et al. [29] leverage GAN to generate missing turbulence data similar to image inpainting. However, these GAN-based models primarily focus on single-snapshot generation. Since such GANs are trained on isolated flow snapshots (treated as 2D or 3D spatial images without temporal coherence), they intrinsically lack the capacity for sequential spatiotemporal turbulent flow synthesis. While certain research endeavors have integrated GAN with sequential networks [20]; however, these GANs largely serve as deterministic encoders, with their adversarial training paradigm remaining decoupled from sequential networks, limiting their true stochastic generative potential. Only a few studies have rigorously explored the potential of GANs to approximate the probabilistic distribution of flow sequence data without input–output labels for training. Notably, Xie et al. [30] proposed TempoGAN, a combination of GAN and RNN, designed to generate stochastic, temporally consistent fluid simulations with SR capability. Similar to TempoGAN, Kim and Lee [31] developed a combined WGAN+RNN model to generate 2D inlet turbulence for 3D channel flow simulations. Despite the promise of GANs in turbulent flow synthesis, they face several challenges: (1) the training of GAN can often be notoriously unstable, leading to frequent oscillations between the generator and discriminator [32]; (2) moreover, GANs also often suffer from “mode collapse”, producing limited varieties of outputs [33].

**Normalizing flow (NF)-based generative models.** Normalizing flows (NFs) have emerged as another notable subclass of generative models. At their core, NFs transform a basic, predefined probability distribution through a series of invertible, differentiable functions, morphing it into a complex one that approximates the underlying data distribution. The strength of NFs is their potential to model a complex distribution directly, bypassing the need for sampling-based approximations. Conceptually, this process of transformation resembles a continuous flow, progressively shaping the initial distribution to align with the target data distribution. Geneva and Zabaras [34] proposed a multi-fidelity deep generative model using NFs to simulate turbulent flows at different Reynolds numbers. Efficient low-fidelity solver is leveraged to generate high-fidelity solution, significantly reducing the computational costs. Their model incorporates a conditional invertible neural network with LSTM connections, trained through both data and governing equations using a variational loss function. Another recent study integrated an attention-based sequence model into the NF-based generative architecture, enabling the probabilistic synthesis of time-evolving turbulence [35]. Despite their promise, NFs are known to have scalability issues with high-dimensional data, like turbulent flows, due to the requirement of computing Jacobians for the transformations. Furthermore, developing conditional variants of NFs is notably challenging, limiting their adaptation to new turbulent flow conditions.

**Diffusion-based generative models.** Diffusion models have recently marked a significant footprint in generative modeling, demonstrating superior capabilities compared to GANs or NFs in multiple computer vision applications [36–38]. These models typically involve a noise-addition process followed by a reverse denoising operation via deep neural networks. Categorically, diffusion models can be classified into (i) Denoising Diffusion Probabilistic Models (DDPMs) [37], (ii) Score-based diffusion models [36], and (iii) Stochastic Differential Equation (SDE) based models [39], with the latter considered as an overarching framework for the former two. Diffusion generative models offer three primary advantages. First, their training process is straightforward, demanding minimal architecture and hyperparameter fine-tuning. Second, their hidden variables mirror the physical characteristics of samples, enabling seamless multi-scale feature capturing. Lastly, operating within a Bayesian framework, they provide a more direct way to enable conditioned generation compared to normalizing flow-based models. Despite their evident success in fields like image generation and super-resolution, their application in the physics domain, particularly turbulence modeling, remains limited. Some recent works have made preliminary inroads on 2D Kolmogorov flows, targeting specific sub-tasks like super-resolution [40–42]. However, a comprehensive application of diffusion models for spatiotemporal generation of inhomogeneous, anisotropic turbulence, particularly via conditional frameworks, remains a largely uncharted territory. A gap exists in systematically generating intricate turbulent flows under various conditional inputs, such as initial fields, boundary conditions, turbulence statistics, external forces, etc.

To this end, this work delves into the development and systematic investigation of conditional diffusion models designed for generating complex inhomogeneous and anisotropic turbulent flows under various conditions. Unlike the majority of existing literature on generative modeling that are primarily limited to single-snapshot generation, our emphasis lies in capturing the underlying probabilistic distribution of time-evolving turbulent flow sequences. Essentially, our diffusion model can be conceptualized as a stochastic spatiotemporal process, thereby allowing to generate new realizations of instantaneous turbulent flow sequences through randomly sampling. Furthermore, to enhance the versatility of our generator, we integrate advanced Bayesian conditional sampling techniques. By conditioning the generation (sampling) process on condition parameters such as initial flow fields, boundary conditions, turbulent statistics, and coarse-grained solutions, our model adeptly synthesizes turbulent flows tailored to specific constraints. The proposed Bayesian conditional sampling approach not only bridges parametric conditions with generated spatial-temporal turbulent flows, but also magnifies the model's flexibility, making it highly adaptable to a range of real-world scenarios. As depicted in Fig. 1, the versatility of our proposed method is evident across a variety of turbulence generation scenarios. In the context of 2D unsteady vortex flows over a backward-facing step, the model can reproduce LES-like eddy-resolved turbulence given the corresponding low-fidelity URANS simulated solutions. For 3D turbulent channel flows, the figure emphasizes the model's capability at generating spatiotemporal sequences from varied initial conditions, specific flow statistics, or even entirely from scratch. Furthermore, the 3D compressible supersonic turbulent boundary layers scenario showcased in the figure showcase our model's ability in super-resolution generation—demonstrating a conditioned generation from low-resolution inputs to high-fidelity DNS of high-speed turbulent boundary layers. Collectively, Fig. 1 illustrates the robust adaptability and comprehensive capability of the proposed conditional diffusion model in addressing diverse turbulence generation challenges.

The **primary contributions** of the paper are summarized as follows: (a) We present a novel generative framework for spatiotemporal turbulent flow, rooted in probabilistic diffusion modeling. This model operates within the Bayesian context, trained using the

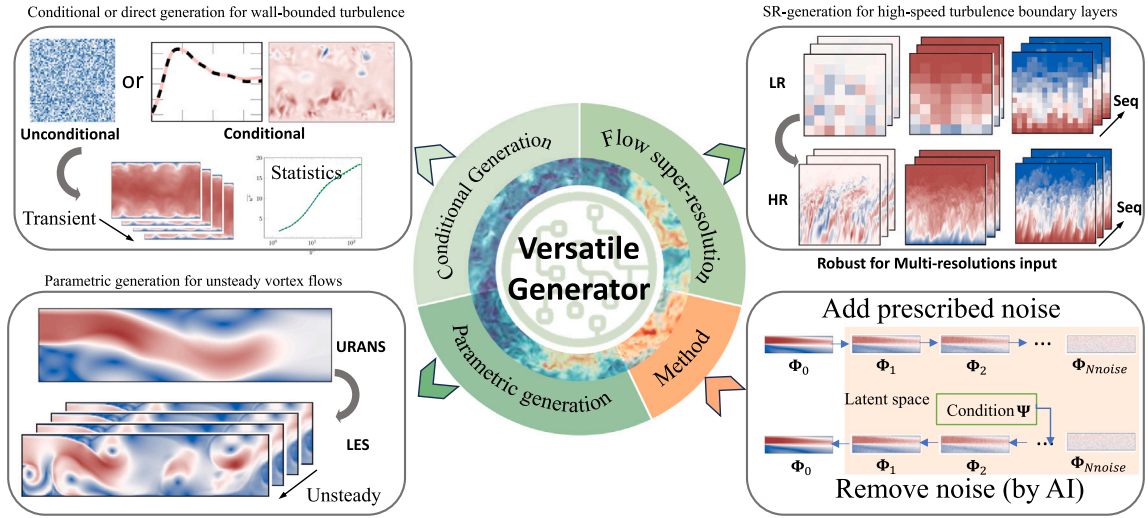


Fig. 1. Overview of the proposed versatile spatiotemporal turbulence generator via conditional diffusion modeling.

evidence lower bound, and is designed to randomly generate temporarily-coherent turbulent flow snapshots over a long-time span. This approach sets it apart from other GAN-based methods that typically focus on generating single snapshots. (b) We proposed a systematic approach to conditional generation, addressing two distinct scenarios of conditioning. The first establishes a differentiable relationship between specified conditions and the corresponding spatial-temporal turbulence solutions. Notably, we leverage the gradient from this connection via automatic differentiation to directly modify the unconditional score functions, allowing conditional sampling without necessitating model retraining. For the second scenario, where conditions lack a direct correlation to the turbulent flow, we treat them as additional inputs to our diffusion model. With transfer-retraining, the conditioned probability distribution can be learned from conditioned datasets. (c) We proposed autoregressive conditioned generation techniques that enables long-span flow generation. Its efficacy is benchmarked against the replacement method by Ho et al. [43] designed for prolonged video creation. (d) Our framework, when conditioned on URANS simulations, is capable of generating LES-like solutions with unsteady dynamics that are not captured by URANS, including phenomena such as Kelvin-Helmholtz instability and vortex shedding. (e) Through a series of comprehensive experiments, we evaluate the performance of the proposed method in generating anisotropic, inhomogenous wall-bounded turbulence, under different given conditions. These generated flows, both from unconditional and gradient-based conditional sampling, are benchmarked against DNS results to affirm their fidelity and accuracy. (f) We also showcased our model's versatility through its application in super-resolution tasks for a high-speed turbulent boundary layer flow, with a particular emphasis on handling various input resolutions.

The remainder of the paper is structured as follows. Section 2 delves into the methodology of the our diffusion-model-based turbulence generator. Specifically, the formulation of the variational diffusion model for spatiotemporal turbulent flows is presented in Section 2.1, while the unconditional and conditional sampling methods are discussed in Sections 2.2 and 2.3, respectively. The generation of extended flow sequences using autoregressive gradient-based conditional sampling, which circumvents re-training, is detailed in Section 2.4. In Section 3, we present a comprehensive set of numerical experiments to demonstrate and evaluate the capability of the proposed method in diverse turbulence generation contexts. This section showcases: (1) the model's proficiency in simulating 2D unsteady flows over a backward-facing step, especially in generating LES-like turbulent flows from URANS solutions; (2) Its capability in generating wall-bounded turbulence, whether from specified initial conditions, statistical data, or from scratch; (3) Its ability in SR generation for compressible supersonic turbulent boundary layer flows, demonstrating resilience to diverse input resolutions. Finally, the paper is concluded in Section 4, summarizing our principal findings and projecting avenues for subsequent enhancements.

## 2. Methodology

### 2.1. Learning probability distribution of spatiotemporal turbulence via variational diffusion models

Turbulent flows are intrinsically characterized as stochastic spatiotemporal processes, reflecting their chaotic and random nature across spatial and temporal scales. In this context, a generative model aims to capture and reproduce these complex dynamics by learning the underlying probability distribution  $p(\Phi_0)$  of the spatiotemporal turbulent flow state variables  $\Phi_0(\mathbf{x}, t)$ , which can be discretized into a sequence of spatial fields  $\Phi_0 \in \mathbb{R}^{N_{\text{length}} \times N_{\text{dof}}}$ , encompassing  $N_{\text{length}}$  snapshots, with each having  $N_{\text{dof}}$  degree of freedom. Our objective, therefore, is to construct a learning model  $p_\theta(\Phi_0)$  that approximates the true probability density  $p(\Phi_0)$  by



learning from a training dataset  $\mathcal{A}_{\text{train}} \subset \mathbb{R}^{N_{\text{length}} \times N_{\text{dof}}}$  that contains many realizations of  $\Phi_0$ . This can be theoretically realized by maximizing the likelihood of all training realizations  $\Phi_0 \in \mathcal{A}_{\text{train}}$  as expressed by,

$$\underset{\theta}{\text{maximize}} \sum_{\Phi_0 \in \mathcal{A}_{\text{train}}} \log p_{\theta}(\Phi_0), \quad (1)$$

where  $\theta \in \mathbb{R}^{N_{\theta}}$  represents the model parameter vector, such as those in neural networks, which is to be optimized. Specifically, we propose constructing a probabilistic diffusion model [44] that implicitly learns this underlying distribution, enabling rapid sampling (generation) of new realizations of spatiotemporal turbulent flow fields. To this end, we introduce a series of latent variables,  $\Phi_1, \dots, \Phi_{N_{\text{noise}}} \in \mathbb{R}^{N_{\text{length}} \times N_{\text{dof}}}$ , which are generated from the turbulent flow sequence  $\Phi_0$  via a Markovian process,

$$q(\Phi_j | \Phi_i) := \mathcal{N}(\alpha_{j|i} \Phi_i, \sigma_{j|i}^2 I), \quad \text{for any } 0 \leq i < j \leq N_{\text{noise}}, \quad (2)$$

where  $\mathcal{N}(\mathbf{a}, \mathbf{b})$  represents a multi-variant Gaussian distribution with the mean  $\mathbf{a} \in \mathbb{R}^{N_{\text{length}} \times N_{\text{dof}}}$  and covariance matrix  $\mathbf{b} \in \mathbb{R}^{(N_{\text{length}} \times N_{\text{dof}})^2}$ ;  $I \in \mathbb{R}^{(N_{\text{length}} \times N_{\text{dof}})^2}$  is the identity matrix; the set of scalar  $\{\sigma_{j|i}, \alpha_{j|i}\}$  for any  $0 \leq i < j \leq N_{\text{noise}}$  are predefined such that the marginalized distribution  $p(\Phi_{N_{\text{noise}}} | \Phi_0)$  converge to a zero-mean Gaussian distribution  $p(\Phi_{N_{\text{noise}}}) = \mathcal{N}(\mathbf{0}, \sigma_{N_{\text{noise}}|0}^2 I)$ , which is straightforward to sample [45]. The conditional distribution of each perturbed state given the turbulent flow sequence data  $\Phi_0$  is Gaussian,

$$q(\Phi_i | \Phi_0) = \mathcal{N}(\alpha_i \Phi_0, \sigma_i^2 I), \quad (3)$$

where  $\{\alpha_i, \sigma_i\}$  for any  $1 \leq i \leq N_{\text{noise}}$  are scalars that can be analytically derived from the predefined set  $\{\sigma_{j|i}, \alpha_{j|i}\}$  for any  $0 \leq i < j \leq N_{\text{noise}}$  [37,46]. This process is known as the *forward diffusion process*, which is determined by handcrafting the transition kernels to progressively transform the data distribution  $p(\Phi_0)$  into an accessible Gaussian distribution  $p(\Phi_{N_{\text{noise}}})$ .

If this diffusion process can be inverted, such an *reverse diffusion process* would enable the synthesis of new realizations of turbulent flow sequences simply by sampling from the Gaussian distribution  $p(\Phi_{N_{\text{noise}}})$ . However, the reverse transition kernel  $q(\Phi_{i-1} | \Phi_i)$  is not known *a priori* and therefore necessitates learning through neural network parametrization. It is noteworthy that the reverse transition kernel becomes tractable when conditioned on  $\Phi_0$  [46,47],

$$q(\Phi_{i-1} | \Phi_i) = q(\Phi_{i-1} | \Phi_i, \Phi_0) = \frac{q(\Phi_i | \Phi_{i-1}, \Phi_0) q(\Phi_{i-1} | \Phi_0)}{q(\Phi_i | \Phi_0)} \quad \text{for any } 2 \leq i \leq N_{\text{noise}}, \quad (4)$$

where the first equality is from the Markovian property and the second is based on the Bayes rule. If the perturbation step  $\sigma_i^2$  is sufficiently small, the reverse transition kernel is also Gaussian, which can be analytically derived by substituting  $q(\Phi_i | \Phi_{i-1})$  from Eq. (2), and both  $q(\Phi_{i-1} | \Phi_0)$  and  $q(\Phi_i | \Phi_0)$  from (3),

$$q(\Phi_{i-1} | \Phi_i) = q(\Phi_{i-1} | \Phi_i, \Phi_0) = \mathcal{N}(\Phi_{i-1}; \tilde{\mu}(\Phi_i, \Phi_0), \tilde{\sigma}_i^2 I), \quad \text{for any } 2 \leq i \leq N_{\text{noise}}, \quad (5)$$

with mean  $\tilde{\mu}(\Phi_i, \Phi_0) = \frac{\alpha_{i|i-1} \sigma_{i-1}^2}{\sigma_i^2} \Phi_i + \frac{\alpha_{i-1} \sigma_{i|i-1}^2}{\sigma_i^2} \Phi_0$  and covariance  $\tilde{\sigma}_i^2 I = \frac{\sigma_{i|i-1}^2 \sigma_{i-1}^2}{\sigma_i^2} I$  [45]. Given that  $\Phi_0$  represent the generated flow outcomes, which are not known *a priori*, we approximate  $q(\Phi_{i-1} | \Phi_i, \Phi_0)$  by  $p_{\theta}(\Phi_{i-1} | \Phi_i)$  parameterized with trainable  $\theta$ , which can be optimized by maximizing the evidence lower bound (ELBO) given the training set  $\Phi_0 \in \mathcal{A}_{\text{train}}$ . The expression for ELBO is given as,

$$\begin{aligned} \text{ELBO}_{\theta}(\Phi_0) &= \mathbb{E}_{q(\Phi_1 | \Phi_0)} \left[ \log p_{\theta}(\Phi_0 | \Phi_1) \right] \\ &\quad - D_{\text{KL}} \left( q(\Phi_{N_{\text{noise}}} | \Phi_0) \parallel p(\Phi_{N_{\text{noise}}}) \right) \\ &\quad - \sum_{i=2}^{N_{\text{noise}}} \mathbb{E}_{q(\Phi_i | \Phi_0)} \left[ D_{\text{KL}} \left( q(\Phi_{i-1} | \Phi_i, \Phi_0) \parallel p_{\theta}(\Phi_{i-1} | \Phi_i) \right) \right]. \end{aligned} \quad (6)$$

Since the second term  $-D_{\text{KL}}(q(\Phi_{N_{\text{noise}}} | \Phi_0) \parallel p(\Phi_{N_{\text{noise}}}))$  of Eq. (7) is non-trainable and close to zero given Eq. (2), we only need to minimize the following terms,

$$-\mathbb{E}_{q(\Phi_1 | \Phi_0)} \left[ \log p_{\theta}(\Phi_0 | \Phi_1) \right] + \sum_{i=2}^{N_{\text{noise}}} \mathbb{E}_{q(\Phi_i | \Phi_0)} \left[ D_{\text{KL}} \left( q(\Phi_{i-1} | \Phi_i, \Phi_0) \parallel p_{\theta}(\Phi_{i-1} | \Phi_i) \right) \right]. \quad (7)$$

To parameterize  $p_{\theta}$ , we use the Eq. (5)

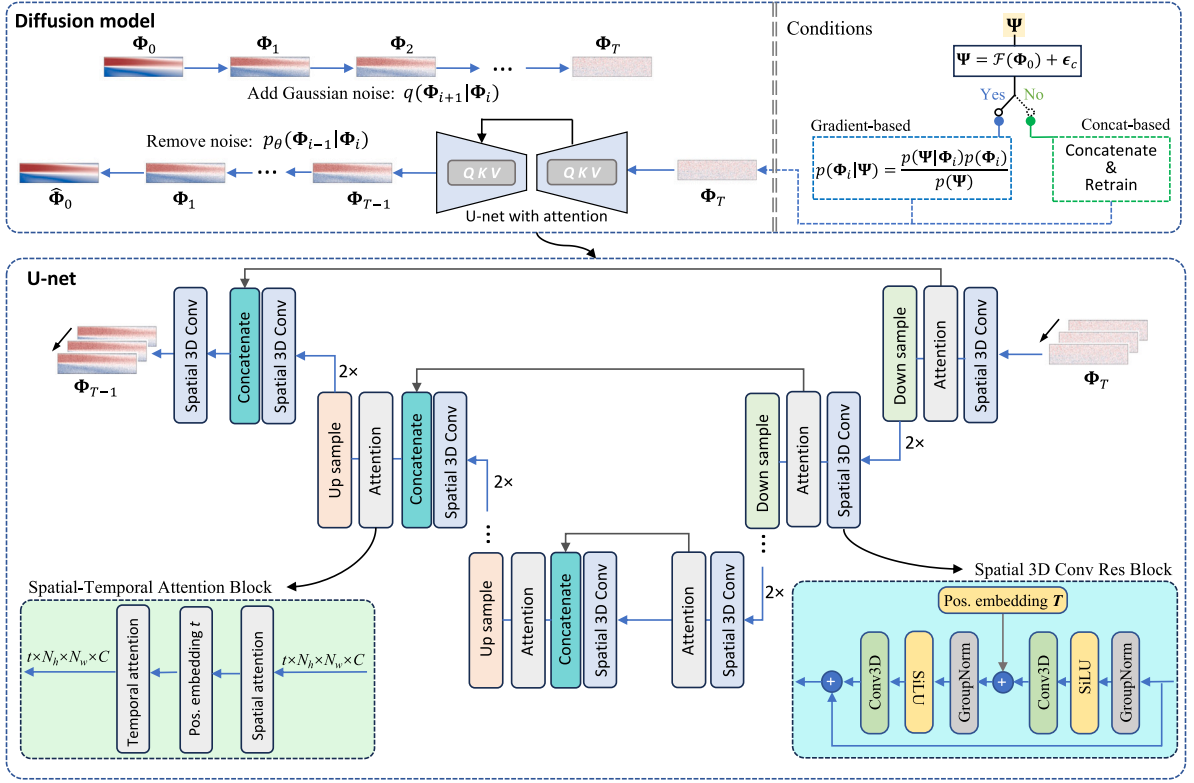
$$p_{\theta}(\Phi_{i-1} | \Phi_i) := q(\Phi_{i-1} | \Phi_i, \hat{\Phi}_{\theta}(\Phi_i; i)), \quad (8)$$

where  $\Phi_0$  is parameterized using deep neural networks with trainable parameters  $\theta$ ,

$$\Phi_0 \approx \hat{\Phi}_{\theta}(\Phi_i; i) = \mathbb{E}_{\Phi_0 \sim p(\Phi_0 | \Phi_i)} [\Phi_0]. \quad (9)$$

In particular, from re-parametrization trick given by Eq. (3),  $\Phi_0$  can be expressed as,

$$\hat{\Phi}_{\theta}(\Phi_i; i) = \frac{\Phi_i - \sigma_i \epsilon_{\theta}(\Phi_i; i)}{\alpha_i} \quad (10)$$



**Fig. 2.** The schematic illustrates the proposed Bayesian conditional diffusion model alongside the detailed architecture of the 3D U-Net used for denoising. Turbulent flow sequences are represented as 4D tensors, with dimensions of time×height×width×channel ( $t \times N_h \times N_w \times C$ ). Spatial convolutions with residual connections (shown in the lower right corner) are applied independently across temporal snapshots, scaling the  $N_h \times N_w$  dimensions by a factor of 2 while maintaining a fixed channel size of 32. Additionally, a multi-head attention mechanism (shown in the lower left corner) is employed at each snapshot to enhance spatial correlations and ensure temporal coherence. It is noteworthy that the position embedding for the diffusion step  $T$  is incorporated into every spatial convolution operation. Furthermore, the position embedding for the time step  $t$  is introduced before the temporal attention processing.

where the noise  $\epsilon_\theta(\Phi_i; i)$  is parameterized by a U-Net variant with residual blocks, attention, and positional embedding [43,47]. The overall schematic of proposed Bayesian condition diffusion model is presented in Fig. 2, where details regarding the neural network structure of  $\epsilon_\theta(\Phi_i; i)$  can be found in the bottom panel. Given that the KL divergence between two Gaussian distributions has a closed form, maximizing the ELBO defined in Eq. (7) can be simplified to following optimization,

$$\theta^* = \arg \min_{\theta} \sum_{\Phi_0 \in \mathcal{A}_{\text{train}}} \mathbb{E}_{\epsilon \sim \mathcal{N}(\mathbf{0}, I), i \sim \mathcal{U}(1, N_{\text{noise}})} [C_i \|\hat{\Phi}_\theta(\Phi_i; i) - \Phi_0\|_{L_2}^2], \quad (11)$$

where  $C_i = \frac{\alpha_{i-1}^2 \sigma_{i-1}^2}{2\sigma_{i-1}^2 \sigma_i^2}$  for  $i \geq 2$  and  $C_1 = 1$ ;  $\mathcal{U}(1, N_{\text{noise}})$  is the discrete uniform distribution from 1 to  $N_{\text{noise}}$ ;  $\Phi_i$  is sampled using re-parametrization based on Eq. (3) [37,45,47]. Namely, it can be computed by  $\Phi_i = \alpha_i \Phi_0 + \sigma_i \epsilon$ , where  $\epsilon$  is standard Gaussian noise, i.e.,  $\epsilon \sim \mathcal{N}(\mathbf{0}, I)$  and  $\Phi_0 \in \mathcal{A}_{\text{train}}$ .

## 2.2. Unconditional generation of spatiotemporal turbulent flow sequences

After training the diffusion model, we can generate new spatiotemporal turbulent flow sequences. This is achieved by sampling the learned distribution via the reverse diffusion process, starting from a multivariate Gaussian distribution  $\Phi_{N_{\text{noise}}} \sim \mathcal{N}(\mathbf{0}, \sigma_{N_{\text{noise}}|0}^2 I)$ . Specifically, we progressively sample  $\Phi_{i-1}$  based on  $\Phi_i$  using the learned reverse transition kernel,

$$p_{\theta^*}(\Phi_{i-1}|\Phi_i) = \mathcal{N}(\tilde{\mu}_{\theta^*}(\Phi_i; i), \tilde{\sigma}_i^2 I) \quad \text{for any } 2 \leq i \leq N_{\text{noise}}, \quad (12)$$

where  $\tilde{\sigma}_i^2 = \frac{\sigma_{i-1}^2 \sigma_i^2}{\sigma_i^2}$  and  $\tilde{\mu}_{\theta^*}(\Phi_i; i) = \frac{\alpha_{i-1} \sigma_{i-1}^2}{\sigma_i^2} \Phi_i + \frac{\alpha_{i-1} \sigma_{i-1}^2}{\sigma_i^2} \hat{\Phi}_{\theta^*}(\Phi_i; i)$ . The Eq. (12) illustrates how we can sample  $\Phi_{i-1}$  based on  $\Phi_i$  from a multivariate Gaussian with a mean of  $\mu_{\theta^*}(\Phi_i; i)$ . This process can be equivalently perceived as  $\Phi_i$  being denoised to move closer to  $\hat{\Phi}_{\theta^*}(\Phi_i; i)$  [45]. Namely, the mean of the learned reverse transition kernel can be expressed in terms of the neural

network-based noise estimate  $\epsilon_{\theta^*}(\Phi_i; i)$  as,

$$\tilde{\mu}_{\theta^*}(\Phi_i; i) = \frac{1}{\alpha_{i|i-1}} \Phi_i - \frac{\sigma_{i|i-1}^2}{\alpha_{i|i-1} \sigma_i} \epsilon_{\theta^*}(\Phi_i; i), \quad (13)$$

To improve the quality of samples, a second-order noise correction is employed [43] as detailed,

$$\epsilon_{\theta^*}^{\text{correct}}(\Phi_i; i) = \frac{1}{2} \left( \epsilon_{\theta^*}(\Phi_i; i) + \epsilon_{\theta^*}(\tilde{\mu}_{\theta^*}(\Phi_i; i)) \right), \quad (14)$$

Subsequently, by substituting the original noise estimate,  $\epsilon_{\theta^*}(\Phi_i; i)$  in Eq. (13) with  $\epsilon_{\theta^*}^{\text{correct}}(\Phi_i; i)$ , the corrected mean is derived as

$$\tilde{\mu}_{\theta^*}^{\text{correct}}(\Phi_i; i) = \frac{1}{\alpha_{i|i-1}} \Phi_i - \frac{\sigma_{i|i-1}^2}{\alpha_{i|i-1} \sigma_i} \epsilon_{\theta^*}^{\text{correct}}(\Phi_i; i). \quad (15)$$

From Eqs. (12) and (15), we introduce a mapping,  $S_{\text{uc}} : (\Phi_i, \epsilon) \mapsto \Phi_{i-1}$

$$\Phi_{i-1} = \tilde{\mu}_{\theta^*}^{\text{correct}}(\Phi_i; i) + \tilde{\sigma}_i \epsilon \quad (16)$$

where  $\epsilon$  represent standard multivariate Gaussian noise. Iteratively applying Eq. (16), we can sample a new  $\Phi_0$  starting from  $\Phi_{N_{\text{noise}}}$ . To set the stage for the discussion on conditional generation in Section 2.3, we can reinterpret Eq. (13) through the lens of the score-based generative modeling framework [39],

$$\tilde{\mu}_{\theta^*}(\Phi_i; i) = \frac{1}{\alpha_{i|i-1}} \Phi_i + \frac{\sigma_{i|i-1}^2}{\alpha_{i|i-1}} s_{\theta^*}(\Phi_i; i), \quad (17)$$

where

$$s_{\theta^*}(\Phi_i; i) := \nabla_{\Phi_i} \log p(\Phi_i) = \frac{\alpha_i \hat{\Phi}_{\theta^*}(\Phi_i; i) - \Phi_i}{\sigma_i^2} \quad (18)$$

is the score model that approximates the fastest increasing direction of the probability density within the state space, as known as stein score [36,39]. The recursive unconditional sampling outlined in this section illustrates the progression from a point of low-probability-density,  $\Phi_{N_{\text{noise}}}$ , towards the region of high probability density,  $\Phi_0$ .

### 2.3. Generation of spatiotemporal turbulent flow sequences with conditions

The capability to generate turbulent flows becomes substantially more significant when tailored to specific parameters, prior knowledge or certain constraints. This conditional generation method enables the synthesis of turbulent flows for specified conditions or scenarios, augmenting its intrinsic value. For example, it will be very useful if we can use the data from efficient low-fidelity (LF) Unsteady Reynolds-Averaged Navier–Stokes (URANS) simulations as the condition to generate corresponding eddy-resolved instantaneous spatiotemporal turbulence solutions, which typically requires high-fidelity simulations, such as Large Eddy Simulations (LES) or Direct Numerical Simulations (DNS) that are computationally demanding. Moreover, when faced with low-resolution turbulence datasets – either due to storage constraints or measurement limitations – conditional sampling will super-resolved generation of high-resolution data, recovering back high-frequency, high-wavenumber details of the flow. Another practical applications of the conditional generation is for data assimilation. Namely, it is advantageous for the generator to yield flow realizations in alignment with available measurement data, which are often sparse or indirect in nature. Building on the foundational methodology introduced in Section 2.2, this subsection delves into the conditional turbulence generation.

Mathematically, any set of conditions can be parameterized as a vector,  $\Psi \in \mathbb{R}^{N_\Psi}$ , referred as to the condition vector in this work. Examples of the condition vector  $\Psi$  include the URANS solution vector, instantaneous flow measurements, flow configuration parameters, low-resolution snapshots, among others. In the context of probabilistic generation modeling, taking into account these additional conditions in the generation process is mathematically equivalent to obtaining the conditional probability  $p(\Phi_0|\Psi)$  of the state  $\Phi_0$  given the condition vector  $\Psi$ . Similar to the unconditional generation discussed above, our objective here is to build a learning model  $p_\theta(\Phi_0|\Psi)$  to approximate the true conditional probability distribution  $p(\Phi_0|\Psi)$ . From Bayes's rule, the learned conditional density can be expressed as,

$$p_\theta(\Phi_0|\Psi) \propto p_\theta(\Psi|\Phi_0)p_\theta(\Phi_0). \quad (19)$$

In this context, we classify conditioning into two categories (a) scenarios in which  $p_\theta(\Psi|\Phi_0)$  is explicitly differentiable; (b) scenarios in which the relationship between generated state  $\Phi$  and condition vector  $\Psi$  is not differentiable and thus  $p_\theta(\Psi|\Phi_0)$  cannot be explicitly obtained. Namely, for category (a), there exists a clear functional relationship between  $\Phi_0$  and  $\Psi$ , whereas for category (b), such a relationship is absent. To address these two situations, we propose two different conditioning methods, (a) gradient-based conditional sampling without retraining and (b) conditional generation necessitating retraining, as described in the subsequent sections.

### 2.3.1. Gradient-based conditional generation without retraining

In the first scenario, the turbulent state variable is directly correlated with the condition vector through the relation  $\Psi = F(\Phi_0) + \epsilon_c$ , where  $F: \mathbb{R}^{N_{\text{length}} \times N_{\text{dof}}} \rightarrow \mathbb{R}^{N_\Psi}$  is a functional mapping from spatiotemporal turbulent field  $\Phi_0$  to its associated condition vector  $\Psi$  (e.g., partial observation derived from  $\Phi_0$ );  $\epsilon_c$  represents zero-mean Gaussian noises with variance  $\sigma_c^2$ . By leveraging the Bayes' theorem, all the latent states conditioned on the vector  $\Psi$  can be expressed as,

$$p(\Phi_i|\Psi) = \frac{p(\Psi|\Phi_i)p(\Phi_i)}{p(\Psi)}, \quad (20)$$

where  $p(\Psi)$  is a normalizing constant that is intractable to compute in general. However, when using the score functions in terms conditional probability, the troublesome denominator is eliminated. Namely, we have,

$$\nabla_{\Phi_i} \log p(\Phi_i|\Psi) = \nabla_{\Phi_i} \log p(\Psi|\Phi_i) + \nabla_{\Phi_i} \log p(\Phi_i), \quad (21)$$

where the second term  $\nabla_{\Phi_i} \log p(\Phi_i)$  can be obtained by the pre-trained score function  $s_{\theta^*}(\Phi_i; i)$  given by Eq. (18). However, as there is not explicit dependency between  $\Psi$  and  $\Phi_i$ , the first term  $\nabla_{\Phi_i} \log p(\Psi|\Phi_i)$  (gradient of log-likelihood term) needs to be approximated based on the relationship between  $\Psi$  and  $\Phi_0$ . To this end, we factorize  $p(\Psi|\Phi_i)$  as follows,

$$\begin{aligned} p(\Psi|\Phi_i) &= \int p(\Psi, \Phi_0|\Phi_i) d\Phi_0 = \int p(\Psi|\Phi_0, \Phi_i) p(\Phi_0|\Phi_i) d\Phi_0 \\ &= \int p(\Psi|\Phi_0) p(\Phi_0|\Phi_i) d\Phi_0 = \mathbb{E}_{\Phi_0 \sim p(\Phi_0|\Phi_i)} [p(\Psi|\Phi_0)], \end{aligned} \quad (22)$$

which can be approximated as [48],

$$\mathbb{E}_{\Phi_0 \sim p(\Phi_0|\Phi_i)} [p(\Psi|\Phi_0)] \approx p(\Psi|\mathbb{E}_{\Phi_0 \sim p(\Phi_0|\Phi_i)}[\Phi_0]). \quad (23)$$

The approximation error is theoretically bounded based on Jensen's inequality, known as the Jensen gap [49]. Now the conditional density  $p(\Psi|\Phi_i)$  can be approximated by  $p(\Psi|\mathbb{E}_{\Phi_0 \sim p(\Phi_0|\Phi_i)}[\Phi_0])$ , where  $\mathbb{E}_{\Phi_0 \sim p(\Phi_0|\Phi_i)}[\Phi_0]$  is the pretrained diffusion model  $\hat{\Phi}_{\theta^*}(\Phi_i; i)$  with optimized parameter  $\theta^*$  (see Eq. (9)). Accordingly, the gradient of the log likelihood in Eq. (21) can be approximated as,

$$\nabla_{\Phi_i} \log p(\Psi|\Phi_i) \approx \nabla_{\Phi_i} \log p(\Psi|\hat{\Phi}_{\theta^*}(\Phi_i; i)), \quad (24)$$

Based on the direct relationship between  $\Psi$  and  $\Phi_0$  as previously outlined, the conditional probability can be represented as

$$p(\Psi|\Phi_0) \approx p(\Psi|\hat{\Phi}_{\theta^*}(\Phi_i; i)) \sim \mathcal{N}(F(\hat{\Phi}_{\theta^*}(\Phi_i; i)), \sigma_c^2 I) \quad (25)$$

By differentiating  $\log p(\Psi|\Phi_i)$  with respect to  $\Phi_i$  using the approximation in Eq. (24), we have

$$\nabla_{\Phi_i} \log p(\Psi|\Phi_i) \approx -\frac{1}{\sigma_c^2} \frac{\partial \|F(\hat{\Phi}_{\theta^*}(\Phi_i; i)) - \Psi\|_{L2}^2}{\partial \Phi_i}, \quad (26)$$

which can be obtained using the automatic differentiation capability of the neural networks. Namely,

$$\frac{\partial F(\hat{\Phi}_{\theta^*}(\Phi_i; i))}{\partial \Phi_i} = \frac{\partial F(\hat{\Phi}_{\theta^*}(\Phi_i; i))}{\partial \hat{\Phi}_{\theta^*}(\Phi_i; i)} \frac{\partial \hat{\Phi}_{\theta^*}(\Phi_i; i)}{\partial \Phi_i}. \quad (27)$$

As such, Eq. (21) can be computed as,

$$\begin{aligned} \nabla_{\Phi_i} \log p(\Phi_i|\Psi) &\approx \nabla_{\Phi_i} \log p_{\theta^*}(\Psi|\Phi_i) + \nabla_{\Phi_i} \log p_{\theta^*}(\Phi_i) \\ &= -\frac{1}{\sigma_c^2} \nabla_{\Phi_i} \left\| F(\hat{\Phi}_{\theta^*}(\Phi_i; i)) - \Psi \right\|_{L2}^2 + s_{\theta^*}(\Phi_i; i), \end{aligned} \quad (28)$$

For conditional generation, the recursive unconditional sampling process as defined by Eq. (17) is guided by the gradient of the log likelihood term  $\nabla_{\Phi_i} \log p_{\theta^*}(\Psi|\Phi_i)$ , which can be modified as,

$$\tilde{\mu}_{\theta^*}^{\text{guide}}(\Phi_i; \Psi, i) = \frac{1}{\alpha_{i|i-1}} \Phi_i + \frac{\sigma_{i|i-1}^2}{\alpha_{i|i-1}} \left( s_{\theta^*}(\Phi_i; i) - \omega \frac{1}{\sigma_c^2} \nabla_{\Phi_i} \left\| F(\hat{\Phi}_{\theta^*}(\Phi_i; i)) - \Psi \right\|_{L2}^2 \right), \quad (29)$$

where  $\omega$  represent a weighting parameter to balance the contributions from both gradient direction as we found that naively adding up both terms may cause low-quality generation with instability. In our implementation, we first compute the unit vector of the conditional sampling direction,

$$n_{\theta^*}^{\text{guide}}(\Phi_i; \Psi, i) := \frac{\nabla_{\Phi_i} \log p_{\theta^*}(\Psi|\Phi_i)}{\|\nabla_{\Phi_i} \log p_{\theta^*}(\Psi|\Phi_i)\|_{L2}}, \quad (30)$$

and then the conditional sampling in terms of score functions is given as,

$$\begin{aligned} \tilde{\mu}_{\theta^*}^{\text{guide}}(\Phi_i; i-1, i) &= \mu_{\theta^*}(\Phi_i; i-1, i) + \\ &\quad \beta_i^{\text{guide}} \max \left( \left\| \frac{\sigma_{i|i-1}^2}{\alpha_{i|i-1}} s_{\theta^*}(\Phi_i; i) \right\|_2, (\sigma_i^{\text{guide}})^2 \right) n_{\theta^*}^{\text{guide}}(\Phi_i; \Psi, i), \end{aligned} \quad (31)$$

**Table 1**

A comparison between the training of unconditional and conditional diffusion models.

	Training optimization	
Unconditional	$\theta^* = \arg \min_{\theta}$	$\sum_{\Phi_0 \in \mathcal{A}_{\text{train}}} \mathbb{E}_{\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), i \sim \mathcal{U}(1, N_{\text{noise}})} [C_i \ \hat{\Phi}_{\theta}(\Phi_i; i) - \Phi_0\ _{L_2}^2]$
Conditional	$\theta^{\circ} = \arg \min_{\theta}$	$\sum_{\Phi_0 \in \mathcal{A}_{\text{train}}} \mathbb{E}_{\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), i \sim \mathcal{U}(1, N_{\text{noise}})} [C_i \ \hat{\Phi}_{\theta}(\Phi_i; i, \Psi) - \Phi_0\ _{L_2}^2]$

**Table 2**

A comparison between the sampling of unconditional and conditional diffusion models.

	Generated state	Mean of reverse kernel	Noise	Score function
Unconditional	$\hat{\Phi}_{\theta^*}(\Phi_i; i)$	$\hat{\mu}_{\theta^*}(\Phi_i; i-1, i)$	$\epsilon_{\theta^*}(\Phi_i; i)$	$s_{\theta^*}(\Phi_i; i)$
Conditional	$\hat{\Phi}_{\theta^*}(\Phi_i; i, \Psi)$	$\hat{\mu}_{\theta^*}(\Phi_i; i-1, i, \Psi)$	$\epsilon_{\theta^*}(\Phi_i; i, \Psi)$	$s_{\theta^*}(\Phi_i; i, \Psi)$

where  $\beta_i^{\text{guide}}$  and  $\sigma_i^{\text{guide}}$  are hyper-parameters, whose magnitudes should be kept relatively small based our empirical observations. To further enhance the results, we can iterate the conditional gradient updating in the last denoising step for  $N_{\text{refine}}$  times.

**Remark 1.** Gradient-based conditional sampling does not need to retrain the diffusion model. Namely, the diffusion model can be trained unconditionally. Once trained, it enables the generation of new turbulent flows under various conditions, as indicated by Eq. (31).

**Remark 2.** Maintaining a balance between the two directional terms in Eq. (28) is important. An excessively large gradient of the conditional probability density may result in samples that meet specific conditions but situated in low-density regions of the unconditional distribution, yielding suboptimal samples. Our objective is to generate samples that not only reside within the dense regions of the unconditional distribution but also align with the set conditions. Considering the challenges associated with direct density evaluation in the diffusion model, it is advisable to accentuate the leading gradient of the unconditional density, while leverage the gradient of the conditional density to guide the sampling with a relatively smaller weight.

### 2.3.2. Concatenation-based conditional generation through retraining

In this subsection, we address the second conditioning scenario, where there is not a clear analytical or differentiable relationship between the condition vectors  $\Psi$  and the turbulent flow solutions  $\Phi_0$ . Take, for instance, the generation of eddy-resolved LES turbulence conditioned on its counterpart URANS simulation results. When training the diffusion model on LES datasets, the relationship between URANS and LES outcomes hinges on shared geometry, Reynolds number, and the fact that both models arise from identical governing equations, albeit with varying modeling assumptions. However, the direct mapping function  $F: \Psi \rightarrow \Phi_0$  and its associated gradients, which would establish a tangible link between RANS and LES results, are elusive, given the high-dimensional intricacies (space-time degrees of freedom), solver inconsistencies (as they may originate from distinct solvers), and the considerable computational demands. Such complexities make the gradient-based conditional generation technique inapplicable in this context.

As a viable alternative strategy, conditions can be incorporated into the neural networks' input, necessitating a retraining process with conditioned samples to implicitly approximate the true conditional distribution. More specifically, the U-Net's input is modified by concatenating the condition vector  $\Psi$  with the perturbed hidden state  $\Phi_i$ . The training procedure remains the same as that for the unconditional diffusion model as discussed in Section 2.1, where the Gaussian transition kernel is unaltered. In this way, the conditional mapping  $F: \Psi \rightarrow \Phi_0$  is implicitly learned by the neural networks, and thereby directing the flow generation conditioned on  $\Psi$ . Tables 1 and 2 provide a comparison between the training and sampling processes of the unconditional and conditional diffusion models, respectively.

## 2.4. Generating long-span spatiotemporal turbulence

The proposed diffusion model is adept at generating spatiotemporal turbulence fields  $\Phi_0 \in \mathbb{R}^{N_{\text{length}} \times N_{\text{dof}}}$ , consisting of a fixed set of  $N_{\text{length}}$  temporal snapshots. Due to the intrinsic limitations imposed by memory footprints,  $N_{\text{length}}$  cannot be substantially large during the training phase. Nonetheless, for enhanced applicability in real-world scenarios, there is an imperative need to extrapolate the capabilities of the trained diffusion model to enable the synthesis of turbulent sequences that span duration considerably exceeding the default length  $N_{\text{length}}$ . To this end, we propose an autoregressive conditioning sampling strategy, which allows us to robustly generate long-span turbulent flows with arbitrary temporal length. Specifically, the spatiotemporal turbulent state  $\Phi_0$  of default length is decomposed into two non-overlapping subsequences:

$$\Phi_0 = [\Phi_0^a, \Phi_0^b], \quad (32)$$

where  $\Phi_0^a \in \mathbb{R}^{N_{\text{previous}} \times N_{\text{dof}}}$  represents the preceding subsequence, whereas  $\Phi_0^b \in \mathbb{R}^{(N_{\text{length}} - N_{\text{previous}}) \times N_{\text{dof}}}$  denotes the subsequent flow subsequence. Now the autoregressive conditional generation is formulated as follows: given an preceding flow sequence  $\Phi_0^a$ , the



goal is to generate a subsequent flow sequence  $\Phi_0^b$  based on the conditional probability density  $p(\Phi_0^b|\Phi_0^a)$ . Once the diffusion model is trained with optimal parameters  $\theta^*$ , the subsequent flow sequence can be obtained via conditional sampling,

$$\Phi_0^b \sim p_{\theta^*}(\Phi_0^b|\Phi_0^a), \quad (33)$$

By repetitively sampling from these conditional probability distributions in an autoregressive manner, our approach holds the capability to synthesize turbulent sequences with an arbitrary length. Significantly, this can be achieved without the need of retraining the model.

#### 2.4.1. Gradient-based autoregressive conditioning method

The gradient-based conditional generation method, introduced in Section 2.3.1, can be used to achieve this goal. In this context, the conditional vector is the preceding subsequence,  $\Psi := \Phi_0^a$ . Within the gradient-based conditional generation framework, the gradient of log likelihood associated with the preceding subsequence is used to guide the generation of the subsequent sequence  $\Phi_0^b$ . Specifically, we autoregressively sample the conditional distribution  $p(\Phi_0|\Phi_0^a)$  of the turbulence sequence  $\Phi_0$  with the default length  $N_{\text{length}}$  given the condition vector, i.e.,  $\Phi_0^a$ . The state-to-condition mapping  $F$  is defined as,

$$F: \mathbb{R}^{N_{\text{length}} \times N_{\text{dof}}} \rightarrow \mathbb{R}^{\frac{1}{2} * N_{\text{length}} \times N_{\text{dof}}}, \quad (34)$$

which selected out the first  $\frac{1}{2} * N_{\text{length}}$  snapshots of the generated flow sequence  $\hat{\Phi}_{\theta^*}$ , i.e.,

$$F(\hat{\Phi}_{\theta^*}) = \hat{\Phi}_{\theta^*}^a, \quad (35)$$

where  $\hat{\Phi}_{\theta^*}^a$  is the first half portion of the full sequence  $\hat{\Phi}_{\theta^*}$ . The key for conditional sampling is to approximate the gradient of log likelihood of condition vector  $\nabla_{\Phi_i} \log p(\Psi|\Phi_i)$ , which involve the derivative computation as follows,

$$\frac{\partial \|\hat{\Phi}_{\theta^*}(\Phi_i; i) - \Psi\|_{L2}^2}{\partial \Phi_i} = \frac{\partial \|\hat{\Phi}_{\theta^*}(\Phi_i; i) - \Phi_0\|_2^2}{\partial \Phi_i}. \quad (36)$$

Then, the subsequent operations are detailed in Section 2.3.1. By adopting this autoregressive approach, the model can successively generate segments of the flow sequence. Each subsequent segment is influenced by the preceding segment, ensuring smooth and coherent transitions in the turbulent dynamics. This iterative procedure, when executed repeatedly, facilitates the generation of flow sequences of any desired length. This methodology offers a potent and versatile solution for extending the capabilities of our trained diffusion model beyond its inherent limitations, fulfilling the demand for long-duration turbulent sequences in various practical applications.

#### 2.4.2. Replacement-based autoregressive method

An alternative autoregressive method has been discussed by Ho et al. [43], targeting the generation of extended videos. While the foundational approach remains rooted in unconditional sampling, the strategy involves modifying partial hidden states to enable autoregressive generation. Specifically, the (hidden) state  $\Phi_i$  is partitioned as  $\Phi_i = [\Phi_i^a, \Phi_i^b]$ . We aim to compute the expectation of  $\Phi_0^b$  over the conditioned probability  $p_{\theta^*}(\Phi_0^b|\Phi_0^a, \Phi_i) = p_{\theta^*}(\Phi_0^b|\Phi_0^a, [\Phi_i^a, \Phi_i^b])$  as,

$$\begin{aligned} \mathbb{E}_{\Phi_0^b \sim p_{\theta^*}(\Phi_0^b|\Phi_i)}[\Phi_0^b] &= \int p_{\theta^*}(\Phi_0^b|\Phi_0^a, [\Phi_i^a, \Phi_i^b]) \Phi_0^b d\Phi_0^b \\ &= \int \frac{p_{\theta^*}(\Phi_0^b|\Phi_0^a, [\Phi_i^a, \Phi_i^b]) p_{\theta^*}(\Phi_0^b|[\Phi_0^a, \Phi_i^b])}{p_{\theta^*}(\Phi_0^b|[\Phi_0^a, \Phi_i^b])} \Phi_0^b d\Phi_0^b \\ &= \mathbb{E}_{\Phi_0^b \sim p_{\theta^*}(\Phi_0^b|[\Phi_0^a, \Phi_i^b])} \left[ \frac{\Phi_0^b p_{\theta^*}(\Phi_0^b|\Phi_0^a, [\Phi_i^a, \Phi_i^b])}{p_{\theta^*}(\Phi_0^b|[\Phi_0^a, \Phi_i^b])} \right]. \end{aligned} \quad (37)$$

By assuming  $p_{\theta^*}(\Phi_0^b|\Phi_0^a, [\Phi_i^a, \Phi_i^b]) \approx p_{\theta^*}(\Phi_0^b|[\Phi_0^a, \Phi_i^b])$  as iterative denoised  $\Phi_i^a$  converges to  $\Phi_0^a$ , we have

$$\begin{aligned} \mathbb{E}_{\Phi_0^b \sim p_{\theta^*}(\Phi_0^b|[\Phi_0^a, \Phi_i^b])} \left[ \frac{\Phi_0^b p_{\theta^*}(\Phi_0^b|\Phi_0^a, [\Phi_i^a, \Phi_i^b])}{p_{\theta^*}(\Phi_0^b|[\Phi_0^a, \Phi_i^b])} \right] &\approx \mathbb{E}_{\Phi_0^b \sim p_{\theta^*}(\Phi_0^b|[\Phi_0^a, \Phi_i^b])} [\Phi_0^b] \\ &= \hat{\Phi}_{\theta^*}^b([\Phi_0^a, \Phi_i^b]; i) \end{aligned} \quad (38)$$

where  $\hat{\Phi}_{\theta^*}^b([\Phi_0^a, \Phi_i^b]; i)$  is from the partition of,

$$\hat{\Phi}_{\theta^*}([\Phi_0^a, \Phi_i^b]; i) = [\hat{\Phi}_{\theta^*}^a([\Phi_0^a, \Phi_i^b]; i); \hat{\Phi}_{\theta^*}^b([\Phi_0^a, \Phi_i^b]; i)]. \quad (39)$$

Namely, the core of the replacement method is to replace the first half of the hidden vector  $\Phi_i^a$  with the given  $\Phi_0^a$  during unconditional sampling process.

**Remark 3.** The formulation in (38) operates under the assumption  $p_{\theta^*}(\Phi_0^b|\Phi_0^a, [\Phi_i^a, \Phi_i^b]) \approx p_{\theta^*}(\Phi_0^b|[\Phi_0^a, \Phi_i^b])$ . The validity of this assumption in the context of estimating the density value ratio (which corresponds to the importance sampling weight in (37)) remains unclear, even with an adequately trained model, i.e.,  $i \rightarrow 0$ . Since  $p_{\theta^*}(\Phi_0^b|\Phi_0^a, [\Phi_i^a, \Phi_i^b])$  is always greater than  $p_{\theta^*}(\Phi_0^b|[\Phi_0^a, \Phi_i^b])$  due to more information in the conditioning, the density ratio is always greater than one. The replacement method's disregard for the importance weights exceeding unity might lead to underestimated values. As sequences are progressively generated, this diminishing effect could compound, aligning with our observations in the experimental study presented in Section 3.1.

### 3. Numerical results and discussions

To showcase the robust and versatile turbulence generation capabilities of the proposed conditional diffusion model, we conducted on a series of numerical experiments, exploring various generation scenarios on three distinct turbulent flow cases:

1. 2D unsteady flows over a backward-facing step: In this case, we highlight the model's capability in generating LES-like instantaneous eddy-resolved turbulent flows when provided with URANS simulated flow solutions.
2. 3D turbulent channel flows: Here, the diffusion model is trained to adeptly generate instantaneous spatiotemporal sequences of turbulent channel flow, given specified initial conditions, statistics, or entirely from scratch.
3. 3D compressible supersonic turbulent boundary layers: This case underscores the model's super-resolution generation capabilities—where the high-resolution DNS high-speed turbulent boundary layers are generated conditioned on low-resolution input measurements.

#### 3.1. RANS-conditioned generation of eddy-resolved turbulence over backward-facing step

In this section, we showcase the capabilities of our proposed method in generating LES-like spatiotemporal flow realizations, predicted upon a URANS-simulated flow. We further compare the performance of different variants of our proposed method. Consider a 2D channel featuring a backward-facing step; the specific configuration of this flow case is depicted in Fig. A.15. Our focus revolves around the unsteady velocity, as a spatiotemporal vector field  $\mathbf{u}(x, y, t) = [u_1(x, y, t), u_2(x, y, t)]^T : \Omega \times \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}^2$ , where  $\Omega$  represents the computational domain. A key area of our interest is the flow separation zone after the step, indicated as the shaded area in Fig. A.15. In LES simulations, introducing random perturbations to the inlet yields multiple realizations of unsteady flow sequences even at the same Reynolds number. In contrast, the URANS simulation inherently averages out these stochastic fluctuations. Consequently, for a specified Reynolds number, every distinct LES realization of instantaneous flow sequence can be associated with a single URANS simulated flow sequence. Essentially, these LES flow realizations can be interpreted as samples drawn independently from a stochastic distribution conditioned on the URANS result for a given Reynolds number. The LES and URANS simulation details are provided in Table A.3.

##### 3.1.1. Data preparation and model training

The diffusion model is developed to generate eddy-resolved instantaneous flow sequences given different Reynolds number. To produce the flow dataset, we consider a parameter set  $D$  consisting of 11 Reynolds numbers ( $Re$ ) evenly distributed between 5000 to 14000,

$$D = \{Re_i = 5000 + (i - 1) \times 900 | i = 1, \dots, 11\}. \quad (40)$$

At each Reynolds number ( $Re_i$ ), we perform one URANS simulation and the simulated unsteady flow sequence ( $\Psi$ ) contains 240 snapshots

$$\Psi^{Re_i} = [\psi_1^{Re_i}, \dots, \psi_{240}^{Re_i}], \quad i = 1, \dots, 11, \quad (41)$$

and LES simulations with five realizations of random inlet perturbations ( $\Phi$ )

$$\Phi_k^{Re_i} = [\phi_{k,1}^{Re_i}, \dots, \phi_{k,240}^{Re_i}], \quad i = 1, \dots, 11, k = 1, \dots, 5. \quad (42)$$

Subsequently, we partition the parameter set into two subsets: the testing set ( $D_{\text{test}}$ ) and the training set ( $D_{\text{train}}$ ),

$$D_{\text{test}} = \{5900, 13100\}, \quad D_{\text{train}} = D \setminus D_{\text{test}}. \quad (43)$$

Considering memory constraints and the need for data augmentation, the complete flow sequence of 240 snapshots is divided into 200 shorter subsequences, each containing 40 snapshots, for training purposes. In testing phase, long flow sequence can be generated using auto-regressive conditional sampling as discussed above. Specifically, the training sets for condition vectors and corresponding flow solutions are given as,

$$\begin{aligned} \mathcal{V} &= \{ \{ \psi_j^{Re_i}, \dots, \psi_{j+40}^{Re_i} \} \mid Re_i \in D_{\text{train}}, j = 1, \dots, 200 \}, \\ \mathcal{W} &= \{ \{ \phi_{k,j}^{Re_i}, \dots, \phi_{k,j+40}^{Re_i} \} \mid Re_i \in D_{\text{train}}, j = 1, \dots, 200, k = 1, \dots, 5 \}. \end{aligned} \quad (44)$$

**Remark 4.** We consistently adopt the data augmentation strategy outlined in (44) to address the challenges posed by limited training data. This approach offers an advantage in generating a greater number of sequences compared to simply dividing the lengthy sequence into discrete non-overlapping segments. Although having overlapping sequences might seem redundant, it is essential for ensuring comprehensive learning by the model. This method is a commonly used data augmentation technique for learning sequential data, enabling the model to capture transitional dynamics more accurately, thus enhancing predictive accuracy. It is important to note that this overlapping-based data augmentation serves to enrich the training dataset and improve training efficiency. Once trained, the model can efficiently generate non-overlapping sequences of up to 40 steps in length.

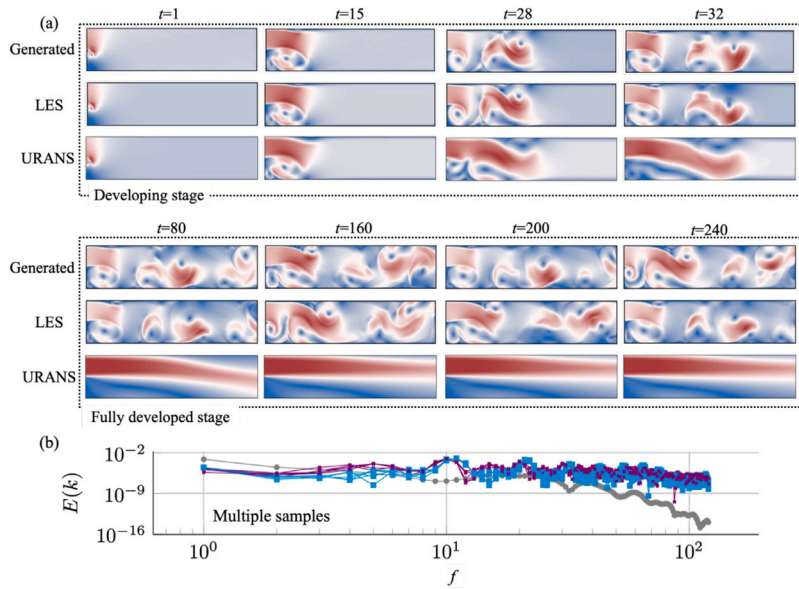


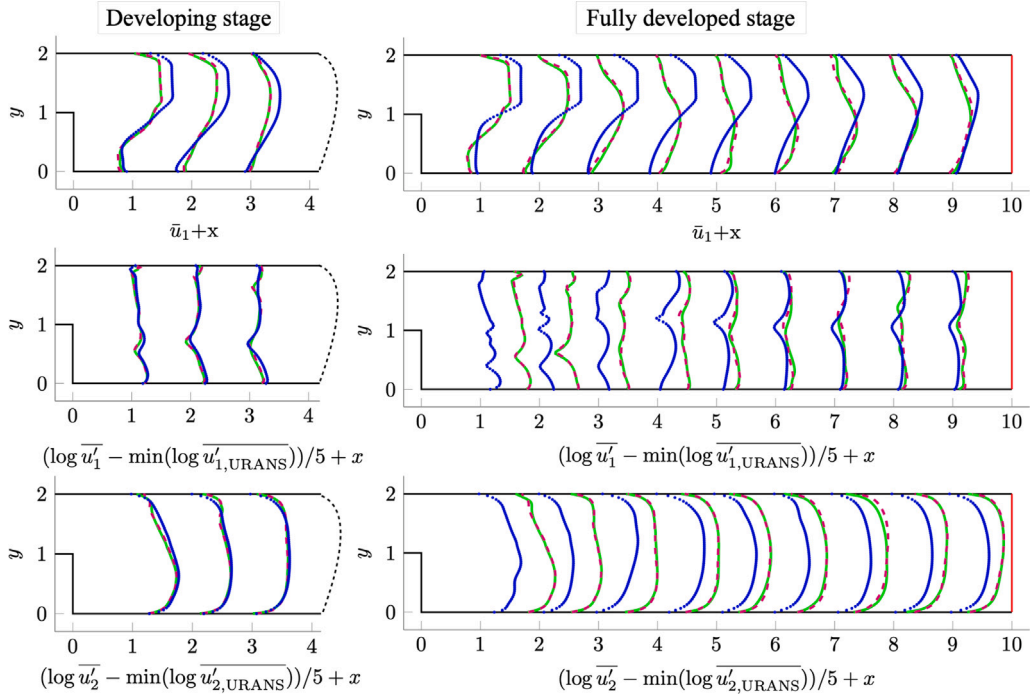
Fig. 3. (a) Velocity magnitude of one generated samples, LES and URANS at  $Re = 5900$  at developing and fully developed stages. (b) the energy spectra for multiple generated samples (—), compared with LES (—) and URANS (—) at  $x = 4, y = 1$ .

### 3.1.2. URANS-conditioned generation of LES flow sequences on testing $Re$

Given a testing Reynolds number (e.g.,  $Re = 5900$ ), URANS is conducted to serve as the conditioning for the LES-like generation. As shown in Fig. 3, a long LES-like turbulent flow over a backward-facing step is generated by the gradient-based autoregressive method. We first focus on the flow development phase (first 40 steps), specifically observing the formation and growth of the recirculation bubble as the flow progresses past the step. For one realization of the generated flow sequence (see contours in Fig. 3), it is noticeable that the generated sequence retains similarities with the URANS simulated one up to approximately the 15th timestep, with significant divergences emerging after the 20th timestep. Post the 40th timestep, although the URANS-simulated conditions tend to stabilize, the generated samples retain their turbulent nature, marked by the dynamically changed vortex structures, unaffected by the nearly-steady URANS conditioning. This observation suggests that our model actually generates new flow features, advancing beyond mere replication or upscaling of URANS data, which is fundamentally different from previous SR works [34]. By comparing the contours of one realization of generated flow sequences with one LES sequence, we found the unsteady flow patterns, vortex structures and level of details are very similar. It is important to note that the instantaneous flow fields between the generated and actual LES samples inherently differ, as they are independent random realizations and lack any direct comparison.

To further compare our generative model with LES statistically, we generate multiple realizations based on the same URANS simulation. The energy spectra of the realizations from both LES and diffusion models behind the step ( $x = 4, y = 1$ ) are plotted in Fig. 3, where the URANS result is also shown for comparison. As observed in Fig. 3, the URANS simulation's energy spectrum decays rapidly at high frequencies and notably lacks the pronounced peak at  $f = 10$ . This absence of a peak in the URANS spectrum is indicative of the model's limited capacity to resolve the eddies and capture the recirculation region in the wake after the step. In contrast, all realizations from our diffusion model effectively generate these eddies and the recirculation regions and their alignment with the LES results is evident, as their energy spectra show a consistent peak value and statistical characteristics. Additionally, the energy spectra for different realizations exhibit slight differences, indicating the generation process in our model is non-deterministic. Similar conclusions are drawn for a higher testing Reynolds number ( $Re = 13100$ ), as evidenced in Fig. A.17. This demonstrates the capability of our Bayesian-based diffusion model to produce diverse LES-like instantaneous eddy-resolved turbulent flows when provided with URANS-simulated flow solutions for a specified  $Re$ . It is also worth noting the model's strong performance in generating accurate results for Reynolds numbers beyond those it was trained on, underscoring its ability to generalize to interpolated Reynolds numbers.

We also examine the mean velocities and their fluctuations at various locations in the wake region to spatially assess the statistics of our generated samples. Fig. 4 shows a comparison between the diffusion-generated results and the LES/URANS simulations for  $Re = 5900$  across nine representative locations, including recirculation, reattachment, and recovery areas. The profiles of the first and second moments of the generated samples agree with those of LES simulated results very well in both developing and fully developed stages. While URANS simulations reasonably capture the mean velocities and their fluctuations during the developing stage, they significantly underperform for the fully developed flows. Notably, URANS simulations substantially underestimate velocity fluctuations when the flow become fully developed, primarily due to their inability to accurately capture the recirculation region in the wake. This leads to notable discrepancies in mean velocities, especially in the region defined by  $x = [0, 6]$  and  $y < 1$ . Beyond the recirculation region, particularly in the area of  $x = [7, 9]$  and  $y < 1$ , URANS simulations align more closely with LES



**Fig. 4.** Mean of streamwise velocity (first row), variance of streamwise velocity (second row) and variance of wall-normal velocity (third row) at  $Re = 5900$  for developing stage (left column) and fully developed stage (right column) from LES (●), URANS (●) and diffusion model (●) evaluated at different streamwise locations  $x$ . The  $x$ -axis labels on the subplots in the second and third rows are designed to represent the normalized velocity fluctuations along the spatial coordinates.

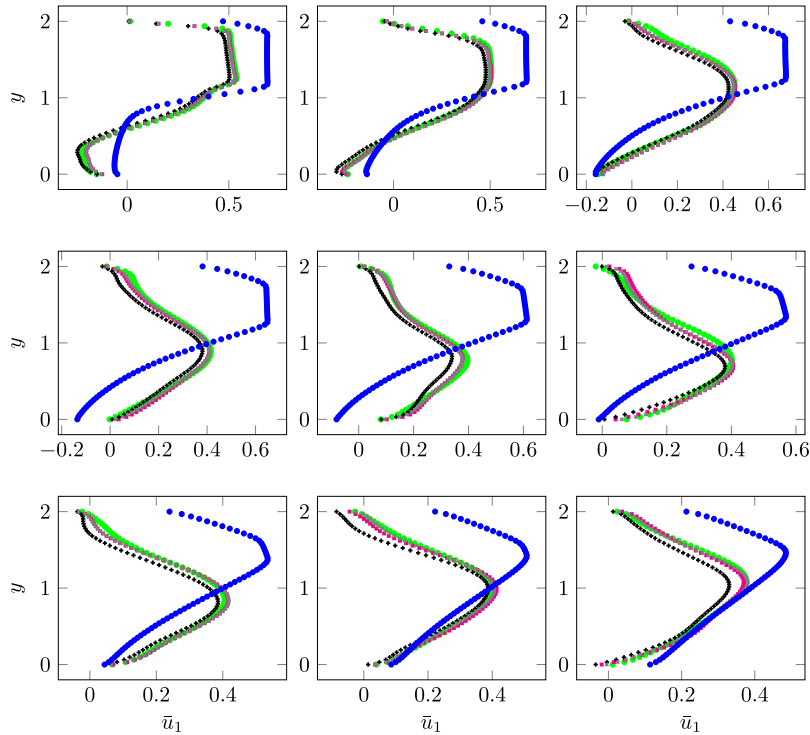
results. However, URANS tends to overestimate mean velocities in regions where  $y > 1$ . Conversely, our diffusion model results can generate intricate vortices and flow patterns which align well with LES results, as evident by their agreements in velocity mean and fluctuation profiles across all of these wake regions. The same observation and conclusion can be obtained for the case of  $Re = 13100$ , as shown in Fig. A.18.

In addition to evaluating turbulence statistics, we have utilized the Continuous Ranked Probability Score (CRPS) to quantify the agreement between our model-generated samples and simulation samples. We calculated the CRPS for our model's generated turbulent flow samples using a reference set of LES/DNS samples. Moreover, we also computed the CRPS between two distinct sets of LES/DNS simulated samples to establish a baseline for comparison. The CRPS values obtained by the Diff-gradient method for  $Re = 13100$  are detailed in Table D.6. This analysis shows that the CRPS values of samples generated by our model are comparable to those derived from more computationally-intensive LES/DNS simulations.

### 3.1.3. Comparison of different autoregressive sampling methods

We previously introduced gradient-based (Section 2.3.1, referred to as Diff-Gradient) and replacement-based (Section 2.3.2, referred to as Diff-Replace) autoregressive sampling method in details. This section aims to evaluate these two auto-regressive conditional sampling schemes for generating extended spatiotemporal flow sequences, where each newly generated flow sub-sequence is conditioned on the previously generated ones autoregressively. Serving as our baseline, “Diff-Vanilla” refers to the direct application of VideoDiffusion [43] to produce a series of sub-sequences without autoregressive conditioning, thereby lacking temporal coherence between sub-sequences. We conducted a comparative analysis of sample statistics at nine locations ( $x = [1, 9]$ ) for  $Re = 13100$  from the testing set. As shown in Fig. 5, all three methods are capable of generating samples conditioned on URANS, effectively capturing the mean velocity within the recirculation region and far wakes. However, the replacement-based method slightly underestimates the mean velocity, particularly in the far wake region. Regarding long-span generation, the direct generation method is impractical due to memory constraints. Although both autoregressive generation methods can produce arbitrarily long sequences, the replacement-based method tends to exhibit a continuous decrease in velocity magnitude over time. This trend is attributed to noticeable energy dissipation during self-conditioning rollouts, as discussed in Section 2.4.2. Our proposed gradient-based autoregressive generation method demonstrates robust capability in generating LES-like turbulence over extended durations.

We also evaluated the computational cost of three sampling methods using the diffusion model, as shown in Fig. A.16. Since the URANS computation is performed only once and serves as a condition, its cost is not included in the evaluation for the generation of multiple realizations. In contexts requiring numerous short and coherent sequences, such as those focusing exclusively on developed flows, the vanilla method (Section 2.2) is the most efficient, offering significant speed improvements. Conversely, for generating



**Fig. 5.** Mean of streamwise velocity at  $Re = 13100$  in the fully-developed phase: LES (●), URANS (●), Diff-Gradient (●), Diff-Replace (●) and Diff-Vanilla (●). The figures, arranged from left to right and top to bottom, correspond to locations at  $x = [1, 9]$ .

long and coherent sequences, the gradient-based autoregressive method is preferable due to its superior performance and faster operation compared to the replacement-based autoregressive method. It is important to note that this initial example primarily demonstrates the model's ability to parametrically convert dynamics characterized by large-scale features into those exhibiting small-scale and chaotic features, while the speedup observed in this case due to the computational efficiency of the small-scale 2D LES simulation. Although the training cost for this case on the same hardware is about 24 h, it is anticipated that the benefits in terms of computational efficiency for the diffusion model will be more significant in scenarios involving large-scale and more intricate 3D turbulence simulations, which will be further studied in the subsequent sections.

### 3.2. Generation of instantaneous turbulent channel flows under various conditions

We now demonstrate the efficacy of our proposed method in generating instantaneous DNS velocity fields of a 3D turbulent channel flow from scratch or from specified conditions such as initial flow snapshot, RMS profiles of velocity fluctuations, and Reynolds stress. For the fully developed turbulent channel flow at  $Re_\tau = 180$ , governed by the unsteady incompressible Navier–Stokes equations [50], the flow exhibits homogeneity in the streamwise and spanwise directions. In this scenario, the turbulence are statistically homogeneous except in the wall-normal direction ( $y^+$ ) [51]. This one-dimensional characteristic facilitates the synthesis of independent two-dimensional samples from uncorrelated sections perpendicular to the streamwise direction [19,50–52]. This one-dimensional characteristics allows us to synthesize independent two-dimensional instantaneous flow sequences from uncorrelated sections perpendicular to the streamwise directions [19,50–52]. Our focus is on generating the 3D velocity fields ( $\mathbf{u}(y, z, t) = [u_1(y, z, t), u_2(y, z, t), u_3(y, z, t)]^T : \Omega \times \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}^3$ ) at the channel cross-section. We will explore in detail the application of the gradient-based autoregressive conditional diffusion method for generating long-span, spatiotemporal dynamics of the 3D channel flows.

#### 3.2.1. Data preparation and model training

The diffusion model is built to produce the cross-sectional velocity field of a 3D channel of dimensions  $L_x \times L_y \times L_z = 2\pi \times 2 \times \pi$  at  $Re_\tau = 180$ , with an emphasis on generating extended sequences under various conditions. The dataset is obtained through fully-resolved DNS runs, where the cross section is discretized by  $N_y \times N_z = 256 \times 128$ . The instantaneous velocity fields of a total of 14 uncorrelated cross-sections along the streamwise direction, spanning 4 flow-through times, are collected to create our dataset. Each flow-through time includes 300 time steps, resulting in a total of 16,800 snapshots. Of these, 86% are utilized for training, and the remaining 14% serve as test set for turbulence statistics comparison.



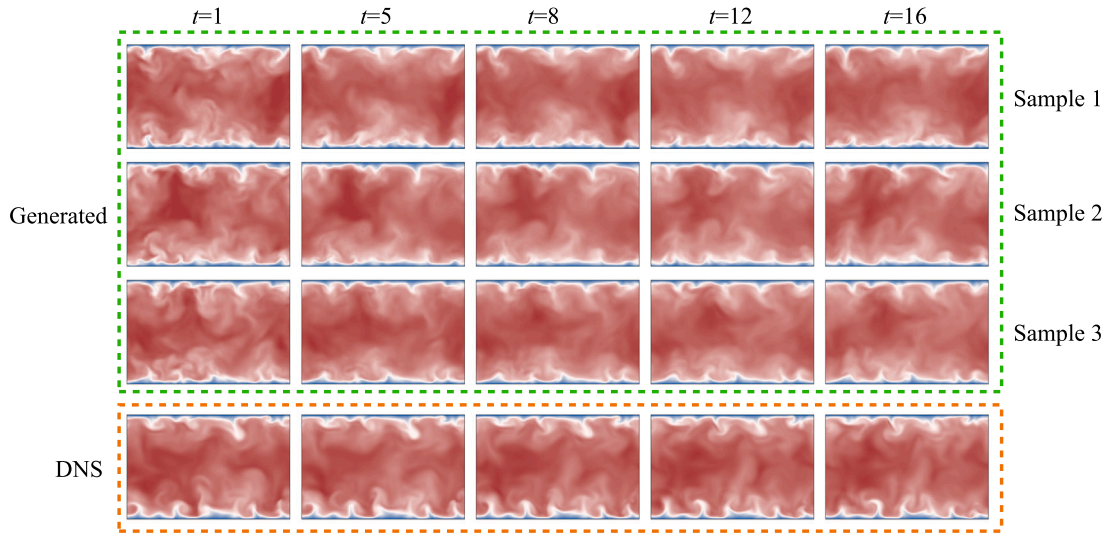


Fig. 6. Spatiotemporal sequences of instantaneous streamwise velocity fields generated from our diffusion model and DNS.

### 3.2.2. Direct generation of long sequential turbulence from scratch

We initially generated the flow field entirely from scratch using the trained diffusion model, without imposing any specific conditions. For this purpose, we employed the gradient-based autoregressive method to produce long-span turbulence sequences spanning 300 time steps (equivalent to one flow-through time, as detailed in Table B.4). This approach was chosen as the velocity tends to dampen in the replacement-based method during self-rollout, as previously mentioned. Figs. 6, B.24, and B.25 display the generated instantaneous velocity fields alongside those simulated via DNS for comparison. Three randomly generated realizations of the spatiotemporal flow sequences are plotted, which all visually resample the DNS reference, showcasing our model's ability to produce diverse and realistic instantaneous velocity fields. This is fundamentally distinct from the approach using sequence neural networks (e.g., ConvLSTM or Transformer), which yields a single deterministic trajectory with a given initial condition. The CRPS values, as shown in Table D.6, closely match those of DNS samples, indicating the similarity of the generated samples to DNS data. To further evaluate the generative performance, we compared the statistics of eight generated flow samples, each with 300 steps, against DNS data as shown in Fig. 7. Remarkably, the mean streamwise velocity profile of the generated data mirrors that of the DNS across various flow regions, including the linear viscous sublayer, buffer layer, and logarithmic region. Furthermore, the root-mean-square (RMS) profiles of velocity fluctuations generated by our model demonstrate a high agreement with those obtained from DNS. This level of accuracy in capturing the essential turbulent characteristics underscores the robustness and versatility of our diffusion-based approach in simulating complex turbulent flows. More statistics, including the spanwise energy spectra and two-point correlations of the velocity components at different wall distances, are provided in Appendix (see Figs. B.21 and B.22). Overall, they are all reasonably in good agreement with those obtained from the DNS, especially in logarithmic region (e.g.  $y^+ = 180$ ). Although slight difference in low-wave-number regions can be observed when probing the viscous sublayer and buffer layer, the anisotropic features along the wall direction are well captured. Moreover, the two-point correlations of all the generated flow samples agree with the DNS reference reasonable well, falling off to almost zero within a half width of the computational domain for both the streamwise and spanwise directions [53]. There are small but noticeable bumps in the streamwise and spanwise two-point correlation of our generated results in the near-wall regions, which is possibly due to large-scale isotropic structures generated around the channel center, which can be removed by introducing additional information using conditional sampling.

### 3.2.3. Conditioned generation given prior sequence

In this section, we demonstrate the model's proficiency in generating conditioned sequences, a process where the model generates a sequence of flow based on a given set of prior flow sequence of varying lengths. To demonstrate this, we utilize an unseen sequence (spanning from  $t = 1 \rightarrow t = 20$ ) from the DNS database. The length of the prior sequence that the diffusion model is conditioned on can be varied from 1 to 19. Namely, the diffusion model generates the remaining portion of the sequence up to  $t = 20$  by sampling the trained model. Notably, the generation step is amplified a hundred times relative to the numerical step, resulting in a total sequence length of 2000 numerical steps. Fig. 8 presents the final snapshots of these generated sequences, compared against the last snapshot from the DNS simulations. The results show an expected trend: fewer prior snapshots lead to greater randomness in the final generated snapshot, while a higher number of prior snapshots results in a closer resemblance to the DNS counterpart. This trend of decreasing error, as illustrated in Fig. B.23, effectively demonstrates the model's adeptness in incorporating conditional information and updating the posterior distribution accordingly. Moreover, it is crucial to highlight that these conditioned generations do not necessitate any retraining of the model. The diffusion model, once trained in an unconditional manner, can be directly employed for sampling with conditioning.

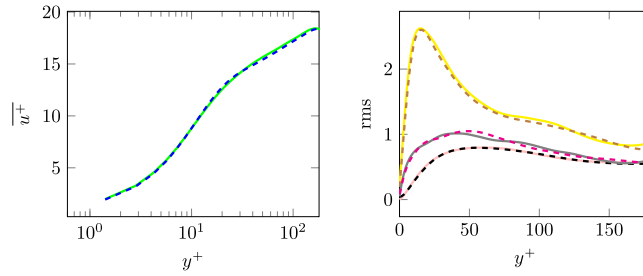


Fig. 7. The mean streamwise velocity profile (left) from DNS (—) and diffusion model (---); The RMS profiles (right) of streamwise velocity fluctuation from DNS (—), diffusion model (---), of wall-normal velocity fluctuation from DNS (—), diffusion model (---), and of spanwise velocity fluctuation from DNS (—), diffusion model (---).

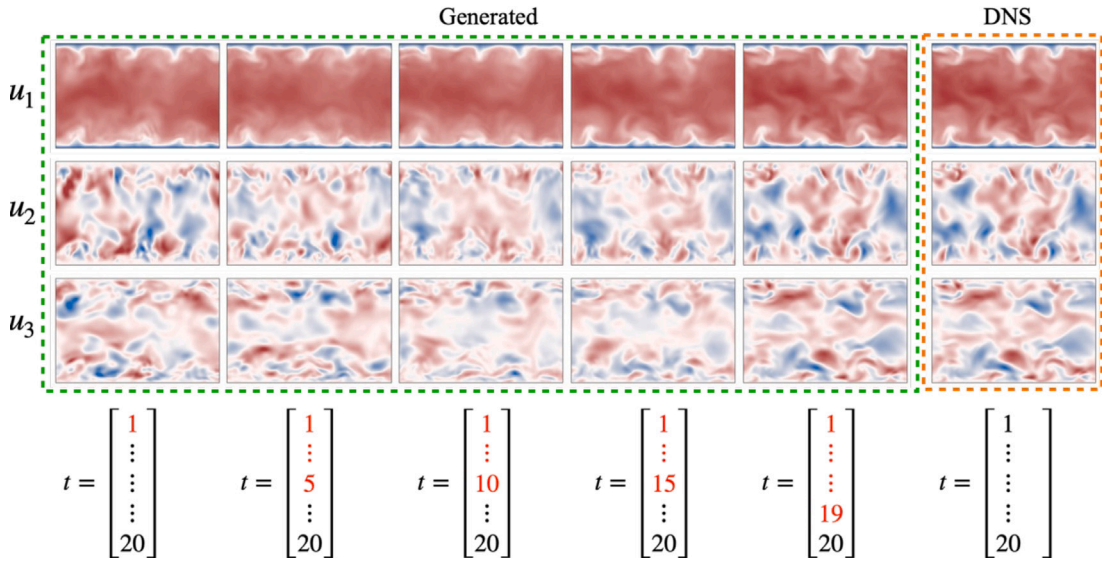


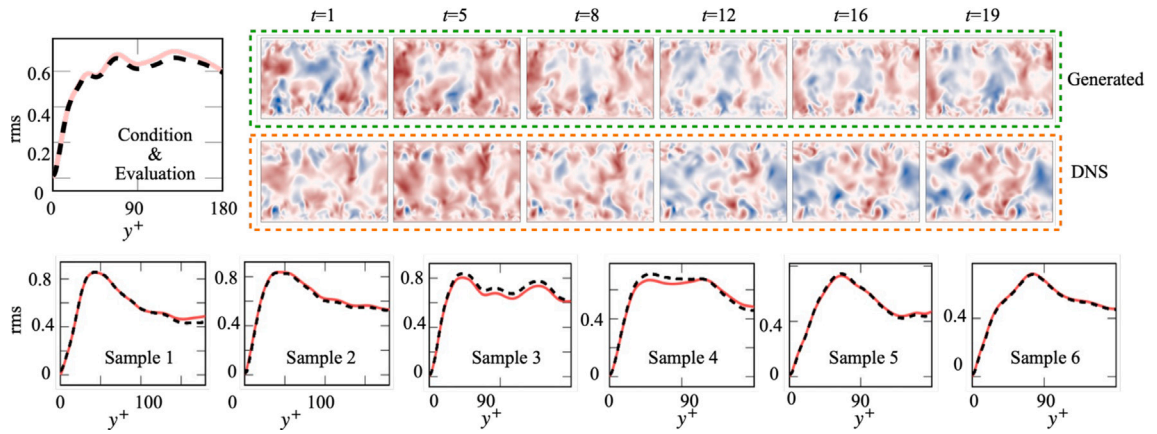
Fig. 8. The contours of streamwise ( $u_1$ ), wall-normal ( $u_2$ ), spanwise ( $u_3$ ) velocity at  $t = 20$  generated by diffusion model and DNS. In the context of  $t$ , the presence of red signifies conditioning, while black represents generation. Each row use a same colorbar to highlight features in the corresponding velocity component. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

### 3.2.4. Conditioned generation with desired statistics

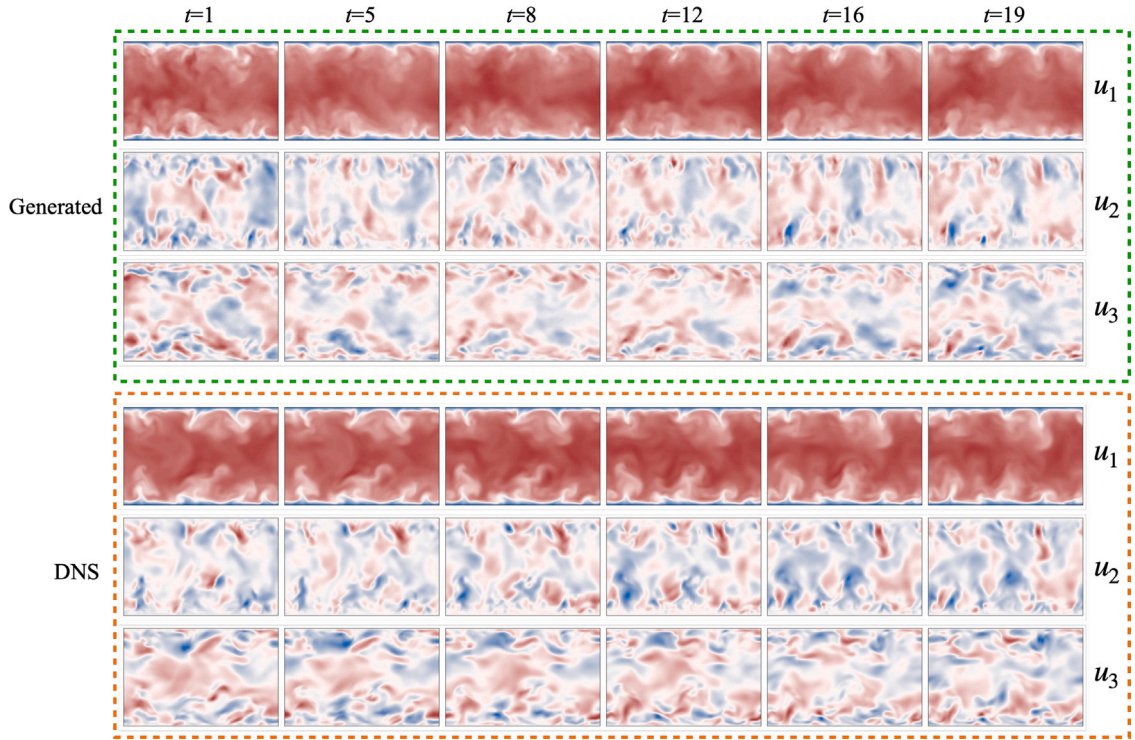
Our model has remarkable flexibility in conditioned generation, capable of producing flow with desired flow features. To demonstrate this capability, we consider generating instantaneous flow sequence given specified mean flow profiles or Reynolds stress fields. Unlike the previous conditional generation methods that rely on adding extra loss terms during the training phase [19], our framework allows for incorporating these additional criteria directly into the inference (sampling) phase, eliminating the need for retraining. This approach is feasible because the specified statistics are differentiable relative to the generated states. Such an approach not only saves the effort of repeated training but also adeptly addresses potential discrepancies between training and testing data, thereby validating the use of online conditioning over offline training.

**Conditioned on 1D mean flow profiles:** As a proof of concept, we conditioned the model on the RMS profiles of wall-normal velocity fluctuation obtained from an unseen DNS sequence. As shown in Fig. 9, while our model generates distinct realizations, the RMS time-averaged fluctuation profile of the generated instantaneous flow agree well with that of the DNS data, suggesting that the condition is successfully imposed during the sampling process. Our model not only generates samples that visually similar to the DNS data but also can enforce the specified flow conditions without requiring retraining for any new mean flow profile requirements. We further tested our model by generating an additional six sequences using the same trained unconditional diffusion model, each conditioned on different RMS mean fluctuation profiles, and observed similarly robust performance (as shown in the lower part of Fig. 9).

**Conditioned on the 2D Reynolds stress field:** We next explored generating flow sequences with specified Reynolds stress fields using the same trained model. In contrast to the 1D RMS mean flow profiles, the Reynolds stress field contains more flow information but also requires a higher-dimensional representation. For this task, we first obtained the Reynolds stress from an unseen DNS flow sequence and then utilized the gradient-based conditional sampling method to generate a sequence of instantaneous flows that



**Fig. 9.** Conditional generation given specified RMS profiles of wall normal velocity fluctuation: The RMS profiles of wall-normal velocity fluctuation from the DNS (—), and recalculated from the generated samples (---). Note that the sequences used to calculate RMS profiles are not in a statistically steady state, resulting in varying profiles across different sequences.



**Fig. 10.** The contours of streamwise ( $u_1$ ), wall-normal ( $u_2$ ), spanwise ( $u_3$ ) velocity at the first, fifth, ninth, thirteenth, seventeenth, twentieth steps generated by diffusion model and DNS. Two models use a same colorbar for the same scalar field to highlight similarity in Reynolds stress and difference in flow fields. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

could lead to the same Reynolds stress field. As shown in Figs. 10 and 11, although the generated instantaneous velocity fields are significantly distinct from those in the DNS data, the six Reynolds stress tensor components computed from the generated flow are almost identical to those in the DNS data. This demonstrates the model's remarkable ability to synthesize instantaneous flow, while simultaneously ensuring the desired complex flow characteristics, such as Reynolds stresses. This capability underscores the potential of our model in simulating intricate turbulent dynamics, where adherence to certain statistical properties is crucial, yet a degree of unpredictability in the flow structures is maintained.

Moreover, it is acknowledged that the performance of generative models diminishes in small data regimes. This limitation can be mitigated by integrating additional prior information through our proposed Bayesian conditional sampling framework. To empirically demonstrate the model's efficacy in addressing limited data scenarios, we conducted an experiment using a training



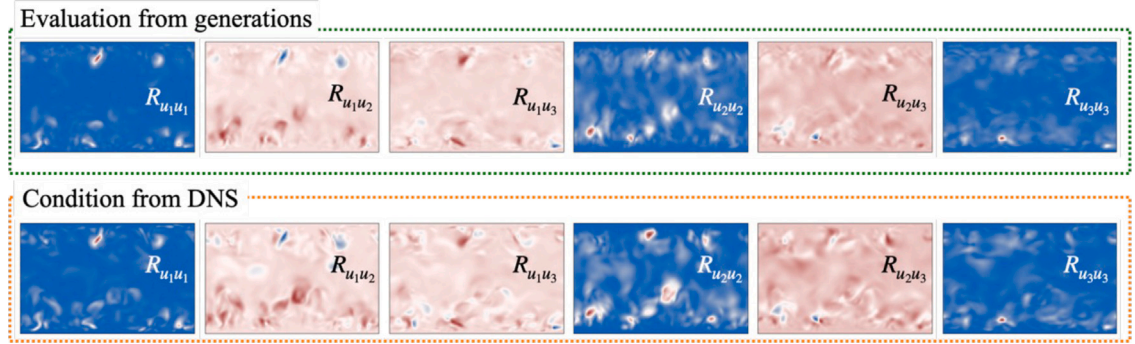


Fig. 11. Re-evaluated and conditioned Reynolds stress between diffusion model and DNS. Two models use a same colorbar for the same scalar field to highlight similarity in Reynolds stress and difference in flow fields. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

dataset that is less than 10% of its original volume. Initially, the model encountered challenges in accurately capturing turbulence characteristics without the integration of prior knowledge. This issue was particularly apparent in the Root-Mean-Square (RMS) of velocity fluctuations, as depicted in Fig. B.27, which exhibited notable discrepancies compared to DNS results. Incorporating prior on the velocity fluctuation RMS through the Bayesian conditional sampling framework significantly enhanced the model's ability to generate spatiotemporal turbulence whose statistics closely align with those from the full dataset. This improvement underscores the value of incorporating prior knowledge as conditions within the Bayesian framework, particularly in limited data scenarios.

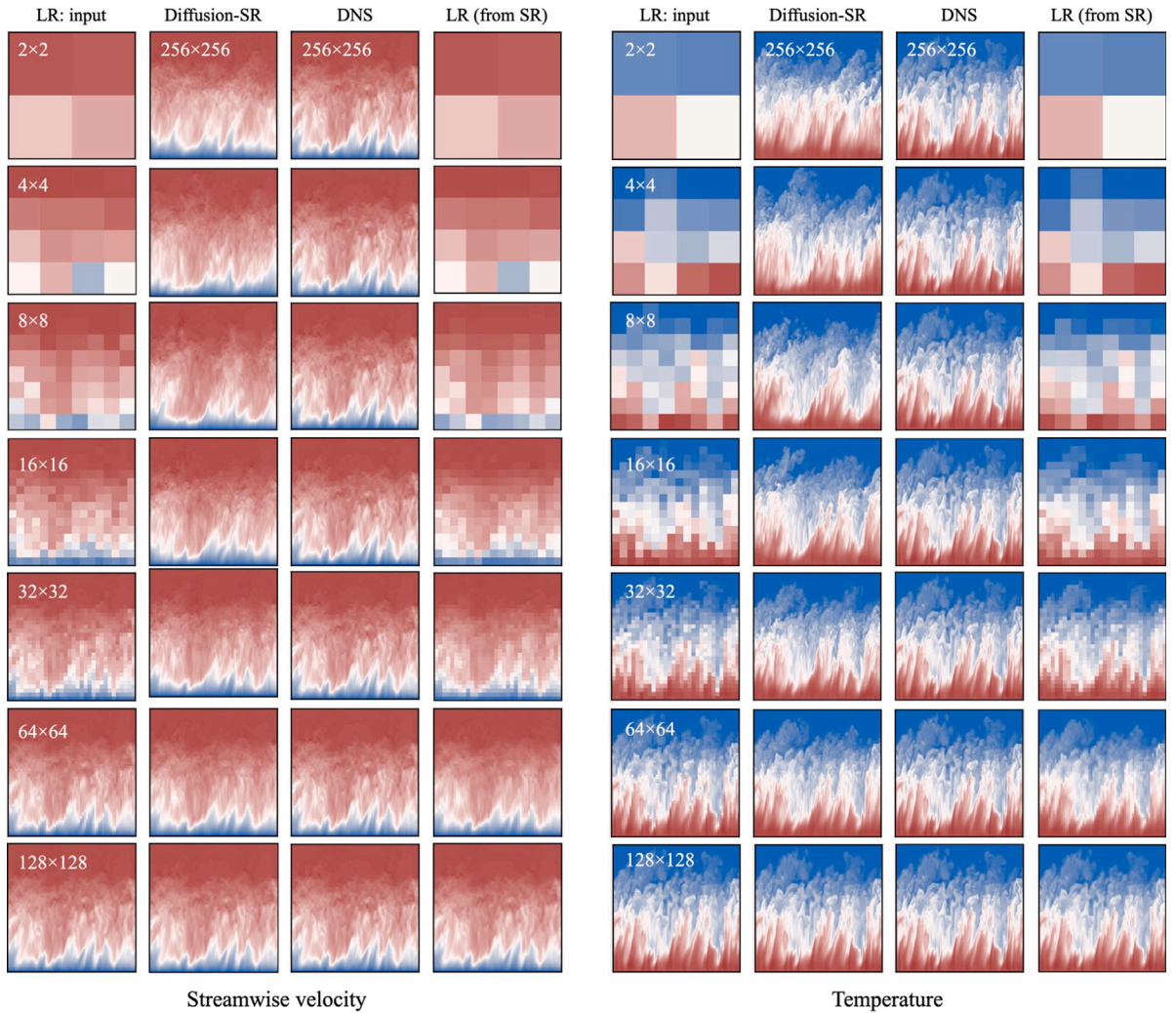
Finally, we briefly discuss the computational cost of different methods, emphasizing the efficiency of our approach. The diffusion model, serving as a surrogate sampler, distinct from traditional numerical simulations by directly generating instantaneous flow within a Bayesian sampling framework. This method not only achieves a remarkably good statistical match and facilitates flexible conditioning but also offers significant speed advantages over conventional simulation methods. Regardless of the training cost, which is around 48 h for this case, we evaluate the inference cost. As shown in Fig. B.26, in this channel flow case, the diffusion model running on a single GPU demonstrates a substantial speedup, being approximately 490 times faster than DNS executed using OpenFOAM [54] on CPUs. Even when compared to DNS conducted using our latest JAX-based, fully vectorized, GPU-enabled CFD solver [6] on the same GPU, the diffusion model maintains a notable speedup, being around 16 times faster.

### 3.3. Super-resolved generation of supersonic turbulent boundary layer

The diffusion model, as a probabilistic modeling technique, offers flexibility in updating the distribution of high-resolution (HR) data based on low-resolution (LR) inputs to yield HR information. Traditional super-resolution (SR) techniques typically learn a deterministic mapping from a fixed input resolution. However, these techniques are not robust or would fail when the resolution of the inputs in inference phase is different from those in the training phase. Moreover, they struggle with extremely LR inputs, as the mapping may become ill-conditioned, leading to ambiguities with multiple possible outputs for a single input. In this section, we explore the efficacy of the proposed diffusion model in generating flow fields from LR data for compressible, supersonic turbulent boundary layer (TBL) flow. Our SR process aims to generate the HR spatiotemporal flow field  $\mathbf{u}(x, y, t) = [u(x, y, t), v(x, y, t), w(x, y, t), T(x, y, t)]^T : \Omega \times \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}^2$ , conditioned on LR flow data, where  $u, v, w, T$  are velocity components and temperature. Specifically, our focus is on examining the enhancement of flow patterns, statistical characteristics, and energy spectra with respect to the different resolution levels of the input LR data.

#### 3.3.1. Data preparation and model training

We extract the 2D data (streamwise wall-normal plane) from a DNS of turbulent boundary layer flows at hypersonic speeds (Mach number  $Ma = 5.86$ ) over a flat plate [55,56]. The computation domain spans  $L_x \times L_y \times L_z = 58.7\delta \times 15.7\delta \times 39.7\delta$ , where  $\delta$  represents the inflow boundary layer thickness ( $\delta = 13.8$  mm). The original DNS resolution in streamwise wall-normal plane is  $1600 \times 500$ . For this study, we collect data from 25 distinct, non-overlapping streamwise sections, with each trajectory containing 160 time steps. To facilitate more efficient processing while retaining critical flow features, we downsampled the spatial resolution to  $256 \times 256$  pixels. Our training dataset comprised 2400 snapshots, covering a broad range of flow features. The model training was conducted once (lasting approximately 90 h), and various SR tasks can be performed by using the gradient-based sampling conditioned on various LR inputs (refer to Section 2.3.1). Note that we did not focus on generating long sequences in this experiment, as this capability has already been demonstrated in Section 3.2.



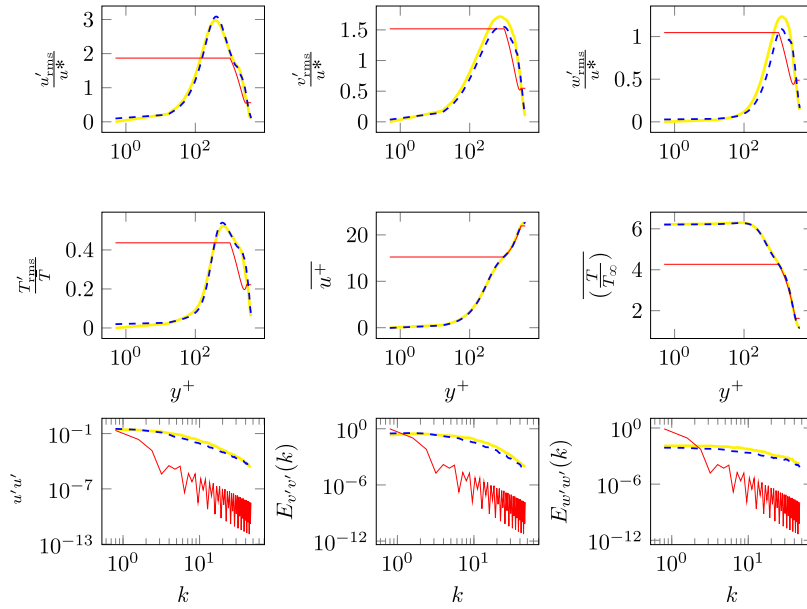
**Fig. 12.** Instantaneous flow fields (streamwise velocity and temperature) of LR input, SR results, DNS and LR downsampled from SR results. The resolution of the LR input varies from  $2 \times 2$ ,  $4 \times 4$ ,  $8 \times 8$ ,  $16 \times 16$ ,  $32 \times 32$ ,  $64 \times 64$  and  $128 \times 128$ . The resolution of both SR results and DNS reference is  $256 \times 256$ .

### 3.3.2. SR generated fields given different low-resolution inputs

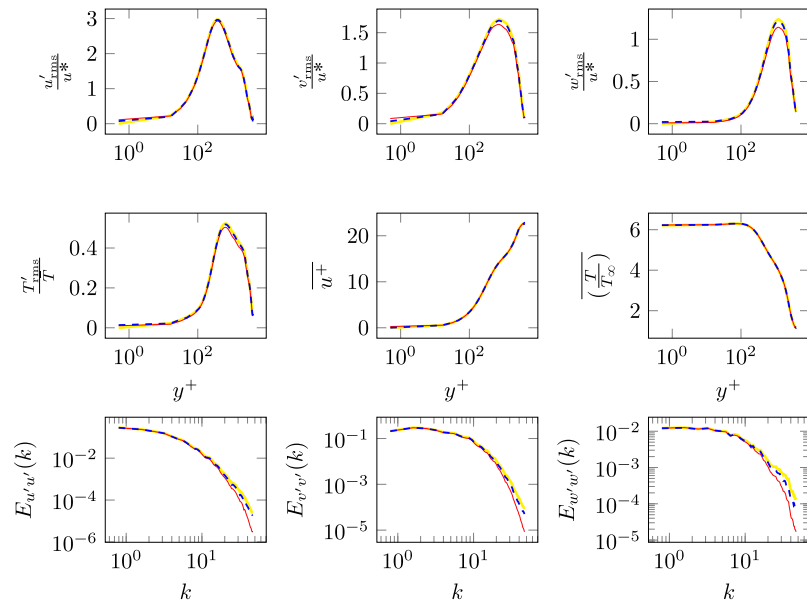
Figs. 12 and C.28 present the SR generated fields of streamwise, spanwise, wall-normal velocities, and temperature, conditioned on the LR inputs at different resolutions. From the visual comparison, it is apparent that the SR diffusion model is capable of enhancing the quality of the input data to produce HR snapshots that resemble the DNS reference to a remarkable degree, even when starting from LR inputs as coarse as  $2 \times 2$ . This suggests that our model possesses an advanced capability to generate the fine details necessary for an HR representation of the flow fields, which is particularly impressive at the lower resolutions. As we progress from the top row (lowest resolution of  $2 \times 2$ ) to the bottom row (highest resolution of  $128 \times 128$ ), there is a notable trend in the SR fields. At the lowest resolutions, the generated HR flow details of the SR output appear more randomized and are different from the DNS reference. As the resolution of the LR input increases, the SR-generated fields exhibit more of the fine structures and variations that are similar to those of the HR DNS reference. The last row ( $128 \times 128$ ) showcases SR output that closely resembles the DNS reference, indicating that our SR diffusion model can generate that particular realization of the DNS instantaneous flow when it is conditioned on higher-resolution input data.

Moreover, we have conducted a comprehensive analysis of flow statistics and energy spectra, as depicted in Figs. 13 and 14 (results for other resolutions of LR inputs are shown in Figs. C.29, C.30, C.31, C.32, and C.33). Remarkably, even from very low-resolution inputs, the model successfully reconstructs the mean velocity and temperature profiles, which were initially indiscernible. The super-resolution process significantly enhances the fluctuating velocity and temperature profiles, especially in the near-wall regions. A critical metric of evaluation is the energy spectrum, which reflects the resolved turbulence scales. For comparative analysis, we upscaled the LR inputs to HR using trilinear interpolation before computing the energy spectrum. Although inputs at a very fine resolution of  $128 \times 128$ , the energy spectrum of trilinear-SR results agree with DNS results well, the discrepancy at





**Fig. 13.** Fluctuation of streamwise, span-wise, wall-normal velocity and temperature, mean profile of streamwise velocity and temperature, energy spectrum (inside boundary layer  $\frac{h}{H} = \frac{4}{256}$ ) of streamwise, span-wise, wall-normal velocity (from left to right, top to bottom) from DNS ( $256 \times 256$ , —), Diffusion-SR ( $256 \times 256$ , - - -) and Trilinear-SR ( $2 \times 2$ , —).



**Fig. 14.** Fluctuation of streamwise, span-wise, wall-normal velocity and temperature, mean profile of streamwise velocity and temperature, energy spectrum (inside boundary layer  $\frac{h}{H} = \frac{4}{256}$ ) of streamwise, span-wise, wall-normal velocity (from left to right, top to bottom) from DNS ( $256 \times 256$ , —), Diffusion-SR output ( $256 \times 256$ , - - -) and Trilinear-SR ( $128 \times 128$ , —).

high wave numbers are still notable. The proposed model adeptly bridges these gaps, facilitating precise and accurate recovery of the missing energy and turbulence scales. The energy spectra of the diffusion-SR results are almost identical to those of DNS reference, regardless of the input resolutions. This analysis reveals that our model effectively recovers this spectrum from LR inputs, indicating its efficiency in reproducing small-scale turbulence. Notably, the CRPS values presented in Table D.6 highlight that, in SR scenarios where our model is conditioned on low-resolution (LR) data, it is capable of generating samples that align more closely with the HR truth than those produced by eddy-resolving simulations without LR information. It is important to note that our diffusion model is trained given DNS data once and then can be used for SR generation given different LR input resolutions without retraining. This is

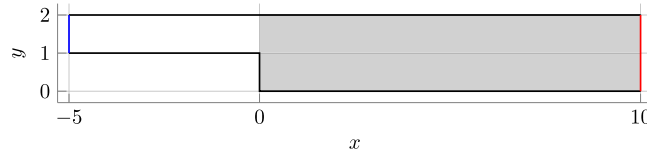


Fig. A.15. Geometry, domain of interest, and boundary conditions for the case of backward-facing step. Boundary conditions: inlet (—), outlet (—) and no-slip wall (—). Shadow region is the domain of interest for generative modeling.

in stark contrast to previous SR works, which are highly reliant on the resolution used in training. Our model is much more robust and generalizable to LR input with different resolutions and qualities.

#### 4. Conclusion

This study has introduced a novel Bayesian conditional diffusion model for generating spatiotemporal turbulent flows. Our proposed model unifies unconditional and conditional sampling strategies, offering a versatile solution across a spectrum of scenarios. We have systematically presented conditional sampling approaches that employs a gradient-based method for scenarios with a directly differentiable relationship between conditions and outcomes, and a replacement-based method for cases without such explicit relationship. Another novel aspect of our framework is its capability in effectively generating arbitrarily long sequences of turbulence flow through autoregressive gradient-based conditional sampling, negating the need for iterative retraining. Our empirical investigations underscore the model's adeptness in tackling a variety of turbulence generation tasks. The model exhibits prowess in synthesizing LES-like eddy-resolved turbulence from URANS inputs, producing turbulent channel flow sequences conditioned on desired flow statistics, and performing super-resolution generation on supersonic turbulent boundary layer flows from low-resolution input data with different resolutions and qualities. These capabilities not only demonstrate the model's versatility but also its potential to revolutionize the field of turbulence modeling and simulation.

Looking forward, we identify several promising trajectories for advancing this research. In scenarios involving differentiable conditions, enhancing estimation accuracy through refined sampling methodologies, such as importance sampling and particle filtering, remains a key area of interest. For non-differentiable conditions, exploring innovative strategies like reinforcement learning and ensemble Kalman filtering could yield significant advancements. Moreover, adapting our model to accommodate large-scale, graph-based data opens up new possibilities for tackling the complexities associated with general unstructured mesh data. Pursuing these avenues promises to yield substantial contributions to the domain of computational fluid dynamics and beyond.

#### CRediT authorship contribution statement

**Han Gao:** Investigation, Conceptualization, Formal analysis, Methodology, Software, Visualization, Writing – original draft. **Xu Han:** Software, Methodology, Conceptualization, Investigation. **Xiantao Fan:** Writing – review & editing, Visualization, Validation, Software, Investigation, Formal analysis, Data curation. **Luning Sun:** Software, Investigation. **Li-Ping Liu:** Supervision, Methodology. **Lian Duan:** Writing – review & editing, Investigation, Data curation. **Jian-Xun Wang:** Supervision, Resources, Project administration, Methodology, Funding acquisition, Conceptualization, Writing – original draft, Writing – review & editing.

#### Declaration of competing interest

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

#### Data availability

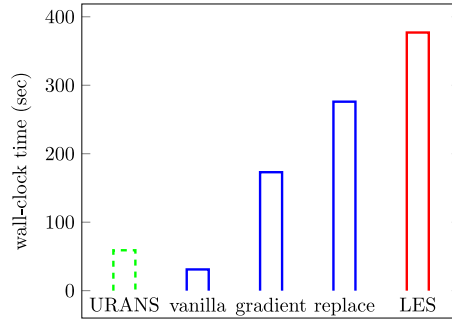
Data will be made available on request.

#### Acknowledgments

The authors would like to acknowledge the funds from Office of Naval Research, USA under award numbers N00014-23-1-2071 and National Science Foundation, USA under award numbers OAC-2047127. XF would also like to acknowledge the fellowship provided by the Environmental Change Initiative and Center for Sustainable Energy at University of Notre Dame. The content of this publication does not necessarily reflect the position or policy of any of these supporters, and no official endorsement should be inferred.

#### Appendix A. Supplementary results of flow over backward-facing step

See Figs. A.15–A.20 and Table A.3.



**Fig. A.16.** The wall-clock time for sampling a sequence of 240 time steps using variants of diffusion method (—) and LES. The URANS (not as a sample method) here is provided for the reference.

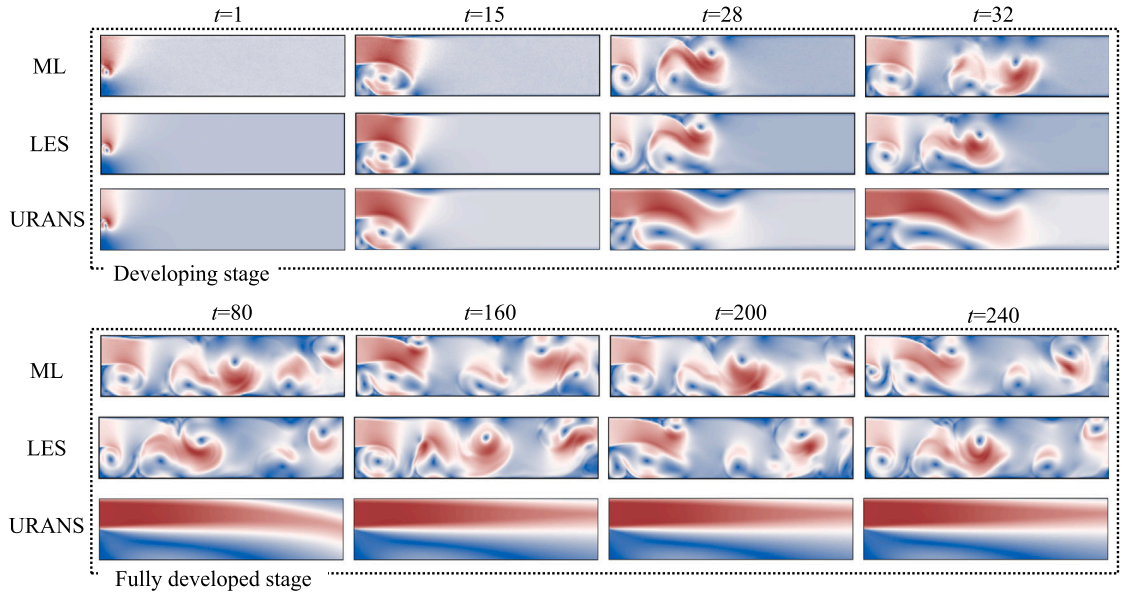
**Table A.3**

The OpenFOAM is performed on AMD-Ryzen-9-7950, and diffusion model is evaluated on NVIDIA-GeForce-RTX-4090. In this case, we reduce the dimension of hyper-parameters by setting  $\sigma_{N_{\text{noise}}}^{\text{guide}} = \dots = \sigma_2^{\text{guide}} = \sigma^{\text{guide}}(1)$ ,  $\sigma_1^{\text{guide}} = \sigma^{\text{guide}}(2)$ ,  $\rho_{N_{\text{noise}}}^{\text{guide}} = \dots = \rho_2^{\text{guide}} = \rho^{\text{guide}}(1)$ ,  $\rho_1^{\text{guide}} = \rho^{\text{guide}}(2)$ .

	Turbulence modeling	Height	Width	# step	Inlet perturbation	Solver
URANS	kEpsilon [57]	64	256	240	No	OpenFOAM [54]
LES	dynamicK [58]	64	256	240	Yes	OpenFOAM [54]

	Diff	$N_{\text{noise}}$	$N_{\text{inpaint}}$	$\sigma^{\text{guide}}$	$\rho^{\text{guide}}$	$N_{\text{previous}}$	$N_{\text{length}}$	device
Vanilla		20	None	None	None	0	40	NVIDIA
Gradient		20	$N_{\text{inpaint}}$	(100, 100)	(0.02, 0.02)	20	40	NVIDIA
Replace		20	5	None	None	20	40	NVIDIA



**Fig. A.17.** Velocity magnitude of generated samples (from first to fifth row) and URANS (last row) at  $Re = 13100$  from developing to fully-developed stage (from left to right).

## Appendix B. Supplementary results of 3D turbulent channel flow

See Figs. B.21–B.27 and Table B.4.

## Appendix C. Supplementary results of supersonic turbulent boundary layers

See Figs. C.28–C.33 and Table C.5.

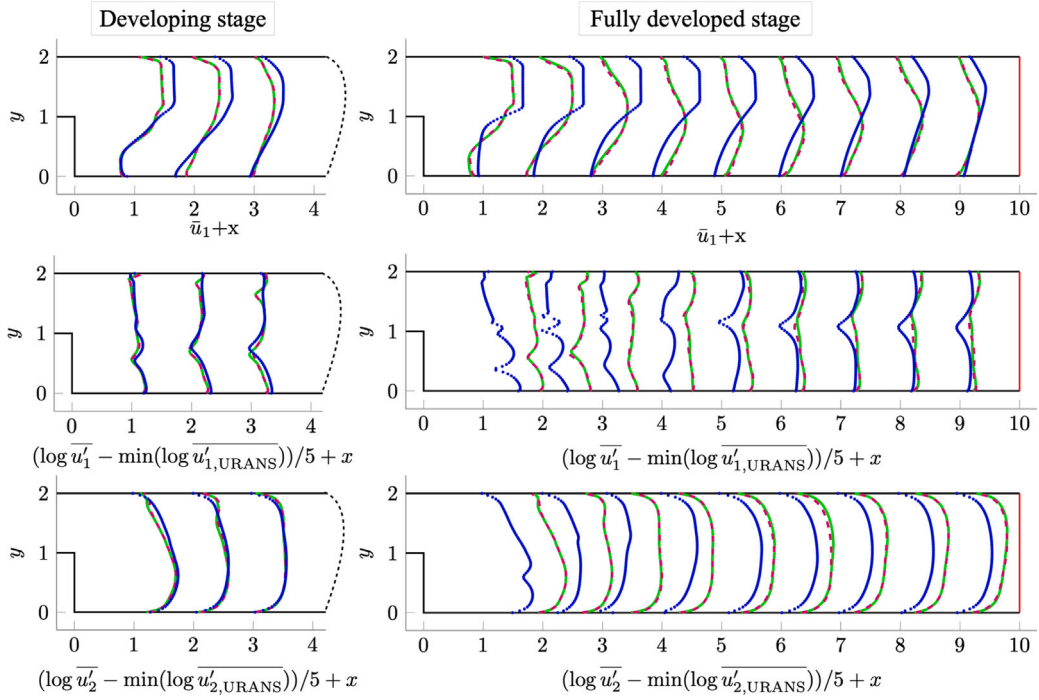


Fig. A.18. Mean of streamwise velocity, variance of streamwise velocity and variance of wall-normal velocity at  $Re = 13100$  from LES (●), URANS (●) and diffusion model (●).

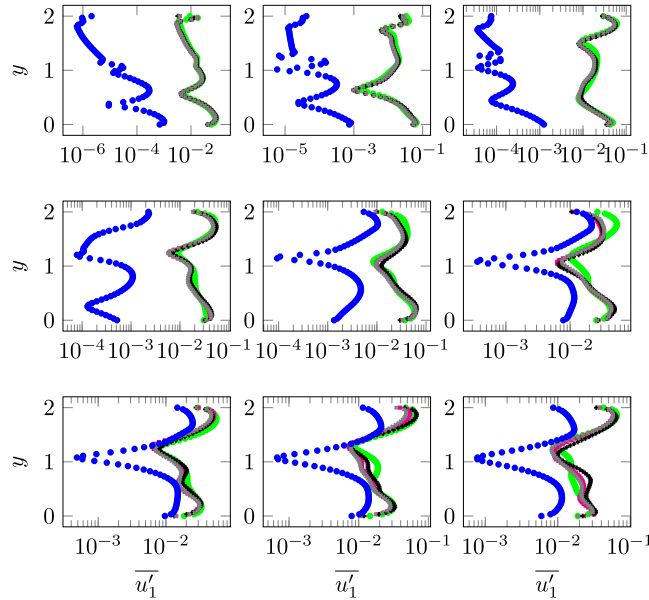
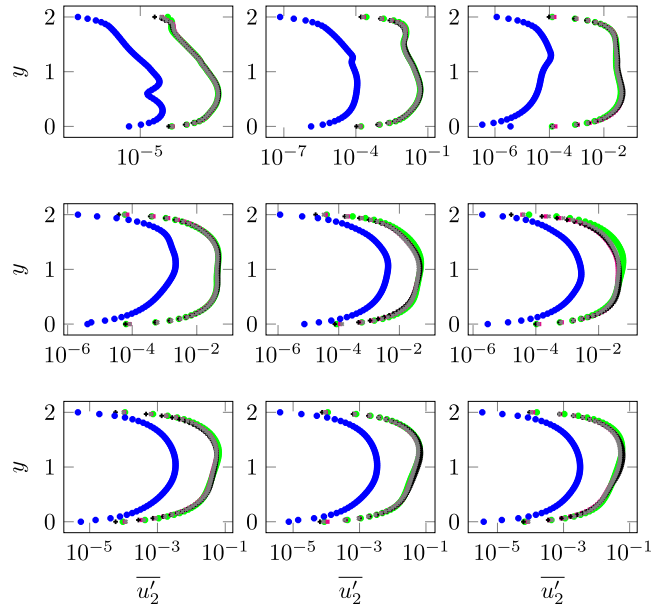


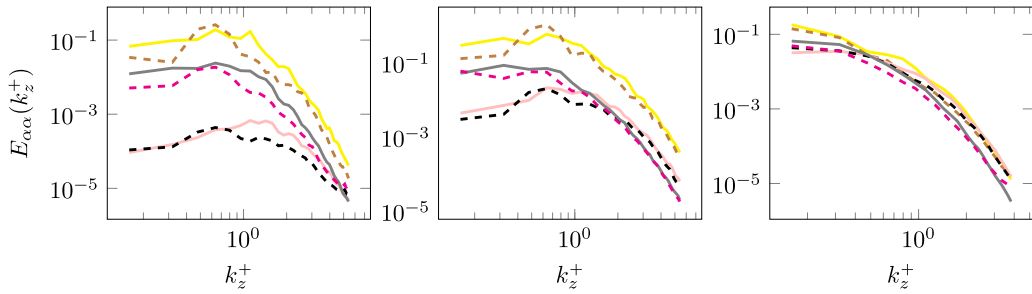
Fig. A.19. Variance of streamwise velocity at  $Re = 13100$  in the fully-developed phase: LES (●), URANS (●), Diff-Gradient (●), Diff-Replace (●) and Diff-Vanilla (●). The figures, arranged from left to right and top to bottom, correspond to locations at  $x = [1, 9]$ .

## Appendix D. Continuous ranked probability score

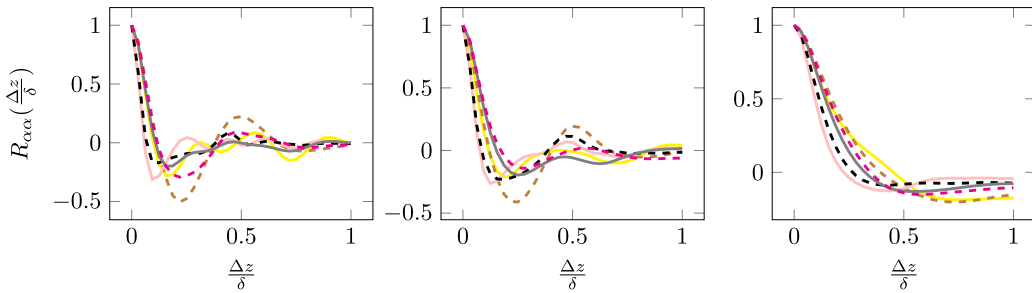
See Table D.6.



**Fig. A.20.** Variance of wall-normal velocity at  $Re = 13100$  in the fully-developed phase: LES (●), URANS (●), Diff-Gradient (●), Diff-Replace (●) and Diff-Vanilla (●). The figures, arranged from left to right and top to bottom, correspond to locations at  $x = [1, 9]$ .



**Fig. B.21.** Energy spectra of the streamwise velocity from DNS (—), diffusion model (---), of the wall-normal velocity from DNS (—), diffusion model (---), of the spanwise velocity from DNS (—) and diffusion model (---) at  $y^+ = 5$  (left),  $y^+ = 20$  (middle) and  $y^+ = 180$  (right).

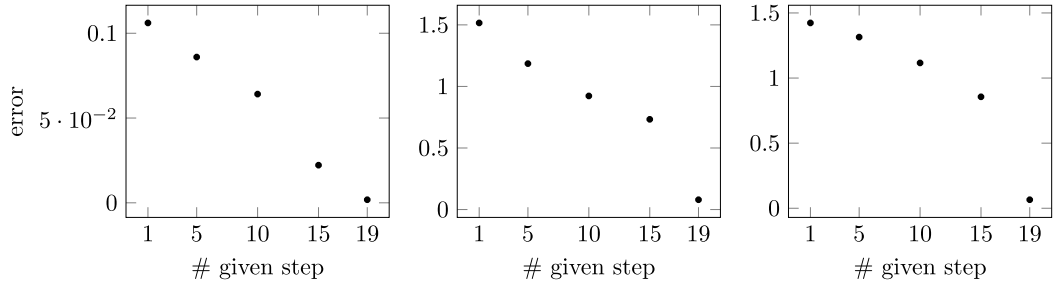


**Fig. B.22.** Spatial-correlations of the streamwise velocity from DNS (—), diffusion model (---), of the wall-normal velocity from DNS (—), diffusion model (---), of the spanwise velocity from DNS (—) and diffusion model (---) at  $y^+ = 5$  (left),  $y^+ = 20$  (middle) and  $y^+ = 180$  (right).

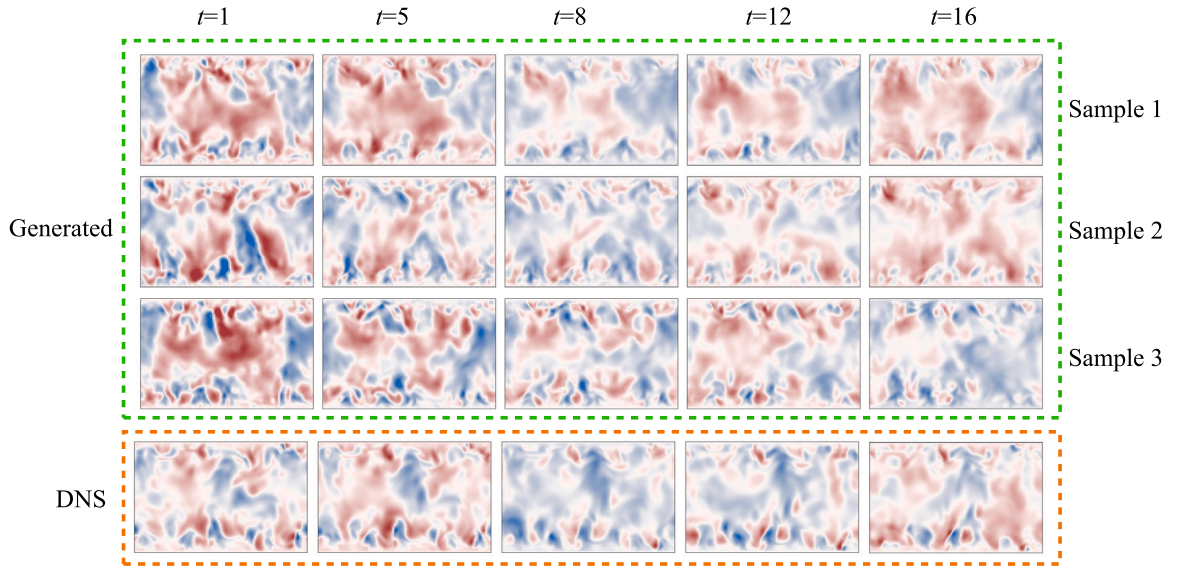
## Appendix E. Supplementary data

Supplementary material related to this article can be found online at <https://doi.org/10.1016/j.cma.2024.117023>.

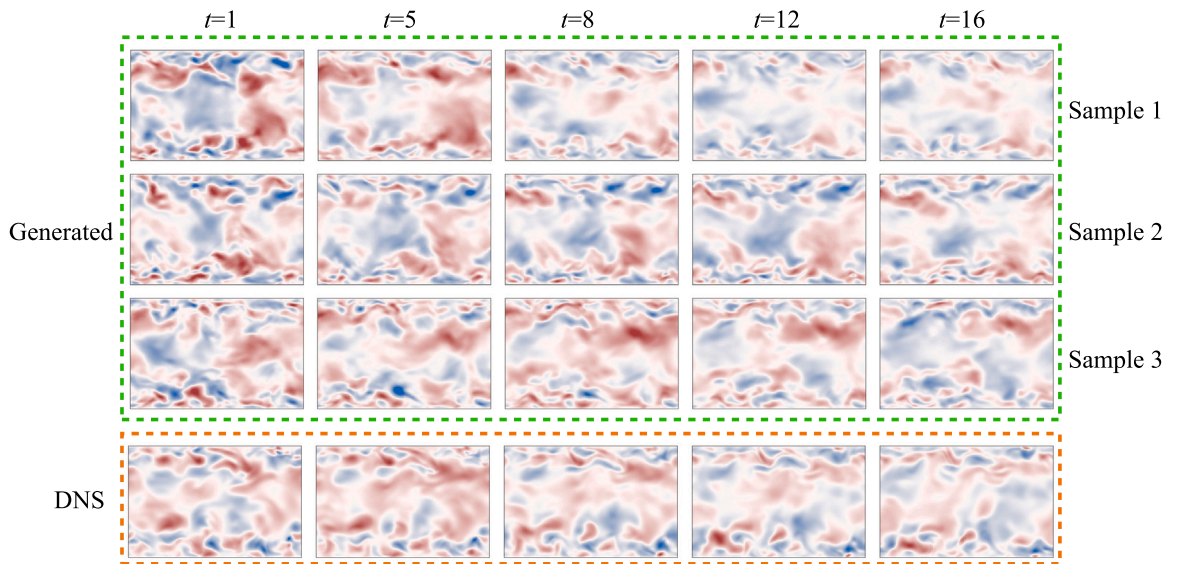




**Fig. B.23.** The error of streamwise (left), wall-normal (middle), spanwise (right) velocity at the twenty step generated by diffusion model given the first, the first five, the first ten, the first fifteen and the first nineteen steps.



**Fig. B.24.** Continuous instantaneous wall-normal velocity fields of samples from diffusion model and DNS.



**Fig. B.25.** Continuous instantaneous spanwise velocity fields of samples from diffusion model and DNS.

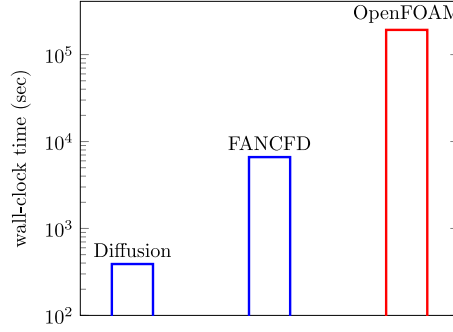
**Table B.4**

In this case, we reduce the dimension of hyper-parameters by setting  $\sigma_{N_{\text{noise}}}^{\text{guide}} = \dots = \sigma_2^{\text{guide}} = \sigma^{\text{guide}}(1)$ ,  $\sigma_1^{\text{guide}} = \sigma^{\text{guide}}(2)$ ,  $\beta_{N_{\text{noise}}}^{\text{guide}} = \dots = \beta_2^{\text{guide}} = \beta^{\text{guide}}(1)$ ,  $\beta_1^{\text{guide}} = \beta^{\text{guide}}(2)$ .

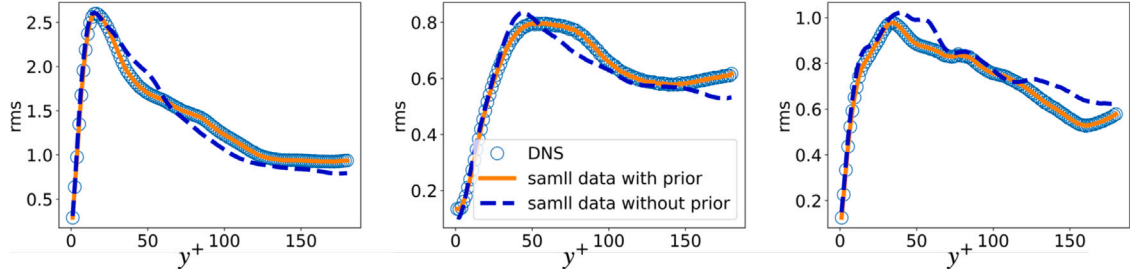
	Solver	Height	Width
	OpenFOAM [54]	256	128
	FanCFD [6]	256	128

Sampling	$N_{\text{noise}}$	$N_{\text{inpaint}}$	$\sigma^{\text{guide}}$	$\beta^{\text{guide}}$	$N_{\text{previous}}$	$N_{\text{length}}$
Unconditional	20	None	(100,100)	(0.018,0.018)	10	20
Condition on initial condition	20	5	None	None	1 – 19	20
Condition on rms	20	None	(100,100)	(0.015,0.015)	None	20
Condition on Reynolds stress	20	None	(100,100)	(0.013,0.013)	None	20



**Fig. B.26.** The wall-clock time for sampling a flowthrough (a sequence of 300 time steps) using the gradient-diffusion method and DNS methods. Specifically, OpenFOAM [54] is performed on CPU (—, Intel Xeon(R) Gold 6138) with 16 cores in parallel; FANCFD [6] and diffusion are performed on single GPU card (—, NVIDIA-GeForce-RTX-4090).



**Fig. B.27.** The RMS of velocity fluctuations for small dataset (10% of the original dataset) with and without prior knowledge. From left to right, they are streamwise velocity  $u$ , wall normal velocity  $v$  and spanwise velocity  $w$ , respectively.

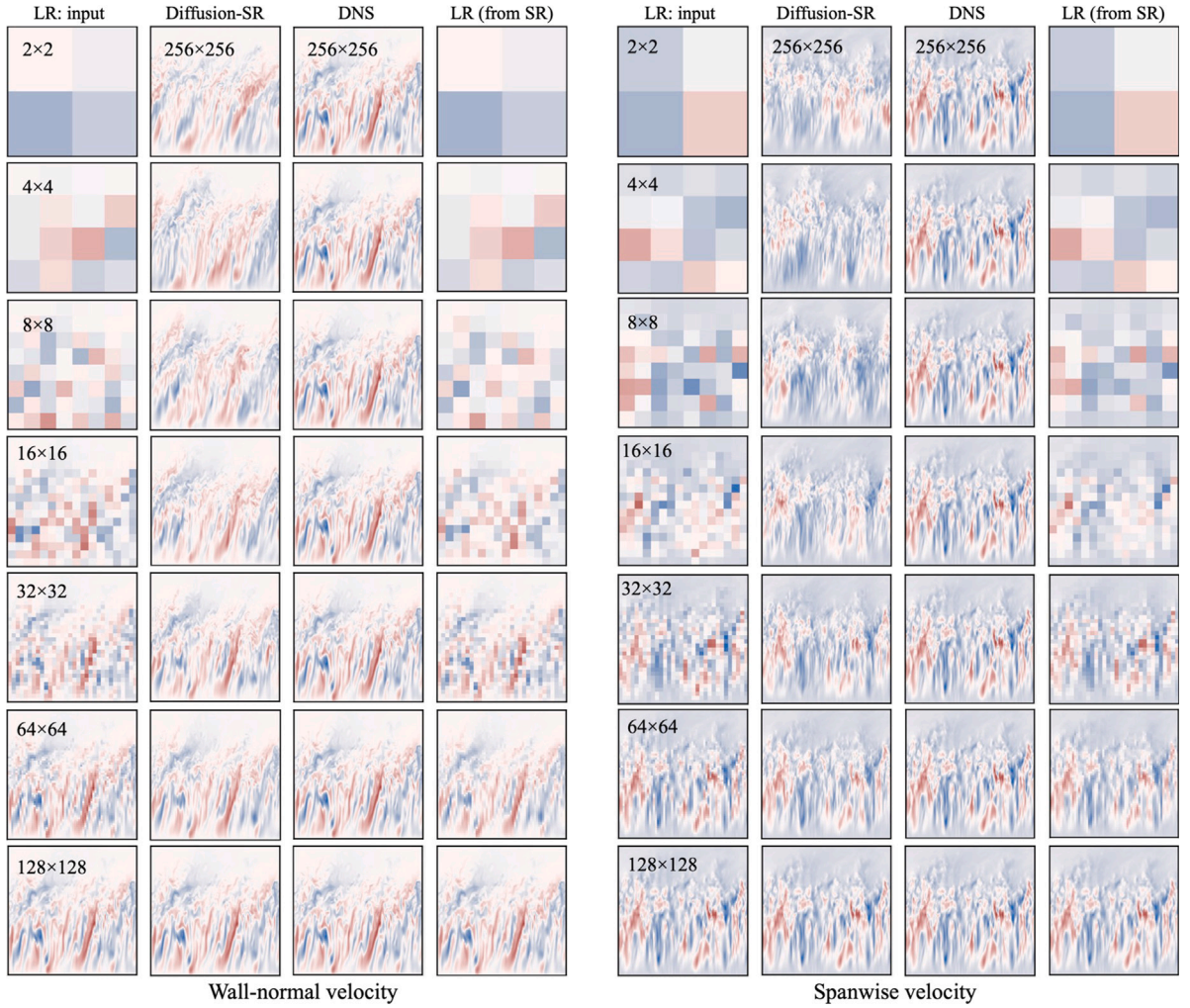
**Table C.5**

In this case, we reduce the dimension of hyper-parameters by setting  $\sigma_{N_{\text{noise}}}^{\text{guide}} = \dots = \sigma_2^{\text{guide}} = \sigma^{\text{guide}}(1)$ ,  $\sigma_1^{\text{guide}} = \sigma^{\text{guide}}(2)$ ,  $\beta_{N_{\text{noise}}}^{\text{guide}} = \dots = \beta_2^{\text{guide}} = \beta^{\text{guide}}(1)$ ,  $\beta_1^{\text{guide}} = \beta^{\text{guide}}(2)$ .

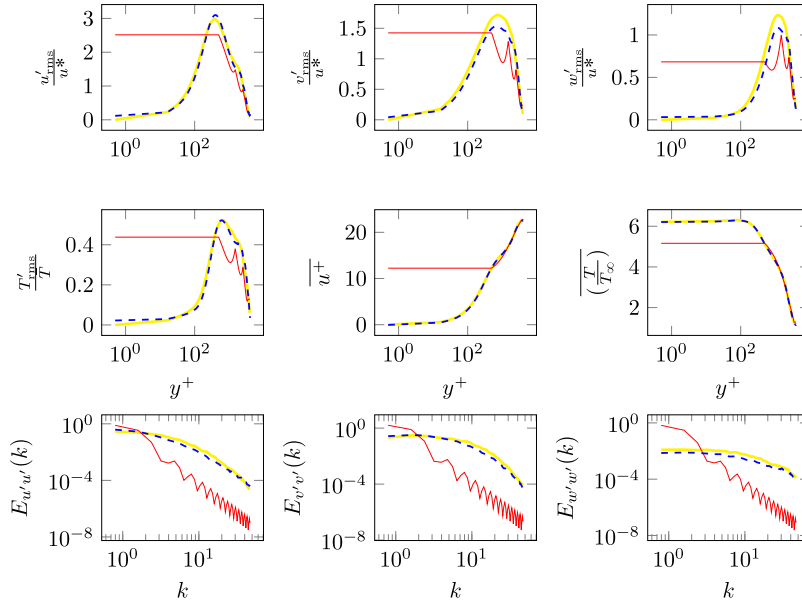
	Solver	Height	Width
	DuanCFD [55]	256	256

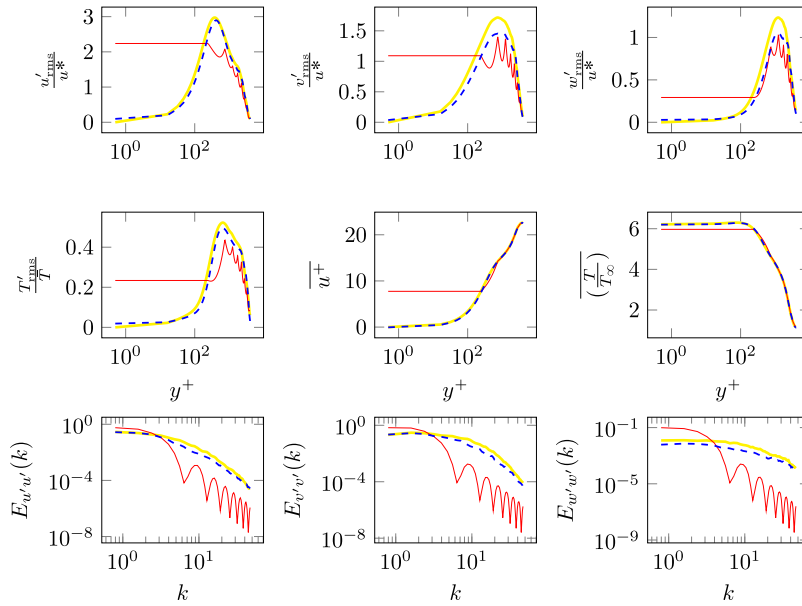
Input resolution	$N_{\text{noise}}$	$N_{\text{inpaint}}$	$\sigma^{\text{guide}}$	$\beta^{\text{guide}}$	$N_{\text{refine}}$	$N_{\text{length}}$
$2 \times 2$	10	None	(100,100)	(0.015,0.001)	10	10
$4 \times 4$	10	None	(100,100)	(0.015,0.001)	10	10
$8 \times 8$	10	None	(100,100)	(0.01,0.001)	10	10
$16 \times 16$	10	None	(100,100)	(0.02,0.02)	1	10
$32 \times 32$	10	None	(100,100)	(0.05,0.02)	1	10
$64 \times 64$	10	None	(100,100)	(0.06,0.02)	1	10
$128 \times 128$	10	None	(100,100)	(0.08,0.0035)	120	10



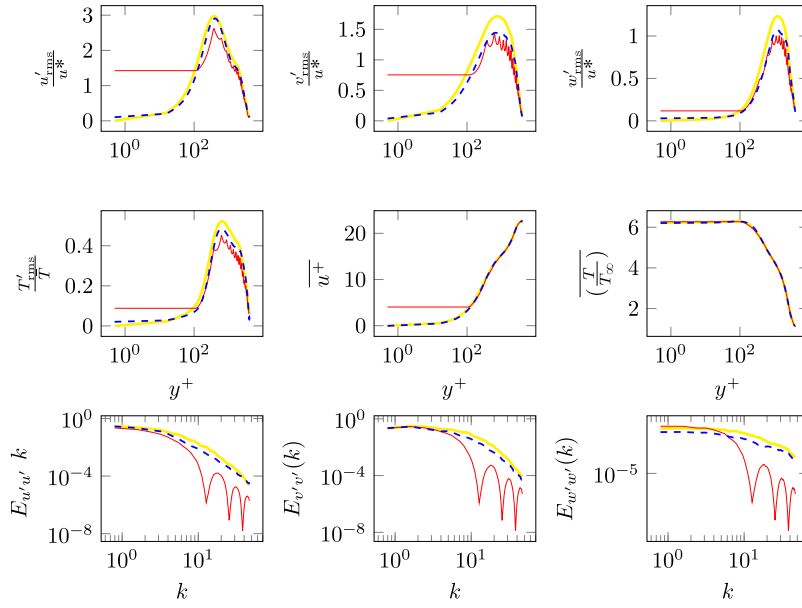
**Fig. C.28.** Instantaneous flow fields (span-wise and wall-normal velocity) of LR input, SR results, DNS and LR downsampled from SR results. The resolution of the LR input varies from  $2 \times 2$ ,  $4 \times 4$ ,  $8 \times 8$ ,  $16 \times 16$ ,  $32 \times 32$ ,  $64 \times 64$  and  $128 \times 128$ . The resolution of both SR results and DNS reference is  $256 \times 256$ .



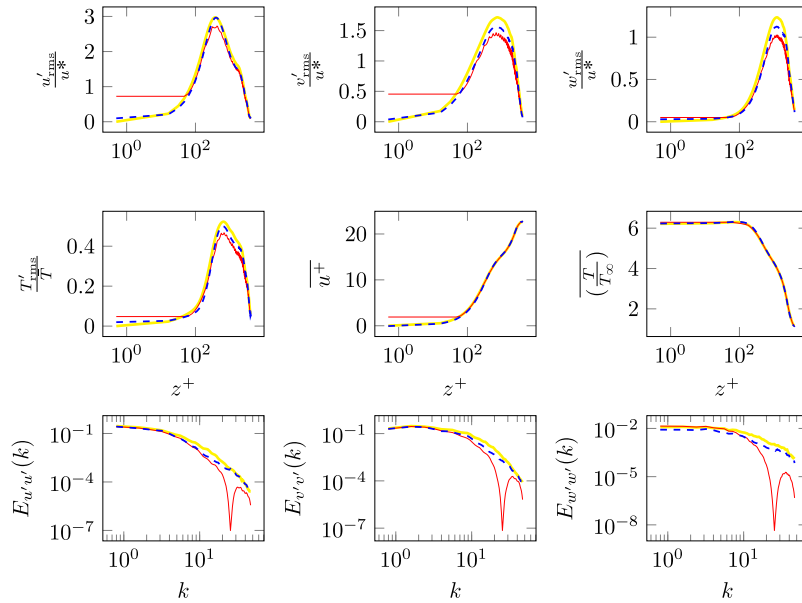
**Fig. C.29.** Fluctuation of streamwise, span-wise, wall-normal velocity and temperature, mean profile of streamwise velocity and temperature, energy spectrum (inside boundary layer  $\frac{h}{H} = \frac{1}{256}$ ) of streamwise, span-wise, wall-normal velocity (from left to right, top to bottom) from DNS ( $256 \times 256$ , —), SR result ( $256 \times 256$ , - - -) and LR input ( $4 \times 4$ , —).



**Fig. C.30.** Fluctuation of streamwise, span-wise, wall-normal velocity and temperature, mean profile of streamwise velocity and temperature, energy spectrum (inside boundary layer  $\frac{h}{H} = \frac{1}{256}$ ) of streamwise, span-wise, wall-normal velocity (from left to right, top to bottom) from DNS ( $256 \times 256$ , —), SR result ( $256 \times 256$ , - - -) and LR input ( $8 \times 8$ , —).

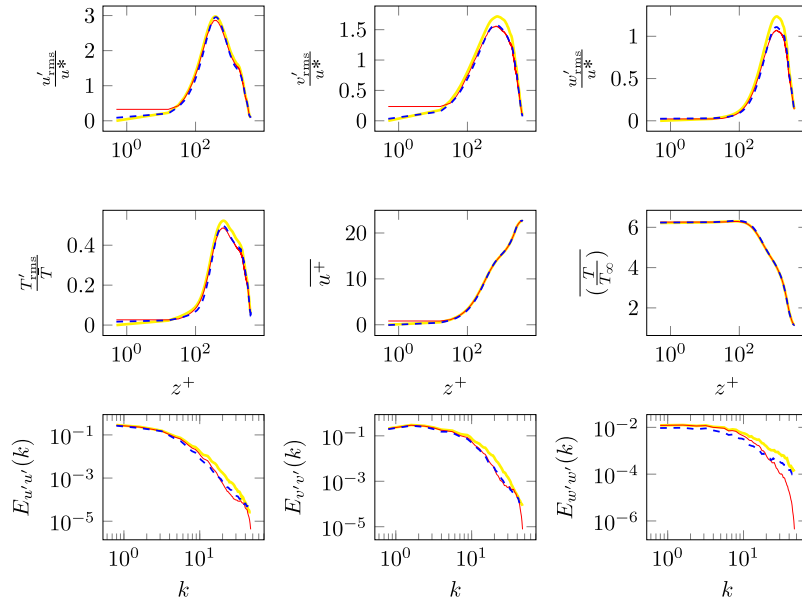


**Fig. C.31.** Fluctuation of streamwise, span-wise, wall-normal velocity and temperature, mean profile of streamwise velocity and temperature, energy spectrum (inside boundary layer  $\frac{h}{\delta} = \frac{1}{256}$ ) of streamwise, span-wise, wall-normal velocity (from left to right, top to bottom) from DNS ( $256 \times 256$ , —), SR result ( $256 \times 256$ , - - -) and LR input ( $16 \times 16$ , —).



**Fig. C.32.** Fluctuation of streamwise, span-wise, wall-normal velocity and temperature, mean profile of streamwise velocity and temperature, energy spectrum (inside boundary layer  $\frac{h}{\delta} = \frac{1}{256}$ ) of streamwise, span-wise, wall-normal velocity (from left to right, top to bottom) from DNS ( $256 \times 256$ , —), SR output ( $256 \times 256$ , - - -) and LR input ( $32 \times 32$ , —).





**Fig. C.33.** Fluctuation of streamwise, span-wise, wall-normal velocity and temperature, mean profile of streamwise velocity and temperature, energy spectrum (inside boundary layer  $\frac{h}{H} = \frac{4}{256}$ ) of streamwise, span-wise, wall-normal velocity (from left to right, top to bottom) from DNS ( $256 \times 256$ , —), SR output ( $256 \times 256$ , - - -) and LR input ( $64 \times 64$ , —).

**Table D.6**

Continuous Ranked Probability Score (CRPS) for all three cases. We calculate the CRPS by contrasting distributions between two distinct sets of simulated samples and those generated by our diffusion model alongside numerical simulations. This meticulous comparison between independent simulation groups and model-generated samples enables us to rigorously evaluate how well the diffusion model replicates the statistical nuances of the simulated data.

Case	Generated samples	Simulation samples
LES-like generation based on URANS	0.114	0.119
Generation of instantaneous turbulent channel flow	0.0261	0.0313
SR of supersonic turbulent boundary layer	0.0461	0.064

## References

- [1] Steven L. Brunton, Bernd R. Noack, Petros Koumoutsakos, Machine learning for fluid mechanics, *Annu. Rev. Fluid Mech.* 52 (2020) 477–508.
- [2] Ricardo Vinuesa, Steven L. Brunton, Enhancing computational fluid dynamics with machine learning, *Nat. Comput. Sci.* 2 (6) (2022) 358–366.
- [3] Yohai Bar-Sinai, Stephan Hoyer, Jason Hickey, Michael P Brenner, Learning data-driven discretizations for partial differential equations, *Proc. Natl. Acad. Sci.* 116 (31) (2019) 15344–15349.
- [4] Dmitrii Kochkov, Jamie A Smith, Ayya Alieva, Qing Wang, Michael P Brenner, Stephan Hoyer, Machine learning-accelerated computational fluid dynamics, *Proc. Natl. Acad. Sci.* 118 (21) (2021) e2101784118.
- [5] Xin-yang Liu, Min Zhu, Lu Lu, Hao Sun, Jian-Xun Wang, Multi-resolution partial differential equations preserved learning framework for spatiotemporal dynamics, *Commun. Phys.* 7 (1) (2024) 31.
- [6] Xiantao Fan, Jian xun Wang, Differentiable hybrid neural modeling for fluid-structure interaction, *J. Comput. Phys.* (2023) 112584.
- [7] Jian-Xun Wang, Jin-Long Wu, Heng Xiao, Physics-informed machine learning approach for reconstructing Reynolds stress modeling discrepancies based on DNS data, *Phys. Rev. Fluids* 2 (3) (2017) 034603.
- [8] Romit Maulik, Omer San, Adil Rasheed, Prakash Vedula, Subgrid modelling for two-dimensional turbulence using neural networks, *J. Fluid Mech.* 858 (2019) 122–144.
- [9] Romit Maulik, Himanshu Sharma, Saumil Patel, Bethany Lusch, Elise Jennings, A turbulent eddy-viscosity surrogate modeling framework for Reynolds-averaged Navier-Stokes simulations, *Comput. & Fluids* 227 (2021) 104777.
- [10] Jonathan Tompson, Kristofer Schlachter, Pablo Sprechmann, Ken Perlin, Accelerating eulerian fluid simulation with convolutional networks, in: *International Conference on Machine Learning*, PMLR, 2017, pp. 3424–3433.
- [11] Maziar Raissi, Hessam Babae, Peyman Givi, Deep learning of turbulent scalar mixing, *Phys. Rev. Fluids* 4 (12) (2019) 124501.
- [12] David M. Blei, Andrew Y. Ng, Michael I. Jordan, Latent Dirichlet allocation, *J. Mach. Learn. Res.* 3 (Jan) (2003) 993–1022.
- [13] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, Generative adversarial nets, in: Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, K.Q. Weinberger (Eds.), in: *Advances in Neural Information Processing Systems*, vol. 27, Curran Associates, Inc, 2014.
- [14] Ivan Kobyzev, Simon J.D. Prince, Marcus A. Brubaker, Normalizing flows: An introduction and review of current methods, *IEEE Trans. Pattern Anal. Mach. Intell.* 43 (11) (2020) 3964–3979.
- [15] Lucas Theis, Aäron van den Oord, Matthias Bethge, A note on the evaluation of generative models, 2015, arXiv preprint arXiv:1511.01844.
- [16] Kai Fukami, Yusuke Nabae, Ken Kawai, Koji Fukagata, Synthetic turbulent inflow generator using machine learning, *Phys. Rev. Fluids* 4 (6) (2019) 064603.

- [17] Kai Fukami, Koji Fukagata, Kunihiko Taira, Super-resolution reconstruction of turbulent flows with machine learning, *J. Fluid Mech.* 870 (2019) 106–120.
- [18] Kai Fukami, Koji Fukagata, Kunihiko Taira, Machine-learning-based spatio-temporal super resolution reconstruction of turbulent flows, *J. Fluid Mech.* 909 (2021) A9.
- [19] Mustafa Z. Yousif, Linqi Yu, HeeChang Lim, Physics-guided deep learning for generating turbulent inflow conditions, *J. Fluid Mech.* 936 (2022) A21.
- [20] Mustafa Z Yousif, Meng Zhang, Linqi Yu, Ricardo Vinuesa, HeeChang Lim, A transformer-based synthetic-inflow generator for spatially developing turbulent boundary layers, *J. Fluid Mech.* 957 (2023) A6.
- [21] Romit Maulik, Kai Fukami, Nesar Ramachandra, Koji Fukagata, Kunihiko Taira, Probabilistic neural networks for fluid flow surrogate modeling and data recovery, *Phys. Rev. Fluids* 5 (10) (2020) 104401.
- [22] Masaki Morimoto, Kai Fukami, Romit Maulik, Ricardo Vinuesa, Koji Fukagata, Assessments of epistemic uncertainty using Gaussian stochastic weight averaging for fluid-flow regression, *Physica D* 440 (2022) 133454.
- [23] Karen Stengel, Andrew Glaws, Dylan Hettinger, Ryan N King, Adversarial super-resolution of climatological wind and solar data, *Proc. Natl. Acad. Sci.* 117 (29) (2020) 16805–16815.
- [24] Claudia Drygala, Benjamin Winhart, Francesca di Mare, Hanno Gottschalk, Generative modeling of turbulence, *Phys. Fluids* 34 (3) (2022).
- [25] Zhiwen Deng, Chuangxin He, Yingzheng Liu, Kyung Chun Kim, Super-resolution reconstruction of turbulent velocity fields using a generative adversarial network-based artificial intelligence framework, *Phys. Fluids* 31 (12) (2019).
- [26] Hyojin Kim, Junhyuk Kim, Sungjin Won, Changhoon Lee, Unsupervised deep learning for super-resolution reconstruction of turbulence, *J. Fluid Mech.* 910 (2021) A29.
- [27] Alejandro Güemes, Stefano Discetti, Andrea Ianiro, Beril Sirmacek, Hossein Azizpour, Ricardo Vinuesa, From coarse wall measurements to turbulent velocity fields through deep learning, *Phys. Fluids* 33 (7) (2021) 075121.
- [28] Linqi Yu, Mustafa Z Yousif, Meng Zhang, Sergio Hoyas, Ricardo Vinuesa, Hee-Chang Lim, Three-dimensional ESRGAN for super-resolution reconstruction of turbulent flows with trubic interpolation-based transfer learning, *Phys. Fluids* 34 (12) (2022).
- [29] Michele Buzzicotti, Fabio Bonaccorso, P Clark Di Leoni, Luca Biferale, Reconstruction of turbulent data with deep generative models for semantic inpainting from TURB-rot database, *Phys. Rev. Fluids* 6 (5) (2021) 050503.
- [30] You Xie, Erik Franz, Mengyu Chu, Nils Thuerey, tempoGAN: A temporally coherent, volumetric GAN for super-resolution fluid flow, *ACM Trans. Graph.* 37 (4) (2018) 1–15.
- [31] Junhyuk Kim, Changhoon Lee, Deep unsupervised learning of turbulence for inflow generation at various Reynolds numbers, *J. Comput. Phys.* 406 (2020) 109216.
- [32] Jie Gui, Zhenan Sun, Yonggang Wen, Dacheng Tao, Jieping Ye, A review on generative adversarial networks: Algorithms, theory, and applications, *IEEE Trans. Knowl. Data Eng.* 35 (4) (2021) 3313–3332.
- [33] David Bau, Jun-Yan Zhu, Jonas Wulff, William Peebles, Hendrik Strobelt, Bolei Zhou, Antonio Torralba, Seeing what a gan cannot generate, in: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 4502–4511.
- [34] Nicholas Geneva, Nicholas Zabarar, Multi-fidelity generative deep learning turbulent flows, 2020, arXiv preprint [arXiv:2006.04731](https://arxiv.org/abs/2006.04731).
- [35] Luning Sun, Xu Han, Han Gao, Jian-Xun Wang, Liping Liu, Unifying predictions of deterministic and stochastic physics in mesh-reduced space with sequential flow generative model, in: *NeurIPS, PMLR*, 2023.
- [36] Yang Song, Stefano Ermon, Generative modeling by estimating gradients of the data distribution, in: *Advances in Neural Information Processing Systems*, vol. 32, 2019.
- [37] Jonathan Ho, Ajay Jain, Pieter Abbeel, Denoising diffusion probabilistic models, in: *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 6840–6851.
- [38] Prafulla Dhariwal, Alexander Nichol, Diffusion models beat gans on image synthesis, in: *Advances in Neural Information Processing Systems*, vol. 34, 2021, pp. 8780–8794.
- [39] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, Ben Poole, Score-based generative modeling through stochastic differential equations, 2020, arXiv preprint [arXiv:2011.13456](https://arxiv.org/abs/2011.13456).
- [40] Dule Shu, Zijie Li, Amir Barati Farimani, A physics-informed diffusion model for high-fidelity flow field reconstruction, *J. Comput. Phys.* 478 (2023) 111972.
- [41] Rucha Apte, Sheel Nidhan, Rishikesh Ranade, Jay Pathak, Diffusion model based data generation for partial differential equations, 2023, arXiv preprint [arXiv:2306.11075](https://arxiv.org/abs/2306.11075).
- [42] Zhong Yi Wan, Ricardo Baptista, Yi-fan Chen, John Anderson, Anudhyan Boral, Fei Sha, Leonardo Zepeda-Núñez, Debias coarsely, sample conditionally: Statistical downscaling through optimal transport and probabilistic diffusion models, 2023, arXiv preprint [arXiv:2305.15618](https://arxiv.org/abs/2305.15618).
- [43] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, David J Fleet, Video diffusion models, 2022, arXiv:2204.03458.
- [44] Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Wentao Zhang, Bin Cui, Ming-Hsuan Yang, Diffusion models: A comprehensive survey of methods and applications, *ACM Comput. Surv.* (2022).
- [45] Diederik Kingma, Tim Salimans, Ben Poole, Jonathan Ho, Variational diffusion models, in: *Advances in Neural Information Processing Systems*, vol. 34, 2021, pp. 21696–21707.
- [46] Calvin Luo, Understanding diffusion models: A unified perspective, 2022, arXiv preprint [arXiv:2208.11970](https://arxiv.org/abs/2208.11970).
- [47] Jiaming Song, Chenlin Meng, Stefano Ermon, Denoising diffusion implicit models, 2020, arXiv preprint [arXiv:2010.02502](https://arxiv.org/abs/2010.02502).
- [48] Hyungjin Chung, Jeongsol Kim, Michael T Mccann, Marc L Klasky, Jong Chul Ye, Diffusion posterior sampling for general noisy inverse problems, 2022, arXiv preprint [arXiv:2209.14687](https://arxiv.org/abs/2209.14687).
- [49] Xiang Gao, Meera Sitharam, Adrian E. Roitberg, Bounds on the Jensen gap, and implications for mean-concentrated distributions, 2017, arXiv preprint [arXiv:1712.05267](https://arxiv.org/abs/1712.05267).
- [50] John Kim, Parviz Moin, Robert Moser, Turbulence statistics in fully developed channel flow at low Reynolds number, *J. Fluid Mech.* 177 (1987) 133–166.
- [51] Stephen B. Pope, Stephen B. Pope, *Turbulent Flows*, Cambridge University Press, 2000.
- [52] Robert D. Moser, John Kim, Nagi N. Mansour, Direct numerical simulation of turbulent channel flow up to  $Re = 590$ , *Physics of fluids* 11 (4) (1999) 943–945.
- [53] Hiroyuki Abe, Hiroshi Kawamura, Yuichi Matsuo, Direct numerical simulation of a fully developed turbulent channel flow with respect to the Reynolds number dependence, *J. Fluids Eng.* 123 (2) (2001) 382–393.
- [54] Hrvoje Jasak, Aleksandar Jemcov, Zeljko Tukovic, et al., OpenFOAM: A C++ library for complex physics simulations, in: *International Workshop on Coupled Methods in Numerical Dynamics*, vol. 1000, 2007, pp. 1–20.
- [55] Chao Zhang, Lian Duan, Meelan M. Choudhari, Direct numerical simulation database for supersonic and hypersonic turbulent boundary layers, *AIAA J.* 56 (11) (2018) 4297–4311.
- [56] Lian Duan, Meelan M. Choudhari, Chao Zhang, Pressure fluctuations induced by a hypersonic turbulent boundary layer, *J. Fluid Mech.* 804 (2016) 578–607.
- [57] Brian Edward Launder, Dudley Brian Spalding, The numerical computation of turbulent flows, in: *Numerical Prediction of Flow, Heat Transfer, Turbulence and Combustion*, Elsevier, 1983, pp. 96–116.
- [58] Won-Wook Kim, Suresh Menon, A new dynamic one-equation subgrid-scale model for large eddy simulations, in: *33rd Aerospace Sciences Meeting and Exhibit*, 1995, p. 356.