RANS-PINN based Simulation Surrogates for Predicting Turbulent Flows

Shinjan Ghosh [1] **Amit Chakraborty** [1] **Georgia Olympia Brikis** [1] **Biswadip Dey** [1]

Abstract

Physics-informed neural networks (PINNs) provide a framework to build surrogate models for dynamical systems governed by differential equations. During the learning process, PINNs incorporate a physics-based regularization term within the loss function to enhance generalization performance. Since simulating dynamics controlled by partial differential equations (PDEs) can be computationally expensive, PINNs have gained popularity in learning parametric surrogates for fluid flow problems governed by Navier-Stokes equations. In this work, we introduce RANS-PINN, a modified PINN framework, to predict flow fields (i.e., velocity and pressure) in high Reynolds number turbulent flow regimes. To account for the additional complexity introduced by turbulence, RANS-PINN employs a 2-equation eddy viscosity model based on a Reynolds-averaged Navier-Stokes (RANS) formulation. Furthermore, we adopt a novel training approach that ensures effective initialization and balance among the various components of the loss function. The effectiveness of the RANS-PINN framework is then demonstrated using a parametric PINN.

1. Introduction

The traditional approach to designing complex devices and systems, for example, aerodynamic surfaces and thermal management systems, involves a back-and-forth interplay between exploring the design and operating space and assessing performance through computationally intensive computational fluid dynamics (CFD) simulations. However, the high computational cost associated with high-fidelity CFD solvers like Simcenter Star-CCM+ or Ansys Fluent undeniably curtails the overall scope of the design optimization process, often leading to suboptimal design choices. In this context, neural networks, with their expressiveness to capture pertinent functional relationships between initial/boundary conditions and the solution field of a PDE and the ability to predict simulation outcomes by invoking a single forward pass, offer an excellent tool for building fast and accurate surrogate models for CFD simulations. Such deep learning based approaches can accelerate design evaluations significantly, facilitating the generation of enhanced design choices through fast predictions of simulation outcomes.

In recent years, there has been considerable attention given to the use of deep learning methods to expedite CFD simulations and thereby improve engineering design processes (Vinuesa & Brunton, 2022; Warey et al., 2020; Zhang et al., 2022). While some approaches use deep learning to accelerate traditional CFD solvers (Hsieh et al., 2019; Kochkov et al., 2021a), a certain body of research treats the flow problems as problems defined over a cartesian grid or an irregular mesh and uses techniques involving convolutional or graph neural operators to predict the flow fields (Hennigh, 2017; Jiang et al., 2020; Wang et al., 2020). Alternatively, in another line of work, physics-informed neural networks (PINNs) exploit *automatic differentiation* and incorporate the underlying PDEs to approximate the solution field (Raissi et al., 2019; White et al., 2019; Nabian & Meidani, 2020; Zhang et al., 2020; Jin et al., 2021). In addition, self-supervised learning methods for solving PDEs with PINNs have also been explored (Dwivedi et al., 2019; Lu et al., 2019; Nabian & Meidani, 2019). This expanding body of research demonstrates the ability of ML-based approaches to accurately predict simulation outcomes, such as flow and temperature profiles over a spatiotemporal domain, utilizing both mesh-based and mesh-free techniques. Notably, the inclusion of physics-based regularization in these formulations have proven instrumental in enhancing the quality of the results.

PINNs combine differential equations, such as compressible and incompressible Navier-Stokes equations, with experimental data or high-fidelity numerical simulations. While their ability to replace existing CFD solvers is a matter of debate, PINNs can accelerate simulations (Kochkov et al., 2021b), reconstruct flow domains from a limited sensor or experimental data (Wang et al., 2022), and create parametric surrogates for design exploration and optimization (Olden-

[1]Siemens Technology, Princeton, NJ 08536, USA. Correspondence to: Shinjan Ghosh <shinjan.ghosh@siemens.com>, Biswadip Dey <biswadip.dey@siemens.com>.

Accepted after peer-review at the 1st workshop on Synergy of Scientific and Machine Learning Modeling, SynS & ML ICML, Honolulu, Hawaii, USA. July, 2023. Copyright 2023 by the author(s).

burg et al., 2022; Sun et al., 2023). However, current PINN methods encounter challenges due to the complex interaction among the individual components of the loss function (both supervised and unsupervised), particularly when dealing with high-dimensional, non-convex PDE based losses. These challenges become more pronounced as the physics of the problem becomes more intricate, e.g., turbulent flows. RANS, the most commonly used turbulent CFD simulation tool, offers reasonably accurate solutions at a lower computational cost compared to high-fidelity *direct numerical simulation* (DNS) and *large eddy simulation* (LES), which require even finer mesh refinement to adequately capture all turbulence scales, further increasing computation time. Since its introduction by Launder & Spalding (1974), k-ϵ model (k is the turbulent kinetic energy and ϵ is the turbulent dissipation rate) has been established as a preferred model for efficient computation and real-world problems (Yang & Shih, 1993; Scott-Pomerantz, 2004; Ghosh et al., 2022).

From this perspective, incorporating RANS-based turbulence modeling can significantly expand the application of PINNs in real-world simulation and design problems. However, using PINNs for RANS-based turbulence modeling is yet to be thoroughly studied (Majchrzak et al., 2023). Previous research by Eivazi et al. (2022) employed RANS within PINNs but utilized a Reynolds-stress formulation instead of a 2-equation model like k-ϵ. In contrast, Xu et al. (2021) employed a PINN with RANS formulation to calculate missing flow components. In this study, we focus on constructing PINN-based surrogate models for turbulent flow problems using a RANS formulation, specifically the k-ϵ model, along with relevant data. We refer to the resulting solution as RANS-PINN and implement it using Nvidia Modulus (22.03) (Mod; Hennigh et al., 2021). The proposed training regime first pre-trains the network using data losses and then introduces the physics losses in a carefully crafted manner. We first assess RANS-PINN on three distinct geometries: a cylinder, an airfoil, and flow over backward facing step; and, then employ it to learn a parametric PINN for flow over a cylinder. This approach improves upon the existing turbulence modeling capabilities of Nvidia Modulus, while also adding to the very few existing studies on RANS based turbulence modeling using PINNs.

2. RANS-PINN

2.1. Governing physics

The underlying physics is governed by the continuity equation (to *conserve mass*), Navier-Stokes equation (to *conserve momentum*), and standard k-ϵ turbulence model. By letting u and p denote the flow velocity and pressure, respectively, continuity and Navier-Stokes equation can be expressed as:

$$\textbf{NS:} \quad \nabla(u) = 0$$
$$\textbf{Cont:} \quad \rho(u \cdot \nabla)u + \nabla(p) - \mu_{eff}\nabla^2 u = 0,$$

where ρ is density of the fluid, ∇ denotes the vector differential operator, and $\mu_{eff} := \mu + \mu_t = \mu + 0.09k^2/\epsilon$ represents the effective viscosity, i.e., the sum of molecular viscosity (μ) and turbulent viscosity (μ_t). In addition, the k-ϵ turbulence model can be expressed as:

$$k\textbf{:} \quad \nabla(\rho u k) = \nabla\left[\left(\mu + \frac{\mu_t}{\sigma_k}\right)\nabla k\right] + P_k - \epsilon$$
$$\epsilon\textbf{:} \quad \nabla(\rho u \epsilon) = \nabla\left[\left(\mu + \frac{\mu_t}{\sigma_\epsilon}\right)\nabla \epsilon\right] + (C_1 P_\epsilon + C_2\epsilon)\frac{\epsilon}{k}$$

where, $C_1 = 1.44$, $C_2 = 1.92$, $\sigma_k = 1$, and $\sigma_\epsilon = 1.3$ are empirical model constants. In addition, P_k and P_ϵ are production terms. The *Reynolds number* for this system is defined as: $Re = \rho u_{inlet} L/\mu$, where u_{inlet} is the inlet velocity and L is the characteristic length.

2.2. RANS-PINN architecture and training regime

The RANS-PINN architecture (Fig. 1) uses Fourier neural operators (Li et al., 2021) with their default hyperparameters used in Modulus (Mod). For each of the individual output variables (i.e., u, p, k, and ϵ), we use separate neural networks, all sharing the same input variables consisting of positional coordinates (x, y) and the associated Reynolds number. These networks are connected to the supervised/data loss, as well as the nodes of the PDE loss components.

Conventional approaches to training PINNs involve introducing data and PDE losses simultaneously at the start of the training phase, often with equal weight multipliers. However, they often results in noisy training losses, slow convergence, and high validation error. RANS-PINN addresses these challenges by employing a pre-training step that only uses the data-driven supervised loss. During pre-training, each of the individual networks is updated independently using their corresponding data loss. Following pre-training, we introduce the PDE constraints into the loss function. Moreover, to normalize the effect of the individual components of the PDE loss function, we scale them by the inverse of their corresponding residual values. We then use *Adam* with a decaying step size (with the initial step size of 0.001 and a decay rate of 0.95) until the training loss converges.

To address the challenges associated with abrupt changes observed in the turbulence dissipation term ϵ near wall and free shear regions, we use a *logarithmic loss function* for both data and PDE losses associated with ϵ. Everything else is computed using an *MSE loss function*. The overall loss function can then be expressed as:

$$\mathcal{L} = \mathcal{L}_{data} + \mathcal{L}_{BC} + \mathcal{L}_{PDE}, \tag{1}$$

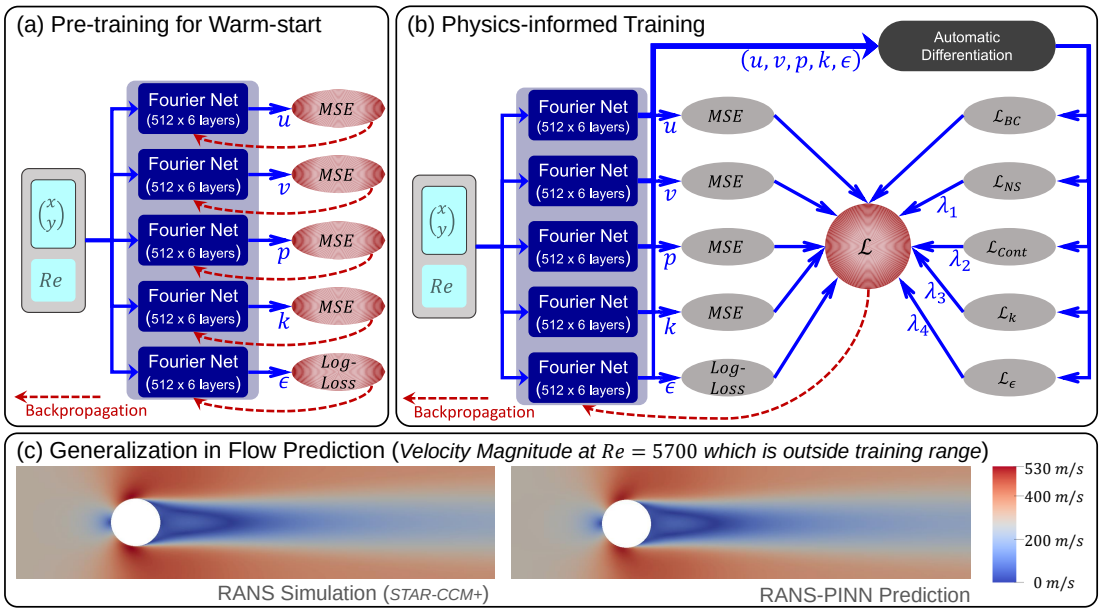

Figure 1: RANS-PINN framework for learning surrogates to predict turbulent flow.

where the PDE loss is defined with weights λ_i's as:

$$\mathcal{L}_{PDE} = \lambda_1 \mathcal{L}_{NS} + \lambda_2 \mathcal{L}_{Cont} + \lambda_3 \mathcal{L}_k + \lambda_4 \mathcal{L}_\epsilon. \quad (2)$$

3. Results and discussions

3.1. Dataset generation using CFD simulation

In this study, we employ Simcenter STAR-CCM+ (*Release 17.02.008*) to simulate turbulent flow scenarios using RANS CFD with the k-ϵ turbulence model. Automatic meshers have been used for each case, with refinement near walls for low wall $y+$, and wall functions for turbulence quantities. Moreover, we have used wake refinements to simulate flow around the cylinder (Fig. 2) and the airfoil. The data generated from the simulation is then normalized using the non-dimensional version of the underlying dynamics (i.e., continuity, Navier-Stokes, and RANS equations). We bring the range of various variables to a comparable order of magnitude by normalizing the spatial coordinates, the velocity, and the pressure with the characteristic length, the inlet velocity, and the dynamic pressure, respectively. Later, the data is denormalized again before visualization.

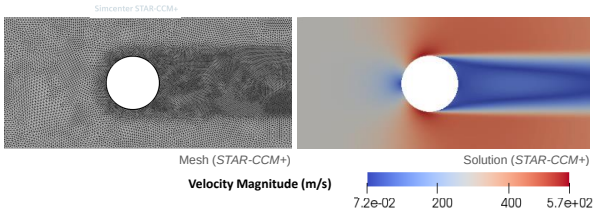

Figure 2: Partial view of mesh for flow over a cylinder with refinement at the cylinder surface and wake regions. This mesh is also used for point cloud sampling in PINN training, and takes into account the density variations. Velocity profile shows gradients in regions of refinement.

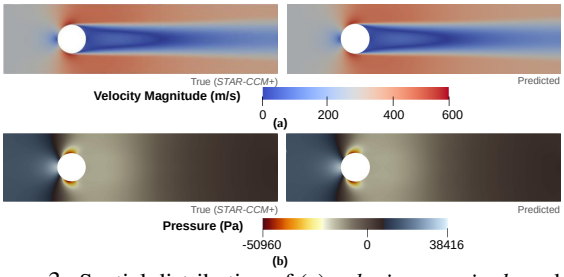

Figure 3: Spatial distribution of (a) *velocity magnitude* and (b) *pressure* for a $Re = 5600$ flow over the cylinder.

3.2. Flow over a cylinder

While the primary objective of this work is to construct a parametric PINN capable of accommodating varying Reynolds numbers (Re), an initial investigation is conducted using single CFD cases (at a fixed Re) to assess the optimal training regime. Flow over a cylinder is a well-studied problem in CFD, for both laminar and turbulent flows. The cylindrical obstacle causes a stagnation zone, and the flow diverts around the obstacle. As a result, flow separation occurs and vortex shedding can be seen in the wake. However, steady RANS models average out the periodic unsteady behaviour, resulting in the time averaged flow field. In this work, we employ a constant velocity inlet, along

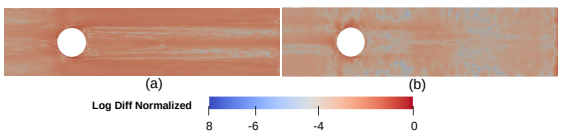

Figure 4: Prediction error in (a) *velocity* and (b) *pressure*. To highlight the prediction error, we use normalized values of the logarithm of difference between the true and the predicted values.

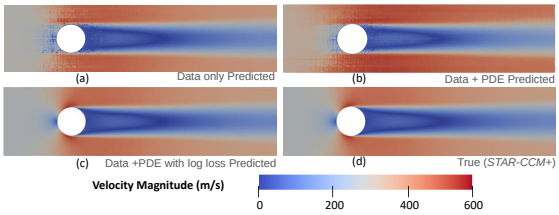

Figure 5: Impact of various choices for the loss function. This figure compares the magnitude of the velocity predicted by a PINN trained with (a) only data loss, (b) both data and PDE loss, and (c) both data and PDE loss with a log-loss for ϵ against its true value from STAR-CCM+ simulation.

with symmetry planes on the top and bottom walls and a zero pressure outlet. For training, 3000 spatially distributed CFD data points are randomly sampled, with an additional 3000 points dedicated to PDE losses. Fig. 3a illustrates the comparison between true and predicted velocity fields, showcasing the aforementioned flow phenomena. The pressure plots (Fig. 3b) show a high pressure stagnation region as well as the low pressure flow separation region in both the true and predicted cases. The differences between true and predicted velocities and pressure for the log-loss training case are shown in Fig. 4. Major losses occur around the cylinder walls, which is known to be a challenging region for all turbulence models due to steep gradients. Moreover, the challenges with using only the data loss or the data+PDE loss but without the logarithmic loss function for ϵ are highlighted in Fig. 5. These choices for the loss function yield flow fields with discontinuities and noise stemming from the combination of data and physics losses. This is further reflected in the validation error values reported in Table 1. In conclusion, the proposed training regime for RANS-PINN exhibits lower validation losses as well as superior predictive performance.

Table 1: Validation errors for flow over cylinder

LOSS FUNCTION	X VEL	Y VEL	PRESSURE
DATA ONLY	0.205	0.284	0.029
DATA+PDE	0.187	0.474	0.066
DATA+PDE W/ LOG-LOSS	0.014	0.03	0.105

3.3. Test on other geometries: Flow over a backwards facing step and NACA 2412 airfoil at a single Re

To understand the general efficacy of the proposed training method, two additional geometries were chosen for investigation. The first geometry involves airfoils which represents external flows, where a pressure gradient is established between the top and bottom surfaces due to acceleration of flow over the top surface (seen in darker red zones of velocity in Fig. 6a and higher pressure magnitudes in Fig. 6b), which causes lift. The second geometry consists of a backwards facing step (Fig. 7), where a separation bubble form due to sudden expansion in the channel. This leads to flow separation and detachment and then re-attachment. Both cases had no-slip walls and constant velocity inlet boundary conditions with a zero pressure exit. Low validation error in Table 2 and visual inspection of Fig. 6 and Fig. 7 show that the flow fields have been successfully predicted.

Table 2: Validation errors for NACA airfoil ($Re = 3 \times 10^5$) and backward facing step ($Re = 5600$).

CASE	X VEL	Y VEL	PRESSURE
NACA 2412	0.091	0.131	0.022
BACKWARDS FACING STEP	0.024	0.146	0.137

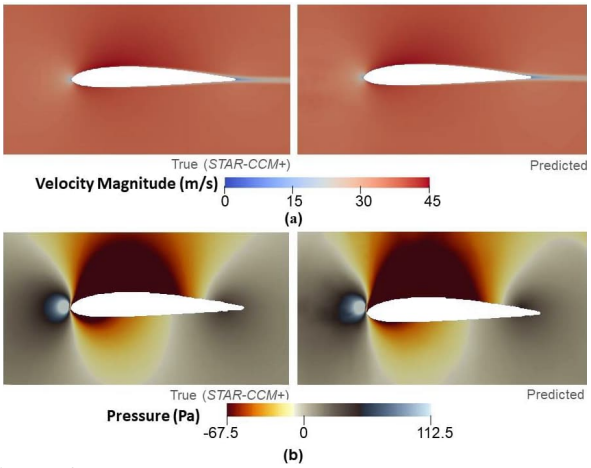

Figure 6: Spatial distribution of (a) *velocity magnitude* and (b) *pressure* for a $Re = 3 \times 10^5$ flow around a NACA 2412 airfoil.

3.4. Parametric PINN for flow over a cylinder

After establishing the training regime with these three flow geometries, we revisit the *flow over a cylinder* problem for creating a parametric PINN. The parametric PINN can predict outcomes of CFD simulations for unseen flow scenarios, in particular for any given Reynolds number (Re), which depends on the inlet velocity inlet velocities. We achieve this by including the Reynolds number as an additional input to the individual neural networks.

In this study, we ran CFD simulations for six different Reynolds numbers ranging from 2800 to 5600, with uniform spacing between the values. We sampled 3000 spatial data points from each simulation and utilized them along with PDE losses to train the parametric PINN with Re as the

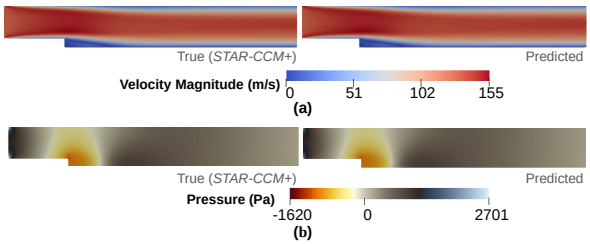

Figure 7: Spatial distribution of (a) *velocity magnitude* and (b) *pressure* for a $Re = 5600$ flow over a backward facing step.

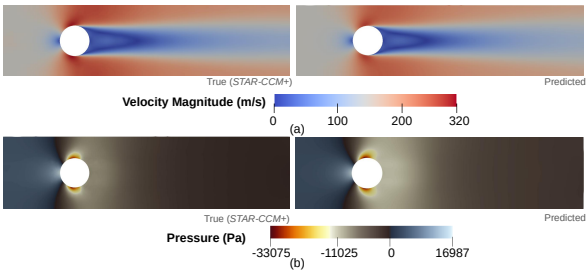

Figure 8: Spatial distribution of (a) *velocity magnitude* and (b) *pressure* for a $Re = 3140$ flow over the cylinder. We have used a parametric PINN to predict the velocity and the pressure.

underlying parameter. Although each CFD simulation has 61000 mesh data points, we trained the parametric PINN using only 3000 points, resulting in faster convergence. By leveraging the parametric PINN, we can now predict flow fields for any given Reynolds number. This is highly beneficial for design optimization and exploration studies, as it eliminates the need for additional CFD data to predict primary flow variables across the entire solution domain. Moreover, compared to the traditional approaches, where each CFD simulation run takes approximately 24 core minutes, the parametric PINN can yield results in a near real-time fashion, significantly accelerating the overall process.

Table 3: Generalization error for parametric PINNs (unseen cases)

CASE	X VEL	Y VEL	PRESSURE
$Re = 3140$	0.139	0.289	0.164
$Re = 5700$	0.153	0.200	0.079

Fig. 8a and Fig. 8b show the velocity and pressure distributions for $Re = 3140$. On the other hand, Fig. 9a and Fig. 9b display the velocity and pressure distributions for $Re = 5700$, which falls outside the training range. In each validation case, we examined 61000 mesh points. Table 3 presents the overall error metrics for validation in the case of the parametric PINN.

4. Conclusion

PINN-based approaches to learning surrogate models for spatiotemporal systems governed by nonlinear PDEs are relatively common in the literature. However, despite playing an instrumental role in many real-world applications, two-equation RANS turbulence models are yet to be integrated into PINN-based approaches. In this work, we adopt a novel training regime to ensure the successful integration of RANS turbulence model physics into PINNs. Once trained with a limited amount of CFD data, RANS-PINN can yield accurate predictions of overall flow fields for a single Reynolds number. Building upon the successful outcomes of these evaluations for three different flow geometries (flow over a cylinder, a backward-facing step, and a NACA 2412 airfoil), we develop a parametric version of the RANS-PINN to predict flow over a cylinder for any given/unforeseen Reynolds numbers. The parametric RANS-PINN, which highlights how whole simulation cases can be inferred without requiring any CFD data from that specific Reynolds number, offers significant potential in solving design exploration and inverse problems for many real world applications.

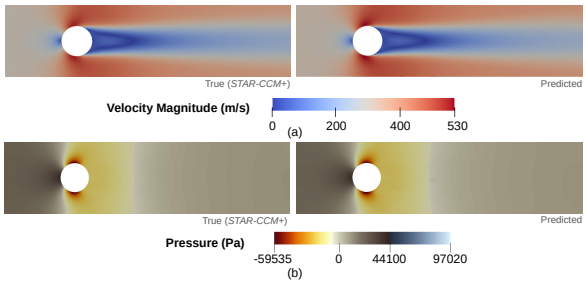

Figure 9: Spatial distribution of (a) *velocity magnitude* and (b) *pressure* for a $Re = 5700$ flow over the cylinder. We have used a parametric PINN to predict the velocity and the pressure.

Broader impact

The current work focuses on turbulent flow problems with two-equation turbulence models. While this perspective is not commonly explored in the PINN literature, these turbulence models hold significant importance in many industrial and academic settings where a lack of computing resources prevents the use of Direct Numerical Simulation (DNS) and Large Eddy Simulation (LES). We can effectively tackle design and inverse problems in many real-world cases by employing a turbulent flow PINN, such as RANS-PINN. The ability to reconstruct a flow field from limited data can help in real-world problems with limited sensor data. Moreover, a parametric PINN trained with minimal CFD data adds significant value to design exploration and optimization by offering a convenient, fast, and computationally efficient means to predict simulation outcomes.

References

Nvidia Modulus. https://developer.nvidia.com/modulus. Accessed: 2022-12-15.

Dwivedi, V., Parashar, N., and Srinivasan, B. Distributed physics informed neural network for data-efficient solution to partial differential equations. *arXiv preprint arXiv:1907.08967*, 2019.

Eivazi, H., Tahani, M., Schlatter, P., and Vinuesa, R. Physics-informed neural networks for solving reynolds-averaged navier–stokes equations. *Physics of Fluids*, 34(7):075117, 2022.

Ghosh, S., Wardell, R., Mondal, S., Fernandez, E., Ray, A., and Kapat, J. Topology Optimization and Experimen-

tal Validation of an Additively Manufactured U-Bend Channel. *Journal of Fluids Engineering*, 144(7), 02 2022. ISSN 0098-2202. doi: 10.1115/1.4052928. URL https://doi.org/10.1115/1.4052928. 071206.

Hennigh, O. Lat-net: Compressing lattice Boltzmann flow simulations using deep neural networks. *arXiv preprint arXiv:1705.09036*, 2017.

Hennigh, O., Narasimhan, S., Nabian, M. A., Subramaniam, A., Tangsali, K., Fang, Z., Rietmann, M., Byeon, W., and Choudhry, S. NVIDIA SimNet™: An AI-accelerated multi-physics simulation framework. In *Computational Science–ICCS 2021: 21st International Conference, Krakow, Poland, June 16–18, 2021, Proceedings, Part V*, pp. 447–461. Springer, 2021.

Hsieh, J.-T., Zhao, S., Eismann, S., Mirabella, L., and Ermon, S. Learning neural pde solvers with convergence guarantees, 2019.

Jiang, C. M., Esmaeilzadeh, S., Azizzadenesheli, K., Kashinath, K., Mustafa, M., Tchelepi, H. A., Marcus, P., Anandkumar, A., et al. MeshfreeFlowNet: A Physics-Constrained Deep Continuous Space-Time Super-Resolution Framework. *arXiv preprint arXiv:2005.01463*, 2020.

Jin, X., Cai, S., Li, H., and Karniadakis, G. E. NSFnets (Navier-Stokes flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations. *Journal of Computational Physics*, 426:109951, 2021.

Kochkov, D., Smith, J. A., Alieva, A., Wang, Q., Brenner, M. P., and Hoyer, S. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21):e2101784118, 2021a.

Kochkov, D., Smith, J. A., Alieva, A., Wang, Q., Brenner, M. P., and Hoyer, S. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21):e2101784118, 2021b.

Launder, B. and Spalding, D. The numerical computation of turbulent flows. *Computer Methods in Applied Mechanics and Engineering*, 3(2):269–289, 1974. ISSN 0045-7825. doi: https://doi.org/10.1016/0045-7825(74)90029-2. URL https://www.sciencedirect.com/science/article/pii/0045782574900292.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Fourier neural operator for parametric partial differential equations. *arXiv Preprint*, 2010.08895, 2021.

Lu, L., Meng, X., Mao, Z., and Karniadakis, G. E. Deepxde: A deep learning library for solving differential equations. *arXiv preprint arXiv:1907.04502*, 2019.

Majchrzak, M., Marciniak-Lukasiak, K., and Lukasiak, P. A survey on the application of machine learning in turbulent flow simulations. *Energies*, 16(4), 2023.

Nabian, M. A. and Meidani, H. A deep learning solution approach for high-dimensional random differential equations. *Probabilistic Engineering Mechanics*, 57:14–25, 2019.

Nabian, M. A. and Meidani, H. Physics-driven regularization of deep neural networks for enhanced engineering design and analysis. *Journal of Computing and Information Science in Engineering*, 20(1), 2020.

Oldenburg, J., Borowski, F., Öner, A., Schmitz, K.-P., and Stiehm, M. Geometry aware physics informed neural network surrogate for solving navier–stokes equation (gapinn). *Advanced Modeling and Simulation in Engineering Sciences*, 9(1):8, 2022.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

Scott-Pomerantz, C. D. *The k-epsilon model in the theory of turbulence*. PhD thesis, University of Pittsburgh, 2004.

Sun, Y., Sengupta, U., and Juniper, M. Physics-informed deep learning for simultaneous surrogate modeling and pde-constrained optimization of an airfoil geometry. *Computer Methods in Applied Mechanics and Engineering*, 411:116042, 2023.

Vinuesa, R. and Brunton, S. L. Enhancing computational fluid dynamics with machine learning. *Nature Computational Science*, 2(6):358–366, 2022.

Wang, H., Liu, Y., and Wang, S. Dense velocity reconstruction from particle image velocimetry/particle tracking velocimetry using a physics-informed neural network. *Physics of Fluids*, 34(1):017116, 2022.

Wang, R., Kashinath, K., Mustafa, M., Albert, A., and Yu, R. Towards physics-informed deep learning for turbulent flow prediction. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1457–1466, 2020.

Warey, A., Kaushik, S., Khalighi, B., Cruse, M., and Venkatesan, G. Data-driven prediction of vehicle cabin thermal comfort: using machine learning and high-fidelity simulation results. *International Journal of Heat and Mass Transfer*, 148:119083, 2020.

White, C., Ushizima, D., and Farhat, C. Fast neural network predictions from constrained aerodynamics datasets. *arXiv Preprint*, 1902.00091, 2019.

Xu, H., Zhang, W., and Wang, Y. Explore missing flow dynamics by physics-informed deep learning: The parameterized governing systems. *Physics of Fluids*, 33(9): 095116, 2021.

Yang, Z. and Shih, T.-H. New time scale based k-epsilon model for near-wall turbulence. *AIAA journal*, 31(7): 1191–1198, 1993.

Zhang, T., Dey, B., Kakkar, P., Dasgupta, A., and Chakraborty, A. Frequency-compensated pinns for fluid-dynamic design problems. *arXiv Preprint*, 2011.01456, 2020.

Zhang, T., Dey, B., Veeraraghavan, K., Kulkarni, H., and Chakraborty, A. Demystifying the data need of ml-surrogates for cfd simulations, 2022.

A. Appendix: Additional Details

Algorithm 1 Training Regime

Sample data points for data loss, pde loss and validation from mesh
Pretraining individual NNs using data loss
repeat
  Normalize weights of PDE loss components using residual values

until Loss is stationary

The training metrics of the parametric PINN has been showed below in figure 10. As seen below, the primary physics errors are less than 10^{-4}. A brief algorithm of the process has also been shown.

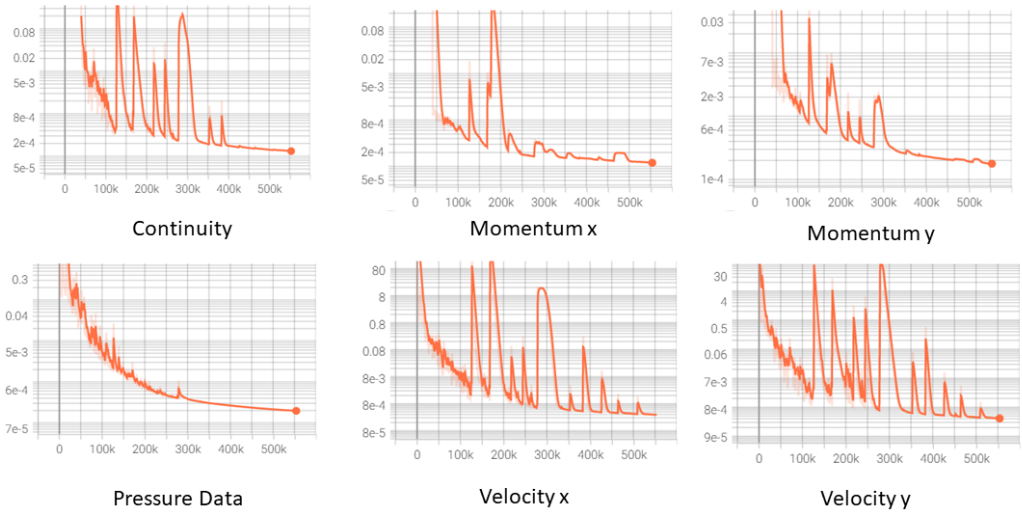

Figure 10: Training metrics of parametric PINN