Multi-Order Loss Functions For Accelerating Unsteady Flow Simulations with Physics-Based AI

1st Wei Xian Lim
Rolls-Royce@NTU Corporate Lab
Nanyang Technological University (NTUsg)
Singapore
weixian001@e.ntu.edu.sg

2nd Naheed Anjum Arafat
Rolls-Royce@NTU Corporate Lab
Nanyang Technological University (NTUsg)
Singapore
naheed_anjum@u.nus.edu

3rd Wai Lee Chan*
School of Mechanical and Aerospace Engineering (MAE)
Nanyang Technological University (NTUsg)
Singapore
chan.wl@ntu.edu.sg

4th Wai-Kin Adams Kong**
College of Computing and Data Science (CCDS)
Nanyang Technological University (NTUsg)
Singapore
AdamsKong@ntu.edu.sg

Abstract*—Studies show that artificial intelligence (AI) with embedded physics solvers has improved the accuracy of predictions on various physics problems, especially those associated with fluid dynamics. The crucial element in optimizing weight training for estimating flow fields within the AI network lies in the choice of the loss function. In addressing regression-type problems, particularly those involving the temporal evolution of flow fields, the mean square error (MSE) loss function is commonly employed at the current and single time step. However, an issue arises in existing methodologies that utilize MSE-based loss functions with single-time step information for predicting unsteady flow. Most of these approaches overlook the significance of incorporating the temporal history of the flow, a factor that cannot be disregarded in the context of numerical solvers. Hence, in this work, a physics-based AI (PbAI) method with higher-order loss functions is applied to unsteady scenarios, in particular to two distinct turbulent flows where a multitude of fine structures is present, namely, forced and decaying turbulence. Direct numerical simulations on uniform Cartesian grids are conducted to simulate these scenarios, generating two distinct datasets for training and inference. Each dataset comprises 32 randomly initialized conditions spanning $4,848$ time steps for each turbulent flow type. Five distinct models are devised, incorporating features such as rollouts from coarse numerical solvers and temporal considerations in the loss function calculation. The constructed PbAI models demonstrate consistent improvements in predictive performance over the entire temporal domain. These findings are further corroborated through vorticity correlation analyses. The empirical result demonstrates that the accuracy of the baseline case improves by up to 48% and 30% for forced and decaying turbulence, respectively. These results significantly underscore the importance of the temporal histories of flow in the loss function in enhancing predictive capabilities for complex and unsteady turbulent flows.

***Index Terms*—loss functions, unsteady simulations, turbulence, physics-based AI**

I. Introduction

Deep learning has recently gained attention for its potential in predicting physical systems [1], thereby offering an alternative method in the field of computational fluid dynamics (CFD). The fluid flows and dynamics are governed by the Navier Stokes (NS) equations, a set of partial differential equations (PDEs), that can be solved directly using direct numerical simulation (DNS) [2], large-eddy simulation (LES) [3], or Reynolds-averaged Navier Stokes (RANS) [4], ordered in decreasing computational demand. Fine grids and small time steps are often required for obtaining high-fidelity solutions, which incur enormous computational costs and the simulation may become intractable. Thus, deep learning becomes an alternative tool to reduce computational resources while maintaining accuracy.

Most works that leverage different types of deep learning methods in various physical systems, such as fluid dynamics [5, 6, 7], solid mechanics [8, 9], chemical mixing [10], combustion [11], and weather forecasting [12], focus on accelerating the simulation process through super-resolution [5], improving LES or RANS modeling [6, 7, 13, 14, 15], training the closure models [11, 16], and embedding physical laws into neural networks [7, 17, 18, 19].

Many works have shown the capability of deep learning in reducing the conventional simulation cost. Reference [5] studies the correlation between high (DNS or LES) and low-resolution simulations and showcases the capability of convolutional neural networks (CNN) in achieving excellent accuracy with coarser mesh while promising substantial computational cost reduction. Their work, however, requires structured grids due to the use of CNN and large dataset inputs (high-resolution simulation). Another work [6] embeds the trained CNN in RANS simulation to accelerate the convergence speed. This method shows promising results but is limited to steady flow problems, while most engineering applications involve unsteady flows. References [11, 15, 16] have used supervised learning with high-fidelity datasets to train models to represent the unresolved small scales for turbulence closure.

The selection of an appropriate loss function plays a piv-

*Corresponding author for CFD. **Corresponding author for AI.

979-8-3503-5409-6/24/$31.00 ©2024 IEEE
DOI 10.1109/CAI59869.2024.00173

otal role in training neural networks to accurately model complex relationships within data. Loss functions quantify the disparity between predicted outputs and actual ground truth labels, thereby guiding the optimization process during model training. One commonly employed loss function is the mean squared error (MSE), which calculates the average squared difference between predicted and actual values. This loss is often suitable for regression tasks. For classification problems, cross-entropy loss [20], including binary and categorical variants, is widely used. It evaluates the dissimilarity between predicted probability distributions and actual class distributions. The hinge loss [21], employed in support vector machines and popularized in the context of deep learning with linear support vector machines, is commonly applied to binary classification tasks. Additionally, specialized loss functions such as Huber loss [22] and the Kullback-Leibler divergence [23] serve specific purposes, addressing issues like robustness to outliers and probability distribution divergence, respectively.

The choice of a loss function depends on the specific characteristics of the task, the desired model behavior, and the nature of the data. In the context of fluid dynamics, MSE-based loss functions [5, 9, 8, 7] are commonly employed regardless of the flow type or regime. The MSE-based loss function, owing to its summation of pointwise errors, adeptly captures spatial discrepancies within flow fields. However, the limitation of MSE-based loss functions becomes evident in their failure to adequately encapsulate temporal changes when exclusively minimizing current time-step flow fields [5, 8]. To address this shortfall, novel loss functions incorporating information from past and/or future flow fields have been proposed. The integration of temporal context enables the network to discern temporal evolution trends, thereby enhancing overall predictive performance. Five MSE-based loss functions are proposed to use either coarse field output from a numerical solver or fields from different time steps based on a backward or central difference scheme in order to investigate the influence of different loss functions on predicting unsteady flow type simulations.

The rest of the paper is organized as follows: Section II describes the dataset preparation for training and testing. Section III proposes five different loss functions. Section IV reports the experimental results and comparisons. Finally, Section V draws conclusions and discusses limitations.

II. Simulation Parameters and Datasets

In this work, a comprehensive numerical solver, *JAX-CFD* [5], is employed to facilitate data generation and as the initial step in constructing the machine learning models. The solver incorporates a staggered square mesh structure within a finite volume framework. The computational domain is discretized into distinct computational cells, where the velocity field is assigned to the edges, while the pressure is determined at the cell centers. Rather than utilizing a spectral method, a real-space formulation of the NS equations is solved for its superior adaptability in accommodating boundary conditions.

The incompressible fluid (i.e., density, ρ, is constant) is governed by the NS equation and takes the following form,

$$\nabla \cdot \mathbf{U} = 0 \ , \tag{1a}$$

$$\frac{\partial \mathbf{U}}{\partial t} = -(\mathbf{U} \cdot \nabla)\mathbf{U} + \nu \nabla^2 \mathbf{U} - \frac{1}{\rho}\nabla p + \mathbf{f} \ , \tag{1b}$$

where $\mathbf{U}$, p, ν, and $\mathbf{f}$ represent the velocity vector, pressure, kinematic viscosity of the fluid, and external forcing vector term, respectively. The diffusion term, denoted as the second term on the right-hand side of (1b), is treated implicitly, utilizing a second-order central difference scheme to discretize the Laplace operator. Simultaneously, the convection term, represented by the first term on the right-hand side of (1b), is solved by advecting all velocity components utilizing a high-order scheme based on the Van-Leer flux limiter [24]. Additionally, the Poisson equation is solved at each time step to determine the pressure field. To achieve this, a fast diagonalization approach is employed, utilizing explicit matrix multiplication [25].

To replicate the statistical properties of fully developed turbulent flows, Kolmogorov force [26] is introduced as an external force term in (1b). The Kolmogorov force is defined as follows,

$$\mathbf{f} = (f_x \ , \ f_y) = (K \sin(ky) \ , \ 0) \ , \tag{2}$$

where K represents the Kolmogorov scale, and k denotes the forcing wavenumber. It is important to note that an additional linear forcing proportional to the velocity is incorporated, introducing a negative coefficient to prevent energy from accumulating solely in large vortices. This forcing term exhibits statistical homogeneity and isotropy while adhering to a power-law distribution in the wavenumber space. The incorporation of Kolmogorov forcing within numerical simulations enables the study of turbulence and its statistical behaviors under controlled conditions, where parameters such as the magnitude K and the wavenumber k can be adjusted accordingly.

Two types of Kolmogorov flow simulations are generated for datasets, namely forced and decaying turbulence, specifically at a Reynolds number (Re) of 1000, within a two-dimensional domain size of $[2\pi \times 2\pi]$. The process of generating the datasets encompasses three distinct steps: (i) initiating a burn-in simulation from a random initial condition; (ii) conducting a simulation for a predetermined duration using a high-resolution solver; and (iii) downsampling the obtained solution to a lower-resolution grid for subsequent training and testing of the model.

During the burn-in stage, the initial condition is defined by several key parameters, namely the *random seed number*, *resolution*, and *initial peak wavenumber*. The *initial peak wavenumber* corresponds to the peak wavenumber of the log-normal distribution utilized for sampling random initial conditions. To ensure the reliability of the subsequent analysis, the initial transient results are disregarded, and the duration of this burn-in phase is determined by the maximum amplitude of

TABLE I
PARAMETERS USED IN THE DATASET GENERATION.

Types of Turbulence	Forced		Decaying	
Datasets	Train	Test	Train	Test
initial peak wavenumber	4			
number of initial conditions	32			
burn-in time	40		4.5	
simulation time [s]	34			
simulation resolution	2048×2048			
save resolution	64×64			
maximum velocity [m/s]	7			
viscosity [kg/m/s]	0.001			
Kolmogorov scale	1		N.A	
forcing wavenumber	4		N.A	
linear coefficient	-0.1		N.A	

the initial velocity field, referred to as the *burn-in time*. Subsequently, the velocity field results obtained at the *simulation resolution* are downsampled to the *save resolution*, forming the dataset. The same procedures are applied to generate the decaying turbulence dataset by treating the external force term as zero. The specific values assigned to these aforementioned parameters for training and testing datasets are briefly summarized in Tab. I.

III. Convolution Neural Network (CNN) with Multi-order Loss Functions

The machine learning (ML) model employed in this work is based on the fully convolutional architectures proposed by [5]. The schematic representation of the network, as depicted in Fig. 1, illustrates the integration of CNN with a numerical solver to advance the state of the flow field. In [5], the mean square error (MSE) between the predicted field, $\hat{\mathbf{u}}$ and ground truth, $\mathbf{u}^{GT}$, is utilized as the loss function during training. In this work, the loss function is tailored to encompass not only the present time step but also to incorporate information from preceding and/or subsequent time steps. This modification is envisaged to enable the network to capture temporal variations in velocity and ultimately enhance the overall predictive capacity.

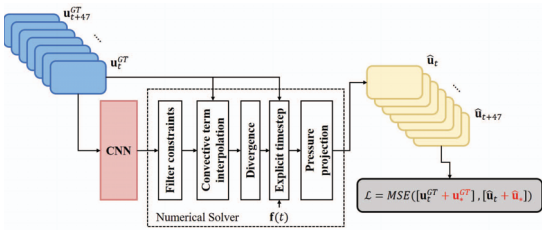

Fig. 1. The pipeline of a convolutional neural network controlling learned approximations inside the convection calculation of a standard numerical solver.

Considering MSE as the main measurement in the loss function, $\mathcal{L}$ as stated in (3), as such,

$$\mathcal{L} = MSE([\mathbf{u}_t^{GT} + \mathbf{u}_*^{GT}], [\hat{\mathbf{u}}_t + \hat{\mathbf{u}}_*]) \,, \tag{3}$$

where $\mathbf{u}_*^{GT}$ and $\hat{\mathbf{u}}_*$ are defined accordingly as follow,

- Baseline Method [5], 'baseline':

$$\mathbf{u}_*^{GT} = 0 \quad ; \quad \hat{\mathbf{u}}_* = 0$$

- Pure numerical solver, 'coarse' (Physics-driven loss function):

$$\mathbf{u}_*^{GT} = \mathbf{u}_t^{GT} \quad ; \quad \hat{\mathbf{u}}_* = \mathbf{u}_t^{NS}$$

- Zeroth order backward, 'zeroOrder':

$$\begin{aligned} \mathbf{u}_*^{GT} &= \mathbf{u}_{t-1}^{GT} \quad \text{for} \quad t = 0, \mathbf{u}_*^{GT} = \mathbf{u}_0^{GT} \\ \hat{\mathbf{u}}_* &= \hat{\mathbf{u}}_{t-1} \quad \text{for} \quad t = 0, \hat{\mathbf{u}}_* = \hat{\mathbf{u}}_0 \end{aligned}$$

- 1st order backward, '1stOrder':

$$\begin{aligned} \mathbf{u}_*^{GT} &= \mathbf{u}_{t-1}^{GT} + (\mathbf{u}_{t-1}^{GT} - \mathbf{u}_{t-2}^{GT}) \,, \\ &\quad \text{for} \quad t = [0,1], \mathbf{u}_*^{GT} = [\mathbf{u}_0^{GT}, \mathbf{u}_1^{GT}] \\ \hat{\mathbf{u}}_* &= \hat{\mathbf{u}}_{t-1} + (\hat{\mathbf{u}}_{t-1} - \hat{\mathbf{u}}_{t-2}) \,, \\ &\quad \text{for} \quad t = [0,1], \hat{\mathbf{u}}_* = [\hat{\mathbf{u}}_0, \hat{\mathbf{u}}_1] \end{aligned}$$

- 2nd order backward, '2ndOrder':

$$\begin{aligned} \mathbf{u}_*^{GT} &= \mathbf{u}_{t-1}^{GT} + (\mathbf{u}_{t-1}^{GT} - \mathbf{u}_{t-2}^{GT}) \\ &\quad + \frac{1}{2}(\mathbf{u}_{t-1}^{GT} - 2\mathbf{u}_{t-2}^{GT} + \mathbf{u}_{t-3}^{GT}) \,, \\ &\quad \text{for} \quad t = [0,1,2], \mathbf{u}_*^{GT} = [\mathbf{u}_0^{GT}, \mathbf{u}_1^{GT}, \mathbf{u}_2^{GT}] \\ \hat{\mathbf{u}}_* &= \hat{\mathbf{u}}_{t-1} + (\hat{\mathbf{u}}_{t-1} - \hat{\mathbf{u}}_{t-2}) \\ &\quad + \frac{1}{2}(\hat{\mathbf{u}}_{t-1} - 2\hat{\mathbf{u}}_{t-2} + \hat{\mathbf{u}}_{t-3}) \,, \\ &\quad \text{for} \quad t = [0,1,2], \hat{\mathbf{u}}_* = [\hat{\mathbf{u}}_0, \hat{\mathbf{u}}_1, \hat{\mathbf{u}}_2] \end{aligned}$$

- Central difference, 'centerDiff':

$$\begin{aligned} \mathbf{u}_*^{GT} &= \mathbf{u}_{t-1}^{GT} + \frac{1}{2}(\mathbf{u}_t^{GT} - \mathbf{u}_{t-2}^{GT}) \,, \\ &\quad \text{for} \quad t = [0,1], \mathbf{u}_*^{GT} = [\mathbf{u}_0^{GT}, \mathbf{u}_1^{GT}] \\ \hat{\mathbf{u}}_* &= \hat{\mathbf{u}}_{t-1} + \frac{1}{2}(\hat{\mathbf{u}}_t - \hat{\mathbf{u}}_{t-2}) \,, \\ &\quad \text{for} \quad t = [0,1], \hat{\mathbf{u}}_* = [\hat{\mathbf{u}}_0, \hat{\mathbf{u}}_1] \end{aligned}$$

The supplementary terms ($\mathbf{u}_*$) introduced in (3) can be acquired through various methodologies. Three primary approaches are considered: (i) 'coarse', which is derived from the output of the pure numerical solver ($\mathbf{u}_t^{NS}$) at coarse grid resolution (64×64), where $\mathbf{u}_t^{NS}$ is the 1-time step roll-out from pure numerical solver that is fed with previous time step $\hat{u}_{t-1}$ as input; (ii) multi-order 'backward', which is obtained by incorporating information from one or more preceding time steps; and (iii) 'centerDiff', which is computed by employing central differencing along the temporal dimension. Additionally, the CNN is configured with hyperparameters specifying 11 hidden layers, 64 hidden channels, 2 output channels, and 5 kernel sizes across all models. The optimization is performed using the Adam optimizer with a learning rate of 10^{-3}, $b1 = 0.9$, and $b2 = 0.98$.

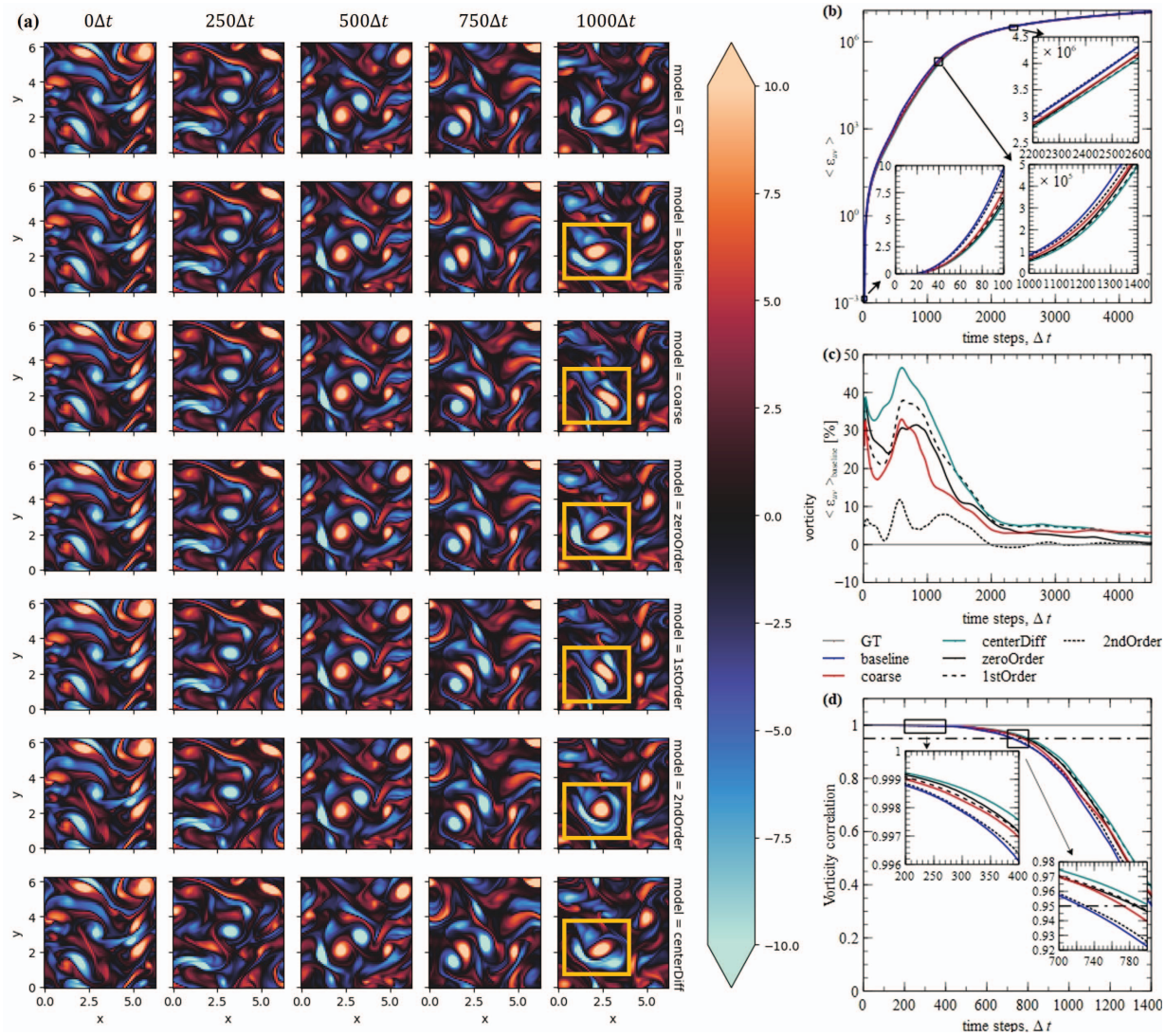

Fig. 2. Forced turbulence: (a) Comparison of vorticity contours of all models. (b) Ensemble average of L2 norm of velocity error distribution over 32 initial conditions against time steps for different models. (c) Ensemble average of L2 norm of velocity error relative to the baseline model. (d) Comparison of the ensemble average of vorticity correlation over 32 different initial conditions for each model.

IV. Results and Discussions

Vorticity is a fundamental concept in the field of fluid dynamics, serving as a descriptor for the local rotation or swirling motion exhibited by fluid particles within a given flow field. As a vector quantity, it is defined as the curl of the velocity vector field, mathematically represented as

$$\boldsymbol{\omega} = \nabla \times \mathbf{U} \, . \tag{4}$$

The vorticity vector is orthogonal to the plane of rotation, with its magnitude denoting the local rate of rotation, while its sign indicates the direction of rotation. Regions characterized by high vorticity signify pronounced rotational motion, often observed in vortex cores or regions exhibiting concentrated turbulence. Conversely, regions of low vorticity correspond to relatively smooth and non-rotating flow regions.

Fig. 2(a) presents a comparative analysis of vorticity field evolution among five models, including the baseline model proposed by Kochkov et al. [5] and ground truth at a specific sample. Each successive column in the snapshot sequence corresponds to a temporal separation of 250 time steps, while each row represents the progress of the turbulent flow under distinct models. The yellow boxes in the last column emphasize the differences in vortical structures between the five models, the baseline, and the ground truth. Both ‘coarse’ and

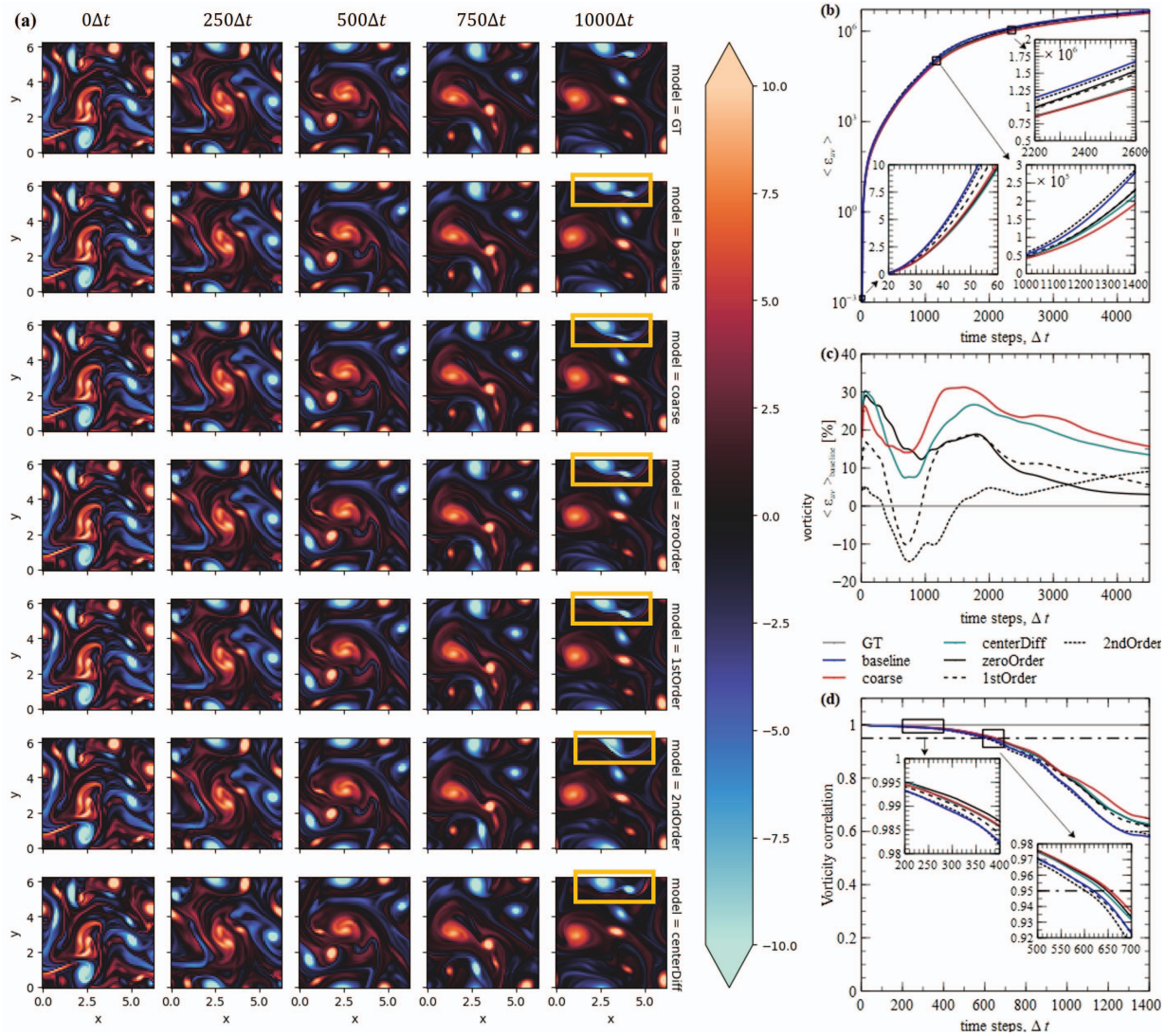

Fig. 3. Decaying turbulence: (a) Comparison of vorticity contour of all models. (b) Ensemble average of L2 norm of velocity error distribution over 32 initial conditions against time steps for different models. (c) Ensemble average of L2 norm of velocity error relative to the baseline model. (d) Comparison of the ensemble average of vorticity correlation over 32 different initial conditions for each different model.

'centerDiff' models exhibit vortices similar to those observed in the baseline and ground truth. This agreement is further corroborated by the ensemble average of the L2 norm of velocity error, presented in Fig. 2(b), defined as,

$$< \epsilon_{uv} >= \frac{1}{N} \sum_{i=1}^{N} ||\hat{\mathbf{u}}_i - \mathbf{u}_i^{GT}||^2 \,, \tag{5}$$

where N is the total number of initial conditions, $\hat{\mathbf{u}}$ and $\mathbf{u}^{GT}$ are the predicted and ground truth velocity fields, respectively, as a specific time instant. A series of subplots are presented at three distinct time instants, indicating that both 'zeroOrder' and 'centerDiff' models outperform the baseline performance at all time points, achieving a maximum reduction in error of 48% compared to baseline as illustrated in Fig. 2(c).

In another perspective, the ensemble average vorticity correlation with respect to the ground truth is plotted in Fig. 2(d), indicating that all models perform better than the baseline, especially the 'centerDiff' model which has the best performance over a longer period, approximately 80 time steps more provided the threshold criterion is set at a correlation of 0.95.

Fig. 3(a) depicts the temporal evolution of the decaying vorticity field originating from a specific initial condition, as generated by the decaying turbulence dataset and processed through various loss function models within the neural net-

TABLE II
SUMMARY OF TRAINING TIME, GPU MEMORY, AND RAM USAGES OF ALL MODELS USED IN BOTH FORCED AND DECAYING TURBULENCE DATASETS.

	Models	Tot. training time [hr]	Peak GPU memory usage during training [GB]	Peak RAM usage during training [GB]
Forced	baseline	3.81	6.66	10.5
	coarse	3.62	11.67	15.8
	zeroOrder	3.30	6.66	10.5
	1stOrder	3.29		
	2ndOrder	3.32		
	centerDiff	3.46		10.9
Decaying	baseline	3.78	6.66	10.5
	coarse	4.41	11.67	15.9
	zeroOrder	3.34	6.66	10.5
	1stOrder	3.32		
	2ndOrder	3.28		
	centerDiff	3.46		10.8

TABLE III
COMPARISON OF INFERENCE AND SIMULATION TIMES OF ALL MODELS FOR BOTH FORCED AND DECAYING TURBULENCE DATASETS.

	Models	Total inference time [s]	Total simulation time [s]
Forced	baseline	30	3654
	coarse	107	
	zeroOrder	53	
	1stOrder	24	
	2ndOrder	44	
	centerDiff	49	
Decaying	baseline	24	2159
	coarse	77	
	zeroOrder	20	
	1stOrder	24	
	2ndOrder	29	
	centerDiff	39	

work framework. The highlighted yellow boxes reveal both the 'coarse' (third row), and 'centerDiff' (last row) models demonstrate compatibility with the ground truth and even outperform the baseline model.

This observation is further substantiated quantitatively by the ensemble average of ϵ_{uv}, which is computed for 32 initial conditions based on (5) and shown in Fig. 3(b) for all models. As corroborated by Fig. 3(c), the 'coarse', 'centerDiff', and 'zeroOrder' models yield better inference results relative to the baseline. Similarly, the vorticity correlation plots in Fig. 3(d) unambiguously demonstrate that all models, except the '2ndOrder' model, exhibit enhanced long-term predictability compared to the baseline, particularly the 'coarse' and 'centerDiff' models. Overall, the 'centerDiff' model offers a stable and more reliable predictive capability across two distinct turbulent flows.

Tab. II provides a concise summary of the total training time required for 32 distinct initial conditions for each model, along with the corresponding peak GPU memory and RAM utilization during the training process. Tab. III offers a comparative assessment of the time necessary for each model to generate a total of 4848 time steps of flow fields for 32 diverse initial conditions, in contrast to the time taken by the pure numerical simulation and baseline model by [5]. Evidently, the network has expedited the simulation of unsteady turbulent flows by a minimum of two orders of magnitude.

V. CONCLUSION

In this work, we propose incorporating preceding and future time steps into the loss function to improve the performance of turbulent flow prediction. The loss function employed by the baseline only considers the current time step. During longer roll-outs, the model only retains information from the most recent time step, neglecting data from previous and future time steps. By incorporating preceding and future time steps into the loss function, the model is compelled to jointly optimize for the current and earlier time steps. This approach enables the model to remember the temporal evolution, thereby learning better.

While this work highlights the potential and feasibility of employing neural networks with higher-order loss functions to advance the simulation of unsteady fluid flows at reduced computational costs, while maintaining a reasonable degree of accuracy, it is pertinent to acknowledge certain limitations. Notably, the present approach is constrained to structured grids due to the inherent nature of CNNs, which are designed for uniform mesh processing. Additionally, errors are introduced when utilizing coarse grids, with cumulative effects occurring when employing neural networks for prediction tasks.

To address the limitations stemming from the use of structured grids and the accumulation of errors when using coarse grids, future works could explore several potential approaches for improvement. First, assembling different loss functions with trainable weights could refine the predictions. Additionally, exploring alternative network architectures, such as graph convolutional networks, which can better handle non-uniform grids, could be beneficial. Finally, conducting a comprehensive sensitivity analysis on grid resolution and network architecture parameters may provide insights into optimizing the model's performance on non-uniform grids. By addressing these aspects, future work can strive to enhance the accuracy and applicability of neural network models in simulating unsteady fluid flows across a broader range of grid types and problems.

ACKNOWLEDGMENT

The research, undertaken in the Rolls-Royce Corporate Lab @ Nanyang Technological University, is supported by the

Singapore Government (Industry Alignment fund IAF-ICP Grant - I1801E0033). We thank Wang Yi and Bryce Conduit from Rolls-Royce plc. for their feedback on this work.

REFERENCES

[1] Ricardo Vinuesa and Steven L Brunton. "Enhancing computational fluid dynamics with machine learning". In: *Nature Computational Science* 2.6 (2022), pp. 358–366.

[2] Steven A Orszag. "Analytical theories of turbulence". In: *Journal of Fluid Mechanics* 41.2 (1970), pp. 363–386.

[3] James W Deardorff. "A numerical study of three-dimensional turbulent channel flow at large Reynolds numbers". In: *Journal of Fluid Mechanics* 41.2 (1970), pp. 453–480.

[4] Hamn C Chen, Viredra C Patel, and Shihao Ju. "Solutions of Reynolds-averaged Navier-Stokes equations for three-dimensional incompressible flows". In: *Journal of Computational Physics* 88.2 (1990), pp. 305–336.

[5] Dmitrii Kochkov et al. "Machine learning–accelerated computational fluid dynamics". In: *Proceedings of the National Academy of Sciences* 118.21 (2021), e2101784118.

[6] Octavi Obiols-Sales et al. "CFDNet: A deep learning-based accelerator for fluid simulations". In: *Proceedings of the 34th ACM international conference on supercomputing*. 2020, pp. 1–12.

[7] Filipe De Avila Belbute-Peres, Thomas Economon, and Zico Kolter. "Combining differentiable PDE solvers and graph neural networks for fluid flow prediction". In: *international conference on machine learning*. PMLR. 2020, pp. 2402–2411.

[8] Tobias Pfaff et al. "Learning mesh-based simulation with graph networks". In: *arXiv preprint arXiv:2010.03409* (2020).

[9] Alvaro Sanchez-Gonzalez et al. "Learning to simulate complex physics with graph networks". In: *International conference on machine learning*. PMLR. 2020, pp. 8459–8468.

[10] Krzysztof Rojek, Roman Wyrzykowski, and Pawel Gepner. "Ai-accelerated cfd simulation based on openfoam and cpu/gpu computing". In: *Computational Science–ICCS 2021: 21st International Conference, Krakow, Poland, June 16–18, 2021, Proceedings, Part II 21*. Springer. 2021, pp. 373–385.

[11] Corentin J Lapeyre et al. "Training convolutional neural networks to estimate turbulent sub-grid scale reaction rates". In: *Combustion and Flame* 203 (2019), pp. 255–264.

[12] Ryan Keisler. "Forecasting global weather with graph neural networks". In: *arXiv preprint arXiv:2202.07575* (2022).

[13] Meire Fortunato et al. "MultiScale MeshGraphNets". In: *ICML 2022 2nd AI for Science Workshop*.

[14] Florent Bonnet et al. "AirfRANS: High Fidelity Computational Fluid Dynamics Dataset for Approximating Reynolds-Averaged Navier-Stokes Solutions". In: *arXiv preprint arXiv:2212.07564* (2022).

[15] Andrea Beck, David Flad, and Claus-Dieter Munz. "Deep neural networks for data-driven LES closure models". In: *Journal of Computational Physics* 398 (2019), p. 108910.

[16] Romit Maulik et al. "Subgrid modelling for two-dimensional turbulence using neural networks". In: *Journal of Fluid Mechanics* 858 (2019), pp. 122–144.

[17] Rishikesh Ranade, Chris Hill, and Jay Pathak. "DiscretizationNet: A machine-learning based solver for Navier–Stokes equations using finite volume discretization". In: *Computer Methods in Applied Mechanics and Engineering* 378 (2021), p. 113722.

[18] Oussama Boussif et al. "MAgNet: Mesh Agnostic Neural PDE Solver". In: *arXiv preprint arXiv:2210.05495* (2022).

[19] Masanobu Horie and Naoto Mitsume. "Physics-Embedded Neural Networks: Graph Neural PDE Solvers with Mixed Boundary Conditions". In: *Advances in Neural Information Processing Systems*.

[20] Zhilu Zhang and Mert Sabuncu. "Generalized cross entropy loss for training deep neural networks with noisy labels". In: *Advances in neural information processing systems* 31 (2018).

[21] Yichao Wu and Yufeng Liu. "Robust truncated hinge loss support vector machines". In: *Journal of the American Statistical Association* (2007), pp. 974–983.

[22] Gregory P Meyer. "An alternative probabilistic interpretation of the huber loss". In: *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*. 2021, pp. 5261–5269.

[23] John R Hershey and Peder A Olsen. "Approximating the Kullback Leibler divergence between Gaussian mixture models". In: *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07*. Vol. 4. IEEE. 2007, pp. IV–317.

[24] Peter K Sweby. "High resolution schemes using flux limiters for hyperbolic conservation laws". In: *SIAM journal on numerical analysis* 21.5 (1984), pp. 995–1011.

[25] Robert E Lynch, John R Rice, and Donald H Thomas. "Direct solution of partial difference equations by tensor product methods". In: *Numerische Mathematik* 6 (1964), pp. 185–199.

[26] Vladimir Igorevich Arnol'd and Lev Dmitrievich Meshalkin. "An Kolmogorov's seminar on selected problems of analysis (1958/1959)". In: *Uspekhi Matematicheskikh Nauk* 15.1 (1960), pp. 247–250.