# Efficient simulation and assimilation of turbulent flow using diffusion transformer

**Zeyu Li[1]; Hongkun Dou[1]; Shen Fang[1]; Wang Han[1,*]; Lijun Yang[1,*]**

[1] School of Astronautics, Beihang University, Beijing, China
[*] Corresponding authors

**Keywords: turbulent flow, diffusion models, super-resolution, data assimilation**

**Abstract.** Simulating and assimilating turbulent flows is crucial across diverse fields, from astronomy to climate research. However, traditional high-fidelity numerical methods are computationally intensive, hindering their scalability. Recent advancements have integrated generative diffusion models with posterior sampling techniques, offering promising solutions for turbulent flow assimilation. Despite their potential, these models often rely on attention-based architectures and multi-step sampling processes, which limit their efficiency for fine-grained predictions. To overcome these limitations, we introduce the Turbulence Diffusion Transformer (TiT), a diffusion-based, data-driven framework designed to efficiently simulate and assimilate turbulent flows. Drawing inspiration from latent diffusion models (LDM), TiT accelerates both training and generation by learning coarse-grained data priors through a diffusion transformer (DiT) with optimized noise scheduling. A physically aligned lifting model then decodes these coarse-grained predictions into fine-grained flow fields, ensuring adherence to the underlying turbulence dynamics. During inference, TiT refines the sampling trajectory to align with the posterior distribution, enabling compliance with sparse observations without the need for retraining. Our approach achieves efficient and flexible predictions, significantly reducing computational costs compared to traditional diffusion models while maintaining physical fidelity. In the challenging 2D Kolmogorov flow scenario, TiT demonstrates competitive accuracy (relative error < 20%) with extremely sparse observations (only 0.1% observable points), showcasing its potential for scalable and accurate turbulence modeling. The source code of TiT can be found in https://github.com/lzy12301/TiT.

## 1. Introduction

Turbulent flows are ubiquitous in nature and play a pivotal role in a wide range of scientific and engineering applications, from astrophysical phenomena [1] to climate modelling [2] and aerodynamics [3]. Accurately simulating and assimilating these flows is essential for understanding complex physical systems and making reliable predictions. However, traditional high-fidelity numerical methods are computationally expensive for high-quality assimilation of turbulent flows, particularly when scaling to real-world problems with fine-grained spatial and temporal resolutions. This computational bottleneck has spurred significant interest in developing machine learning approaches for turbulent flows [4–8].

Recent advances in generative modeling, particularly diffusion models, have shown remarkable potential for capturing complex data distributions [9–11], including those of turbulent flows [12–16]. When combined with posterior sampling techniques [17–19], these models offer a promising framework for assimilating sparse observational data into high-resolution flow fields. However, existing diffusion-based approaches often rely on attention-based architectures and multi-step sampling procedures, which suffers from quickly increasing costs when scaling to finer-grained predictions [20]. These limitations hinder their practical applicability in scenarios requiring rapid and scalable turbulence modeling.

To address these challenges, we propose the **Turbulence Diffusion Transformer (TiT)**, a novel diffusion-based framework designed to efficiently simulate and assimilate turbulent flows. TiT leverages the strengths of latent diffusion models (LDM) [21] by learning coarse-grained data priors using a diffusion transformer (DiT) [22] with modified noise scheduling [23]. A physically aligned lifting model with U-Net architecture then efficiently decodes these coarse-grained predictions into fine-grained flow fields, ensuring that the generated fields remain faithful to the underlying turbulence

dynamics. Since the diffusion model that requires complex network architecture and multi-step sampling is only used for learning the coarse-grained data samples, both training and generation can be significantly accelerated. In the inference stage, a posterior sampling strategy is particularly designed to generate dynamic flow fields according to the given observations, enabling assimilation within a large time window for different types of observation without the need for retraining.

In this work, we demonstrate the effectiveness of TiT on the challenging 2D Kolmogorov flow case [24–26], achieving competitive accuracy (relative error < 20%) with extremely sparse observations (e.g., only 0.1% observable points). TiT generates 100 frames of 256×256 flow fields within 50 diffusion steps, requiring approximately 17 seconds on a single NVIDIA RTX 3090 GPU.

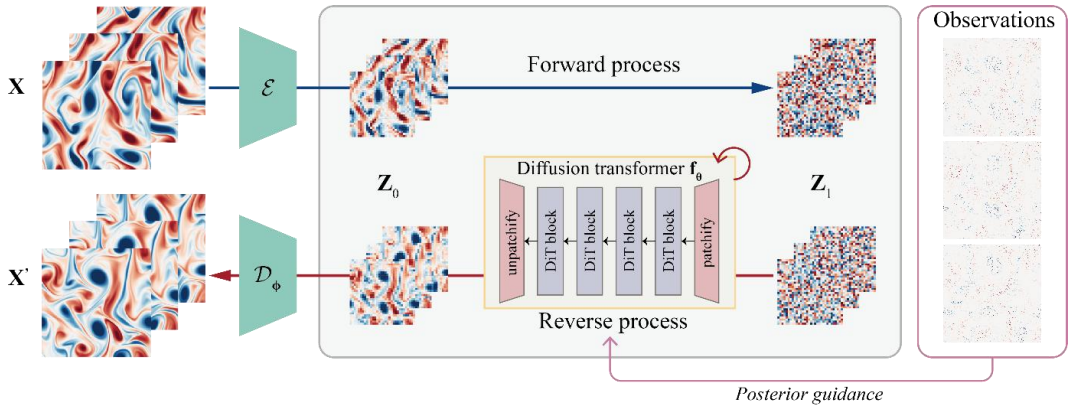


Figure 1 General workflow of the proposed method.

## 2. Preliminaries

### *2.1 Problem setups*

For the simulation and assimilation of fluid flow, we aim to recover the fine-grained fields (denoted as $\mathbf{X}$) from given observations (denoted as $\mathbf{y}$). We have

$$\mathbf{y} = \mathcal{H}(\mathbf{X}) + \boldsymbol{\epsilon} \tag{1}$$

where $\mathbf{X} = \{\mathbf{x}^t\}_{t=1}^T$ is the fine-grained flow field, which contains $T$ snapshots of the flow fields (i.e., $\mathbf{x}^1, \dots, \mathbf{x}^T$). Each snapshot (in 2D case) has shape $\mathbf{x}^t \in \mathbb{R}^{C \times n_x \times n_y}$, where $C$ is the number of field variables, and $n_x \times n_y$ is the grid size. $\mathbf{y}$ is the sparse observation, $\mathcal{H}$ is the observation operator that transform the fine-grained flow field into observations, and $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma_\epsilon \mathbf{I})$ denotes the observation noise. Our task is sampling from the posterior distribution $p(\mathbf{X}|\mathbf{y})$.

### *2.2 Diffusion models*

Diffusion models are based on the idea of gradually transforming a simple distribution (e.g., Gaussian noise) into a complex data distribution through a series of iterative steps. The transformation contains a forward process and a reverse process, written as

$$\mathrm{d}\mathbf{X}_\tau = \mathbf{f}\mathrm{d}\tau + g\mathrm{d}\mathbf{w} \tag{2}$$

$$\mathrm{d}\mathbf{X}_\tau = \left(\mathbf{f} - g^2 \nabla_{\mathbf{X}_\tau} \log p_\tau(\mathbf{X}_\tau)\right)\mathrm{d}\tau + g\mathrm{d}\mathbf{w} \tag{3}$$

where $\tau \in [0,1]$ is the diffusion time step, $\mathbf{w}$ is the standard Wiener process, and $\mathbf{f}$ and $g$ is the drifting term and diffusion coefficient, respectively, which varies across different diffusion models. For example, for VE type diffusion models [10], $\mathbf{f} = \mathbf{0}$ and $g = (\mathrm{d}\sigma_\tau^2/\mathrm{d}\tau)^{1/2}$, where $\sigma_\tau$ is the noise schedule. The forward process gradually corrupts the data sample ($\mathbf{X}_{\tau=0} \sim p(\mathbf{X})$, where $p$ is the data distribution) with Gaussian noise until the sample is fully corrupted. The reverse process, starting from a randomly sampled Gaussian noise, can generate a sample that follows the data distribution once the score

$\mathbf{s}(\mathbf{X}_\tau, \tau) = \nabla_{\mathbf{X}_\tau} \log p_\tau(\mathbf{X}_\tau)$ is approximated. Typically, we use a neural network to learn this score function by minimizing the score matching objective [10].

LDM [21], instead of operating on the original data sample, work with the latent variable $\mathbf{Z} = \mathcal{E}(\mathbf{X})$, where $\mathcal{E}$ is the encoder. In this way, each data sample is paired with a single latent variable. Compared to vanilla diffusion models, LDM learn to generate the latent variable $\mathbf{Z}$ instead of the original data sample $\mathbf{X}$. After generating a latent variable $\mathbf{Z}'$, the corresponding sample in the original data space can be acquired by $\mathbf{X}' = \mathcal{D}(\mathbf{Z}')$, where $\mathcal{D}$ is the decoder.

*2.3 Sampling of diffusion models*

Although directly solving the reverse process in equation (3) is practical, it requires a large number of discretization steps due to the numerical instability when solving a stochastic differential equation (SDE). Therefore, it is more efficient to use the following probability flow ordinary differential equation (PF-ODE) [10,23], which is proven to have the same marginal probability as the original SDE

$$\mathrm{d}\mathbf{X}_\tau = \left(\mathbf{f} - \frac{1}{2} g^2 \nabla_{\mathbf{X}_\tau} \log p_\tau(\mathbf{X}_\tau)\right) \mathrm{d}\tau \tag{5}$$

Integrating the ODE from $\tau = 1$ to $\tau = 0$ can also yield a sample that satisfies the data distribution $p(\mathbf{X})$.

## 3. Methods

*3.1 Training of TiT*

As illustrated in figure 1, TiT contains two components – the autoencoder (including both encoder and decoder) and the score network.

*3.1.1. Autoencoder.* Unlike typical LDM, the encoder of TiT (denoted as $\mathcal{E}$) is just a linear interpolator that interpolate the fine-grained flow field onto a coarse grid [27], which is regarded as the latent variable. Correspondingly, the decoder of TiT is a U-Net-based neural network (denoted as $\mathcal{D}_{\boldsymbol{\phi}}$, where $\boldsymbol{\phi}$ are the learnable parameters) that is pretrained to decode from the coarse-grained latent variable to the fine-grained space. To improve the physical compliance, we use a two-stage strategy to train the decoder [28].

*3.1.2. Score network.* The architecture of score network is based on DiT [22]. The training setups of TiT are generally the same as EDM [23]. Specifically, the objective is defined in the latent space as

$$\mathcal{J}_{\mathbf{s}}(\boldsymbol{\theta}) = \mathbb{E}_\tau \left[\lambda_\tau \cdot \mathbb{E}_{\mathbf{Z}_0} \left[\mathbb{E}_{\mathbf{Z}_\tau} [\|\mathbf{f}_{\boldsymbol{\theta}}(\mathbf{Z}_\tau, \tau) - \mathbf{Z}_0\|^2]\right]\right] \tag{8}$$

where $\mathbf{f}_{\boldsymbol{\theta}}(\mathbf{Z}_\tau, \tau) \approx \mathbf{Z}_\tau + \sigma_\tau^2 \mathbf{s}(\mathbf{X}_\tau, \tau)$ is the reparameterized score network that directly outputs the denoised data sample instead of the score function, $\mathbf{Z}_\tau = \mathbf{Z}_0 + \sigma_t \boldsymbol{\epsilon}, \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is the noised latent variable, and $\mathbf{Z}_0 = \mathcal{E}(\mathbf{X}_0), \mathbf{X}_0 \sim p(\mathbf{X})$ is the noise-free latent variable. This objective can be seen as a variant of equation (4) while substituting the original variable $\mathbf{X}$ with the latent variable $\mathbf{Z}$.

*3.2 Posterior sampling of TiT*

Given the trained TiT, our target is reconstructing the fine-grained flow field from the corresponding observations, which requires to refine the reverse process in equation (5) to generate samples that satisfy the given observation $\mathbf{y}$. To generate long temporal sequence of dynamic flow fields, we generally follow our previous work [15] that introduces sequence and observation consistency into the typical posterior sampling process, which parallelly generate pieces of overlapped sequence and concatenate them into a long sequence. In this work, we further improve this approach by three folds: 1) accelerating the process by leveraging the latent space; 2) using a mixed noise to enhance the sample quality while reducing the required sampling steps; 3) using an adaptive normalization strategy to attain robust results.

*3.2.1. Posterior sampling in latent space.* The posterior sampling injects the observable information by adding an additional likelihood term into the reverse process in equation (5), leading to the following equation [18,29,30]

$$\mathrm{d}\mathbf{Z}_\tau = \left(-\tau \cdot \nabla_{\mathbf{Z}_\tau} \log p_\tau(\mathbf{Z}_\tau) - \zeta \cdot \nabla_{\mathbf{Z}_\tau} \left\|\mathbf{y} - \mathcal{H}\left(\mathcal{D}_{\boldsymbol{\phi}}(\hat{\mathbf{Z}}_0)\right)\right\|_2^2\right) \mathrm{d}\tau \tag{10}$$

where $\hat{\mathbf{Z}}_0 = \mathbf{f_\theta}(\mathbf{Z}_\tau, \tau)$ is the one-step prediction at diffusion step $\tau$, and $\zeta$ is the guidance weight. Here, following EDM [23], we choose $\boldsymbol{f} = \mathbf{0}$ and $\sigma_\tau = \tau$ (i.e., $g = (\mathrm{d}\sigma_\tau^2/\mathrm{d}\tau)^{1/2} = \sqrt{2\tau}$).

*3.2.2. Mixed noise.* Note that $\boldsymbol{f_\theta}(\boldsymbol{Z}_\tau, \tau) \approx \boldsymbol{Z}_\tau - \tau^2 \nabla_{\boldsymbol{Z}_\tau} log\, p_\tau(\boldsymbol{Z}_\tau)$, we can further discretize equation (10) into the following form

$$\mathbf{Z}_{\tau-\Delta\tau} \approx \mathbf{f_\theta}(\mathbf{Z}_\tau, \tau) + (\tau - \Delta\tau)\frac{\mathbf{Z}_\tau - \mathbf{f_\theta}(\mathbf{Z}_\tau, \tau)}{\tau} - \zeta\Delta\tau \cdot \nabla_{\mathbf{Z}_\tau} \left\| \mathbf{y} - \mathcal{H}\left(\mathcal{D}_{\boldsymbol{\Phi}}(\hat{\mathbf{Z}}_0)\right)\right\|_2^2, \tag{11}$$

where $\tau - \Delta\tau = \sigma_{\tau-\Delta\tau}$ is exactly the noise scale at $\tau - \Delta\tau$, and $\left(\mathbf{Z}_\tau - \mathbf{f_\theta}(\mathbf{Z}_\tau, \tau)\right)/\tau$ is the predicted deterministic noise that follows the standard Gaussian distribution. In this way, this formula can be interpreted as a prediction-renoise-correction loop, gradually updating the latent variable towards the desired direction. Furthermore, since the predicted deterministic noise satisfies the standard Gaussian, it is possible to mix a real random Gaussian noise with the 'fake' Gaussian by $\left(\mathbf{Z}_\tau - \mathbf{f_\theta}(\mathbf{Z}_\tau, \tau)\right)/\tau \leftarrow \sqrt{1-\gamma^2} \cdot \left(\mathbf{Z}_\tau - \mathbf{f_\theta}(\mathbf{Z}_\tau, \tau)\right)/\tau + \gamma\boldsymbol{\epsilon}_\tau, \boldsymbol{\epsilon}_\tau \sim \boldsymbol{\mathcal{N}}(\mathbf{0}, \mathbf{I})$, where $\gamma$ is the factor that controls the strength of mixing. Empirically, such mixed noise can yield better sample quality. Here, we choose $\gamma = 0.8$ for all the numerical experiments.

*3.2.3. Adaptive normalization.* In equation (11), the quality of generation highly depends on the guidance weight $\zeta$. We adopted spherical Gaussian constraint [31] for adaptively control the guidance strength.

## 4. Experimental setups

*4.1 Data description*

We use an existing dataset for Kolmogorov flow [12], which has 40 trajectories of dynamic vorticity fields. Each of the trajectory have 320 frames with a $256 \times 256$ spatial grid. We use the first 36 trajectories as training data while the last 4 trajectories as test data.

*4.2 Task description*

Without retraining, we test our model on three different tasks: 1) reconstruction of the fine-grained fields from randomly placed sparse observations with varying sparsity (95%~99.95%); 2) super-resolution of the flow fields with varying down-sampling factors (8x~20x); 3) forecasting from initial frames.

*4.3 Baselines*

*4.3.1. Interpolation.* Trivial methods like nearest interpolation (for the first task) and bicubic interpolation (for the second task) are compared.

*4.3.2. PIDM* [12]. A diffusion-based data assimilation method for turbulent flow, which trains a diffusion model on the original grid to learn the data distribution. In the generation stage, like SDEdit [34], PIDM adds an intermediate noise to a properly initialized sample (e.g., interpolated from the observations) to maintain the information in the observations, followed then by the reverse process of diffusion model that start from this partially noised sample. In this way, it generates high-resolution samples that remain the information contained in the observation.

*4.4 Evaluation metrics*

*4.4.1. nRMSE.* This metric calculates the normalized rooted mean squared error of the vector flattened from the prediction, normalized by the L2 norm of the ground-truth vector. Lower nRMSE indicates higher accuracy.

*4.4.2. Correlation.* Correlation here refers to cosine similarity, which evaluates the statistical similarity between the predicted vector and the ground truth.

*4.4.3. Energy spectrum.* This is a statistical feature of the turbulent flow [25], showing the distribution of the kinetic energy along the spatial frequency.

## 5. Results

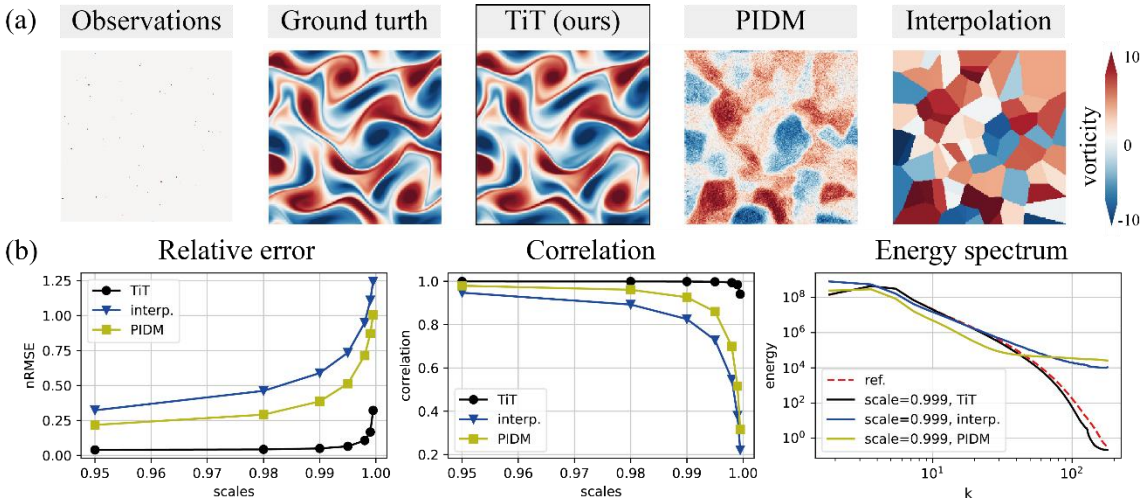


Figure 2 Reconstruction from randomly placed sparse observations. (a) Visualization of the predicted vorticity fields from the observations with 99.9% values masked. (b) Quantitative performance against the increasing of observation sparsity.

*5.1 Reconstruction from randomly placed observations*

We first evaluate the reconstruction performance of TiT on randomly placed observations (figure 2). TiT can make accurate predictions of the turbulence from extremely sparse observations that 99.9% of the values are masked (figure 2(a)). The baseline methods (i.e., nearest interpolation and PIDM), however, fail to capture the spatiotemporal patterns of the flow fields. Notably, since the sampling process of PIDM has no direct guidance for observation consistency, it suffers from significant degeneration of performance when the observations become too sparse. To further quantify the quality of predictions, we run these models with different observation sparsity (95%, 98%, 99%, 99.5%, 99.8%, 99.9% and 99.95%, respectively) and calculate corresponding metrics (figure 2(b)). The results show that the proposed approach consistently outperforms baselines with lower error and higher correlation on all tested sparsity levels. Furthermore, the energy spectrum of the TiT-generated flow fields is more consistent with the spectrum of the ground truth data as seen in the right panel of figure 2(b).

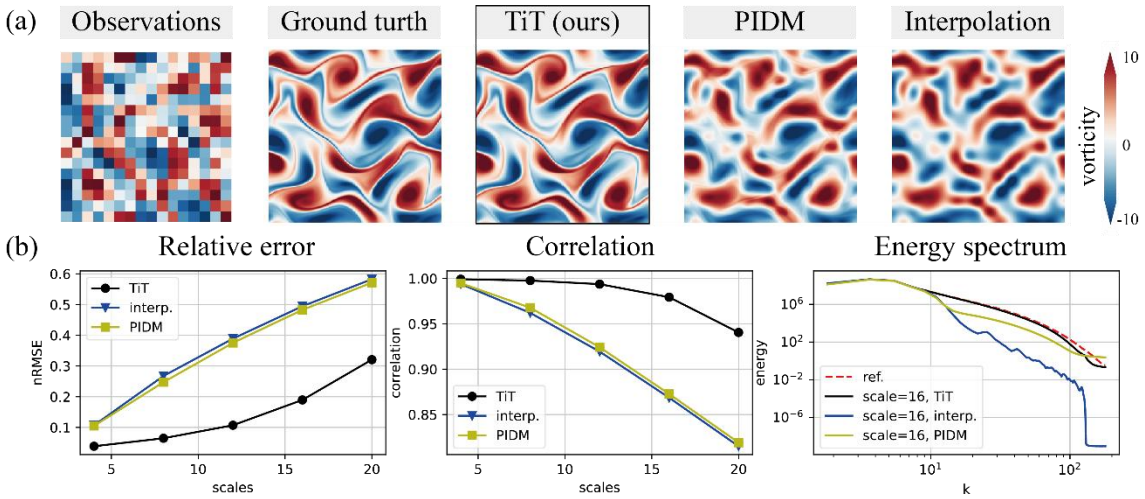


Figure 3 Super-resolution of flow fields. (a) Visualization of the predicted vorticity fields for 16x super-resolution. (b) Quantitative performance against the increasing of down-sampling factor.

*5.2 Super-resolution of flow field*

We also validate our model on super-resolution task (figure 3), in which the observation is regularly down-sampled from the fine-grained data with varying down-sampling factors (4x, 8x, 12x, 16x and

20x super-resolution, respectively). As presented in figure 3(a), even the resolution of the observation is 16x coarser than the fine-grained data, TiT can still accurately recover the fine-grained turbulence, while the baselines fail to reconstruct the spatially high-frequency part of the flow fields, which can be further illustrated by the energy spectrum (see the right panel of figure 3(b)). Quantitatively, we compare the nRMSE and correlation under different down-sampling factors, demonstrating the advancement of the proposed method in handling super-resolution task (figure 3(b)).

*5.3 Forecasting*

Forecasting future states of a chaotic system is another challenging task that can show the numerical stability of the pretrained model as well as the ability in capturing complex physics of the system. Given the first two frames of vorticity field, TiT can make stable predictions of their future states, capturing the dynamics within 100 steps (figure 4). Due to the strongly chaotic nature of the system, the accuracy of predictions can diverge after a period (figure 4 (b)). Nevertheless, the statistics can remain consistent for long term forecasting (see right panel of figure 4 (b)).

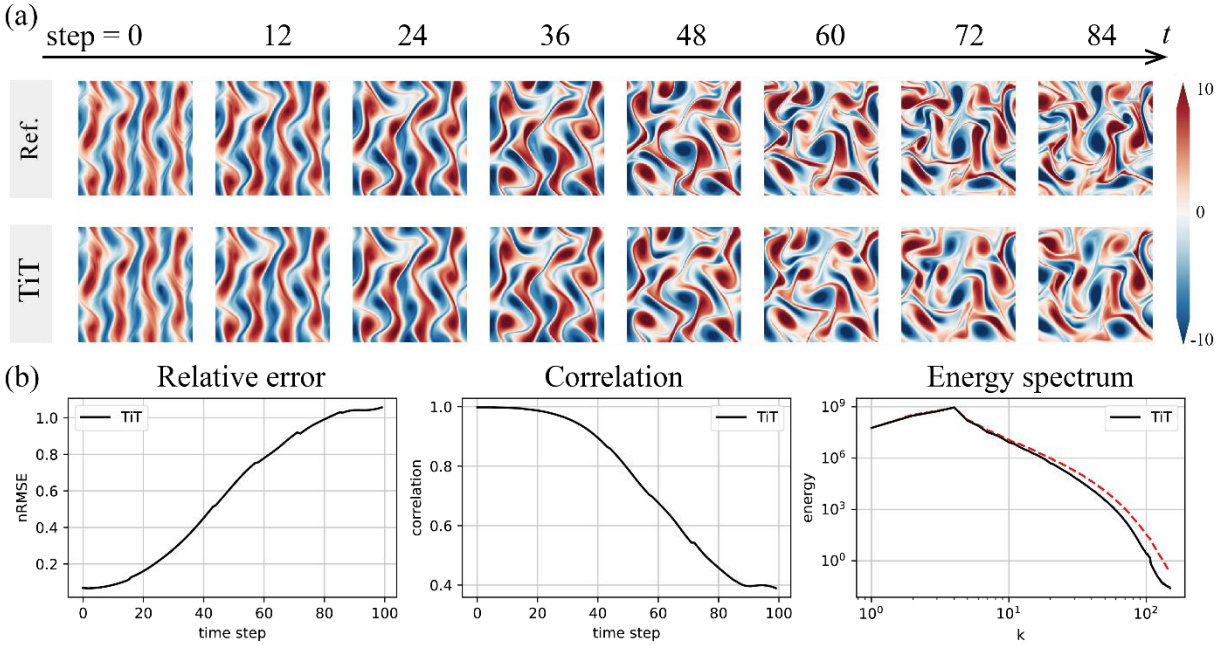


Figure 4 Results for flow field forecasting using TiT. (a) Visualization of the forecasted vorticity fields (the first row shows the ground truth, which is denoted as Ref.). (b) Error accumulation and statistics of the predicted flow fields.

## 6. Conclusion

In this work, based on diffusion probabilistic models, we propose a computational framework for the simulation and assimilation of turbulent flow, namely, TiT. By incorporating the latent modeling approach and properly designed posterior sampling strategy, TiT framework can facilitate efficient and accurate generation from various types of sparse observations. The evaluation on three different tasks presents the flexibility of TiT, showing the advantageous performance in handling extremely sparse observations compared to the diffusion-based baseline. Though TiT is only tested on the Kolmogorov flow case, this general framework can be easily transferred to other complex dynamical systems, providing a solution for broader applications involving fluid systems.

## 7. References

[1] M. Heimpel, J. Aurnou, Turbulent convection in rapidly rotating spherical shells: A model for equatorial and high latitude jets on Jupiter and Saturn, Icarus 187 (2007). https://doi.org/10.1016/j.icarus.2006.10.023.
[2] T. Schneider, J. Teixeira, C.S. Bretherton, F. Brient, K.G. Pressel, C. Schär, A.P. Siebesma, Climate goals and computing the future of clouds, Nat. Clim. Chang. 7 (2017). https://doi.org/10.1038/nclimate3190.
[3] J.G. Marvin, Turbulence modeling for computational aerodynamics, AIAA J. 21 (1983). https://doi.org/10.2514/3.8182.

[4] S. Cheng, C. Quilodran-Casas, S. Ouala, A. Farchi, C. Liu, P. Tandeo, R. Fablet, D. Lucor, B. Iooss, J. Brajard, D. Xiao, T. Janjic, W. Ding, Y. Guo, A. Carrassi, M. Bocquet, R. Arcucci, Machine learning with data assimilation and uncertainty quantification for dynamical systems: a review, IEEE/CAA J. Autom. Sin. 10 (2023) 1361–1387. http://arxiv.org/abs/2303.10462.
[5] G.E. Karniadakis, I.G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, L. Yang, Physics-informed machine learning, Nat. Rev. Phys. 3 (2021) 422–440. https://doi.org/10.1038/s42254-021-00314-5.
[6] M. Buzzicotti, Data reconstruction for complex flows using AI: Recent progress, obstacles, and perspectives, Europhys. Lett. 142 (2023) 23001. https://doi.org/10.1209/0295-5075/acc88c.
[7] R. Vinuesa, S.L. Brunton, Enhancing computational fluid dynamics with machine learning, Nat. Comput. Sci. 2 (2022) 358–366. https://doi.org/10.1038/s43588-022-00264-7.
[8] K. Fukami, K. Fukagata, K. Taira, Super-resolution analysis via machine learning: a survey for fluid flows, Theor. Comput. Fluid Dyn. 37 (2023) 421–444. https://doi.org/10.1007/s00162-023-00663-0.
[9] J. Ho, A. Jain, P. Abbeel, Denoising diffusion probabilistic models, in: Adv. Neural Inf. Process. Syst., 2020: pp. 6840–6851.
[10] Y. Song, J. Sohl-Dickstein, D.P. Kingma, A. Kumar, S. Ermon, B. Poole, Score-Based Generative Modeling Through Stochastic Differential Equations, in: Proc. 9th Int. Conf. Learn. Represent., 2021.
[11] Y. Song, S. Ermon, Generative Modeling by Estimating Gradients of the Data Distribution, in: Adv. Neural Inf. Process. Syst., 2019.
[12] D. Shu, Z. Li, A. Barati, A physics-informed diffusion model for high-fidelity flow field reconstruction, J. Comput. Phys. 478 (2023) 111972. https://doi.org/10.1016/j.jcp.2023.111972.
[13] S.R. Cachay, B. Zhao, H. Joren, R. Yu, DYffusion: A Dynamics-informed Diffusion Model for Spatiotemporal Forecasting, in: Adv. Neural Inf. Process. Syst., 2023. http://arxiv.org/abs/2306.01984.
[14] F. Rozet, G. Louppe, Score-based Data Assimilation, in: Adv. Neural Inf. Process. Syst., 2023. http://arxiv.org/abs/2306.10574.
[15] Z. Li, W. Han, Y. Zhang, Q. Fu, J. Li, L. Qin, R. Dong, H. Sun, Y. Deng, L. Yang, Learning spatiotemporal dynamics with a pretrained generative model, Nat. Mach. Intell. 6 (2024) 1566–1579. https://doi.org/10.1038/s42256-024-00938-z.
[16] T. Li, L. Biferale, F. Bonaccorso, M.A. Scarpolini, M. Buzzicotti, Synthetic Lagrangian turbulence by generative diffusion models, Nat. Mach. Intell. 6 (2024) 393–403. https://doi.org/10.1038/s42256-024-00810-0.
[17] A. Lugmayr, M. Danelljan, A. Romero, F. Yu, R. Timofte, L. Van Gool, RePaint: Inpainting using Denoising Diffusion Probabilistic Models, in: IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2022: pp. 11451–11461. https://doi.org/10.1109/CVPR52688.2022.01117.
[18] H. Chung, J. Kim, M.T. Mccann, M.L. Klasky, J.C. Ye, Diffusion Posterior Sampling for General Noisy Inverse Problems, in: Proc. 11th Int. Conf. Learn. Represent., 2023. http://arxiv.org/abs/2209.14687.
[19] V.I.A.T.O.O. Rdering, P. Sanchez, Pseudoinverse-Guided Diffusion Models for Inverse Problems, in: Proc. 11th Int. Conf. Learn. Represent., 2023.
[20] L. Yang, Z. Zhang, Y. Song, S. Hong, R. Xu, Y. Zhao, W. Zhang, B. Cui, M.H. Yang, Diffusion Models: A Comprehensive Survey of Methods and Applications, ACM Comput. Surv. 56 (2023). https://doi.org/10.1145/3626235.
[21] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, High-Resolution Image Synthesis with Latent Diffusion Models, in: Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., n.d.: pp. 10684–10695.
[22] W. Peebles, S. Xie, Scalable Diffusion Models with Transformers, Proc. IEEE Int. Conf. Comput. Vis. (2023) 4172–4182. https://doi.org/10.1109/ICCV51070.2023.00387.
[23] T. Karras, T. Aila, Elucidating the Design Space of Diffusion-Based Generative Models, in: Adv. Neural Inf. Process. Syst., 2022.
[24] G.J. Chandler, R.R. Kerswell, Invariant recurrent solutions embedded in a turbulent two-dimensional Kolmogorov flow, J. Fluid Mech. 722 (2013) 554–595. https://doi.org/10.1017/jfm.2013.122.
[25] G. Boffetta, R.E. Ecke, Two-dimensional turbulence, Annu. Rev. Fluid Mech. 44 (2011) 427–451. https://doi.org/10.1146/annurev-fluid-120710-101240.
[26] A.N. Kolmogorov, The local structure of turbulence in incompressible viscous fluid for very large Reynolds numbers, Proc. R. Soc. London. Ser. A Math. Phys. Sci. 434 (1991). https://doi.org/10.1098/rspa.1991.0075.
[27] H. Gao, S. Kaltenbach, P. Koumoutsakos, Generative Learning for Forecasting the Dynamics of Complex Systems, Nat. Commun. (2024) 8904. https://doi.org/10.1038/s41467-024-53165-w.
[28] Z. Li, H. Dou, S. Fang, W. Han, Y. Deng, L. Yang, Physics-aligned field reconstruction with diffusion bridge, in: Proc. 13th Int. Conf. Learn. Represent., 2025.
[29] B. Song, S.M. Kwon, Z. Zhang, X. Hu, Q. Qu, L. Shen, Solving Inverse Problems With Latent Diffusion Models Via Hard Data Consistency, 12th Int. Conf. Learn. Represent. ICLR 2024 1 (2024) 1–38.
[30] C. Caramanis, A.G. Dimakis, S. Shakkottai, Solving Linear Inverse Problems Provably via Posterior Sampling with Latent Diffusion Models, in: Adv. Neural Inf. Process. Syst., 2023. https://huggingface.co/spaces/PSLD/PSLD.
[31] L. Yang, S. Ding, Y. Cai, J. Yu, J. Wang, Y. Shi, Guidance with Spherical Gaussian Constraint for Conditional Diffusion, Proc. Mach. Learn. Res. 235 (2024) 56071–56095.
[32] C. Meng, Y. He, Y. Song, J. Song, J. Wu, J.Y. Zhu, S. Ermon, Sdedit: Guided Image Synthesis and Editing With Stochastic Differential Equations, in: Proc. 10th Int. Conf. Learn. Represent., 2022.