# Multi-level datasets training method in Physics-Informed Neural Networks

Yao-Hsuan Tsai[1], Hsiao-Tung Juan[1], Pao-Hsiung Chiu[2,*], and Chao-An Lin[1,*]

[1]*Department of Power Mechanical Engineering, National Tsing Hua University, Hsinchu 30013, Taiwan*

[2]*Institute of High Performance Computing, Agency for Science, Technology and Research (A*STAR), Singapore*

[*]*corresponding authors. Email: calin@pme.nthu.edu.tw(C.A. Lin); chiuph@ihpc.a-star.edu.sg(P.H. Chiu)*

Physics-Informed Neural Networks (PINNs) have emerged as a promising methodology for solving partial differential equations (PDEs), gaining significant attention in computer science and various physics-related fields. Despite being demonstrated the ability to incorporate the physics of laws for versatile applications, PINNs still struggle with the challenging problems which are stiff to be solved and/or have high-frequency components in the solutions, resulting in accuracy and convergence issues. It will not only increase computational costs, but also lead to accuracy loss or solution divergence, in the worst scenario. In this study, an alternative approach is proposed to mitigate the above-mentioned problems. Inspired by the multi-grid method in CFD community, the underlying idea of the current approach is to efficiently remove different frequency errors via training with different levels of training samples, resulting in a simpler way to improve the training accuracy without spending time in fine-tuning of neural network structures, loss weights as well as hyper-parameters. To demonstrate the efficacy of current approach, we first investigate the canonical 1D ODE with high-frequency component and 2D convection-diffusion equation with V-cycle training strategy. Finally, the current method is employed for the classical benchmark problem of steady Lid-driven cavity flows at different Reynolds numbers($Re$), to investigate the applicability and efficacy for the problem involved multiple modes of high and low frequency. By virtue of various training sequence modes, improvement through predictions lead to 30% to 60% accuracy improvement. We also investigate the synergies between current method and transfer learning techniques for more challenging problems (i.e., higher $Re$). From the present results, it also revealed that the current framework can produce good predictions even for the case of $Re = 5000$, demonstrating the ability to solve complex high-frequency PDEs.

Keywords: Physics-informed neural networks, Multi-level datasets training, Dimensional analysis weighting

## I. INTRODUCTION

As machine learning continues to advance rapidly, a specific form known as Neural Networks (NN)[1,2] has become a popular tool for solving various physics problems. When NNs are specifically designed to solve partial differential equations (PDEs) that express physical phenomena, they are referred to as Physics-Informed Neural Networks (PINNs)[3–5]. When employ PINNs for solving PDEs, it offer several advantages, such as ability to acquire gradients via automatic differentiation[6], ability to infer the parameters from limited data and ability to train the model with less or zero data, to name a few. and through stochastic gradient descent optimizations[7,8]. Over the past decade, PINNs have emerged as a key research focus in diverse fields, such as structure mechanics[9], biology[10], fluid mechanics[11], electromagnetic[12], forecasting solutions[13] and assisting numerical solutions[14,15] where they have become valuable tools for prediction and simulation.

In the meantime, several key areas of interest have emerged for using PINNs to solve parametric PDEs in fluid simulations. the investigated topics in this domain include solving large problems through domain decomposition[16,17], predicting solutions in numerical discretization forms such as large-eddy simulation (LES)[18–20] and finite difference methods[21], and eliminating losses related to boundary and geometry conditions[22–24], to name a few. In terms of training methods, transfer learning[25,26], meta-learning[27], and curriculum learning[28,29] which leverage similarities between complex problems and simpler ones, allowing models to pre-learn features and enhance accuracy by utilizing parameters from simpler questions, also been intensively studied.

To improve the training, there are two main issues to be take care with: gradient flow pathology[30] and spectral biases[31]. Gradient pathology refers to issues arising from the numerical stiffness of the problem, especially the lack of sufficient collocation points needed for a stable solution. To mitigate this issue, one may employ suitable weightings in loss calculations, with adaptive weightings[30,32–34]. Another advantage of employing the adaptive weightings is that they avoid reliance on trial and error, with the trade-off of slightly affect training speed. In this paper, we proposed an alternative way to balance accuracy and computational efficiency by performing the dimensional analysis beforehand to manually set the weightings.

Spectral bias refers to the tendency of neural networks to prioritize learning low-frequency functions. This issue can be mitigated by increasing the frequency of the input manifold. Tancik et al.[35,36] introduced Fourier feature mapping, which incorporates sinusoidal functions to enhance the model's capability to predict higher frequency functions from the perspective of the Neural

Tangent Kernel (NTK)[37], yielding significant improvements in solutions[38]. Liu et al.[39] proposed a Multi-scale Deep Neural Network (MscaleDNN) structure that employs multiple sub-networks combined with supporting activation functions, enabling the model to learn the target function across different frequency scales and demonstrating improvements in solving the Poisson and Poisson-Boltzmann equations.

On the other hand, It have been the intensively studies in scientific computing community that multi-grid methods can serve as a preconditioner for Fourier analysis-related frequencies[40]. Although this approach has been implemented in some neural networks[41,42], its mainly usage is still limited to the numerical solvers. There are some dataset setting methods which have explored dataset establishment techniques[43,44]. However, their studies mainly focus on a single sample dataset with specific sampling methodologies rather than modifying dataset loading. It is then to motivated us to address the importance the spectral bias problem and explore the practical usage of the multi-grid method.

Different from the study of Riccieti et al.[45] who previously implemented the mutil-grid method via multiple sub-networks to improve convergence rates for solving non-linear equations, the current proposed framework is to combine concepts which inspired by both the multi-grid method and MscaleDNNs, as new training procedure. The fundamental idea of the current approach is to involve the use of varying datasets during training through transfer learning techniques, so as to mitigate the spectral bias problem. The proposed approach not only leverages dataset variation to expose the model to different frequency scales, but enhances its ability to learn target functions effectively. To emphasize the frequency diversity across datasets, different optimizers and schedulers are applied to each dataset. To validate our approach, we first conducted investigations by solving a high-frequency ODE, followed by a canonical convection-diffusion equation, and subsequently, we investigated different training sequence modes on the classical benchmark computational fluid dynamic problem : lid-driven cavity problem. Through these experiments, we observed improved accuracy using our proposed framework compared to conventional methods that rely on a single dataset.

This paper is structured as follows: In Section II, we present an overview of the multi-dataset training method employed in our model, discussing how variations across datasets contribute to enhanced accuracy. This section includes flowcharts and detailed model settings. Section III validates our approach by comparing it to the base model with different training sequence modes through predictions of an 1D high-frequency ODE and 2D convection-diffusion equation, followed

by investigating and assessing the accuracy of and efficiency of the current approach for the 2D Lid-driven cavity flow problem . Finally, Section IV draw the conclusions, addresses current challenges, and outlines potential directions for further methodological improvements.

## II. METHODOLOGY

In this section, a brief overview of our multi-dataset training method with exploration of the multilevel and multi-grid methods from a frequency perspective will be described. Subsequently, the current state of PINN models and their implementations will be proposed, together with the method of the weighting method which inspired by the dimensional analysis.

### A. Multi-scale NNs and Multi-grid method in frequency perspective

Rahaman et al.[31] analyzed the Fourier spectrum within neural networks, focusing on how activation functions of neurons tend to prioritize learning simpler aspects of problems. Their experiments demonstrated a bias towards lower complexity. To counteract this, Liu et al.[39] introduced a multi-scale structure comprising multiple sub-networks that use inputs with varying multipliers to learn across different frequencies. On the other hand, Fourier frequency issues are commonly addressed by multi-grid method[40] in numerical analysis society. Despite the contrasting goals—PINNs aim to establish frequencies, multi-grid methods aim to mitigate them. Both approaches share a common idea that the spectral bias of smaller collocation sets results in lower training frequencies, while higher frequencies are more attainable with increased collocations. Inspired by MscaleDNNs, in this study we propose an approach akin to the multi-grid method, where we dynamically switch datasets to counter both low and high frequencies directly, so as to enhance learning frequencies using different datasets.

To evaluate the effectiveness of different training sequences, in this study we investigated our approach by ranging from a three-level coarse-to-fine mesh approach, to sequences resembling the procedure of multi-grid method, such as the popular V-cycle formulation transitioning from fine to coarse to fine meshes. To efficiently implement the current methods, in this study we integrate them into a training procedure akin to transfer learning[46,47], summarized as follows: (1) Establish multiple datasets, each containing varying numbers of collocation points. (2) Develop a predictive training model. (3) Sequence the training using these datasets, employing distinct optimizers and

learning rate schedules individually for each dataset during training, and connecting the training progress through transfer learning principles.

## B. Physics-Informed Neural Network(PINNs)

In this section, a brief overview of the calculation procedure in Physics-Informed Neural Networks (PINNs) will be discussed, followed by the implementations of our model, including multi-level dataset structures and settings. Finally, a summary of our current neural network architecture and related settings will be provided.

**Overview of PINN calculation:** With the pioneer work of Karniadakis et al.[3–5], physics-informed neural networks (PINNs) provide an alternative methodology to traditional numerical discretizations for solving parametric PDEs with physical interpretations. The typical PINNs employ a multi-layer perceptron (MLP) structure, take spatial inputs $x \in \Omega$ and temporal inputs $t \in$ (0,T] to predict an output $\hat{u}(x,\, t;\, \theta)$ that satisfies specified PDEs. This general framework can be described as follows[3,11]:

$$\mathscr{N}_t[\hat{u}(x,t)] + \mathscr{N}_x[\hat{u}(x,t)] = 0, x \in \Omega, t \in (0,T] \tag{1a}$$

$$\mathscr{B}[\hat{u}(x,t)] = g(x,t), x \in \Omega, t \in (0,T] \tag{1b}$$

$$\hat{u}(x,0) = u_o(x), x \in \Omega \tag{1c}$$

where $\mathscr{N}_t[\cdot]$ and $\mathscr{N}_x[\cdot]$ denote differential operators for temporal and spatial derivatives, such as $u_x$(x,t), $u_{xx}$(x,t), $u_t$(x,t) representing first-order spatial derivatives, second-order spatial derivatives, and temporal derivatives, respectively. $\mathscr{B}[\cdot]$ represents the boundary condition form, such as Dirichlet, Neumann, Robin, and with $g(x,t)$ represent the given boundary conditions. $u_0$(x) is represents $u$ at t = 0 which specifies initial condition.

To generate predictive output from input $x_0 = (x,t)$, each neuron in Neural Network incorporate with following function:

$$f_j(x_0) = \sigma(w_j x_0 + b_j), j = 1, ..., k+1 \tag{2}$$

where ($f_j \in \mathbb{R}^j$) are hidden states, ($\sigma$) denotes activation functions that transform input components, and ($w_j, b_j$) represent weights and biases in neurons, with ($k$) being the total number of neurons in the layer of the Neural Network. Using the predictive output, we calculate the residuals from the PDE and errors from boundary conditions. A model loss ($\mathscr{L}$) is then formulated and used

to update the model parameters ($\theta$) in current epoch ($n$) which comprises sets of weights ($\mathbf{w}_n$) and bias ($\mathbf{b}_n$) for each neuron in the structure. This process can be expressed as follows:

$$\theta_{n+1} = \theta_n - \eta \nabla_\theta \mathscr{L}(\theta_n) \tag{3}$$

In the formulation of PINNs, the loss function typically comprises three primary components $\mathscr{L}_{DE}$, $\mathscr{L}_{BC}$ and $\mathscr{L}_{IC}$. These components correspond to the residuals of differential equations, boundary conditions, and initial conditions, respectively. The general representation of the loss function is as follows:

$$\mathscr{L} \equiv \lambda_{DE}\mathscr{L}_{DE} + \lambda_{BC}\mathscr{L}_{BC} + \lambda_{IC}\mathscr{L}_{IC} \tag{4}$$

This equation represents the weighted combination of these three fundamental components, where $\lambda$ denotes the weighting factor assigned to each component, indicating their relative importance within the overall loss function.

In this paper, the Mean Square Error (MSE) is used as the common loss evaluation function for each loss component:

$$MSE(\hat{u}, u) = \frac{1}{n}\sum_{i=1}^{n}(\hat{u}_i - u_i)^2 \tag{5}$$

where $\hat{u}_i$ is representing model output, and $u_i$ is corresponding target value, e.g., ground truth. With this function we can define loss components in (eq. 4) from (eq.1a-1c) to become following equations:

$$\mathscr{L}_{DE} = MSE(\hat{u}_t[\cdot;\theta] + \mathscr{N}_x[\hat{u}(\cdot;\theta)], 0) \tag{6a}$$

$$\mathscr{L}_{BC} = MSE(\mathscr{B}[\hat{u}(\cdot;\theta)] - g(\cdot), 0) \tag{6b}$$

$$\mathscr{L}_{IC} = MSE(u(\cdot,0) - u_o, 0) \tag{6c}$$

**Neural Network Implementations:** In this study, the Multilayer Perceptron (MLP) structure is employed. To obtain optimize training model, several crucial settings are considered in this paper:

- Activation functions: While the tanh activation function is commonly used in PINNs[27,34], recent studies[36,48] have shown improved accuracy with the sine activation function as a feature mapping for inputs. Therefore, the sine activation function is chosen for the first layer of the neural network. Additionally, the number of neurons in the first layer is significantly larger than in the inner layers, as larger weight distributions in the first layer help escape local minima. For the inner layers, Swish functions are employed, as previous research[49] indicates that this function is less susceptible to vanishing and exploding gradient problems.

- Weight and Bias Initialization: The initialization of weights and biases is crucial for network training. In this paper, the weights of the first layer , particularly the sine activation functions, are initialized from a normal distribution $\mathcal{N}(0, \sigma^2)$, $\sigma$ =1. The weights of subsequent layers are initialized using the He uniform initialization[50,51] or the similar Xavier uniform initialization, as the setting has been proofed to be more effective[48], while the biases are initialized from a uniform distribution, following the default settings in PyTorch[52].

- Optimizer and Scheduler Settings: The commonly used optimizer is Adam[53], a variant of stochastic gradient descent. It is accompanied by the ReduceLROnPlateau or StepLR scheduler, which dynamically adjusts the learning rate based on the monitored loss value. This adaptive learning rate adjustment enhances model convergence rate and improves training efficiency.

### C. Dimensional analysis weighting method root scheme($DW_{root}$)

The Dimensional analysis weighting method was introduced by Chou et al.[54]. This method involves analyzing the order of magnitude for each component within the loss functions. The primary concept behind this method is to establish a balance among the weights of different components by considering their ratios, ensuring each component carries a similar level of significance. The method implementation is presented in eq.17 and eq.18. This method serves as an alternative way to adapt weighting and offer two main advantages: (1) Good convergence throughout the training sequence and (2) No need for additional computations within the model. For the sake of completeness, the proposed algorithm of this paper is summarized in Alg.1

## III. RESULTS AND DISCUSSION

In this section, the proposed method is demonstrated through various training sequence modes that involve multiple dataset levels to showcase improvements. The examples below illustrate test problems that highlight the efficacy of the method: (i) solve a high-frequency sine function to demonstrate its capability in addressing the spectral bias problem; (ii) investigate different training sequence modes via 2D Smith-Hutton problem to reveal the accuracy and efficiency of current methods; (iii) investigate the 2D steady Lid-driven cavity flow to further demonstrate the applicability, accuracy and efficiency for the challenging system of partial differential equations, e.g.,

**Algorithm 1:** Multi-dataset PINN

**Step 1:** Set up input matrices and model parameters ;

**Step 2:** Construct a Fully connected PINN model with initialized weights and bias (for $k_i$-th iteration represented as $w_k$);

**Step 3:** Declare dimensional analyzed weights;

**Step 4:** Setup optimizers and schedulers for different dataset level;

**Step 5:** Do $K$ gradient descent iterations in each level defined level sequence to update neural network:;

**for** $Level = 1$ *to* $N-1$ **do**

  **for** $k = 1$ *to* $K$ **do**

    Tune neural network weights and bias with respect to $\mathscr{L}$;

    $w_{k+1} \leftarrow \mathscr{L}(w_k)$

  **end**

**end**

**Step 6:** Train the last training level **for** $k = (N-1) * K$ *to* $K_{max}$ **do**

  Tune neural network weights and bias with respect to $\mathscr{L}$;

  $w_{k+1} \leftarrow \mathscr{L}(w_k)$

**end**

Navier-Stokes equations. The limitations with specified networks and datasets, followed by comparisons of the proposed model's accuracy with results from various studies will also be examined.

## A. Experimental setup

For 1D ODE problem , A 2-level random collocation set using 16 and 64 points will be studied. For 2D problems, 3-level uniform collocation sets employ 435, 1770, and 6670 points for Smith-Hutton problem, while different random collocation point sets for lid-driven cavity flow problem are employed, respectively, as shown in Table. I and II. Each collocation set in this paper separates inner points and boundary points for training: inner points are used for calculating the residuals of differential equations ($\mathscr{L}_{DE}$ ), while boundary points satisfy the boundary conditions ($\mathscr{L}_{BC}$). Details of which will be provided in subsequent sections. All PINN structures utilize a

fully connected feed-forward architecture with activation functions shown separately in sections. The width and depth of the neural networks are tailored for each experiment to achieve reasonable results while minimizing training costs from excessive model parameters. Model accuracy is assessed using the Relative $L^2$ error (Rel. $L_2$) metric, defined as follows:

$$\frac{||u-\hat{u}||_2}{||u||_2} \tag{7}$$

For all experiments, we used the Adam optimizer with full batch training, along with either the *StepLR* scheduler or the *ReduceLROnPlateau* scheduler, both provided by PyTorch. All experiments were conducted on a system equipped with an Intel i9-13900K processor and a NVIDIA RTX 4070Ti GPU with 12GB of VRAM.

| Problem | High frequency ODE (sec.III B) | Smith-Hutton problem (sec.III C) Pe = 1000 |
|---|---|---|
| Govern eq. | eq.9 | eq.12 |
| Neural Network structure | (x)-64-64-64-64-($\hat{u}$) | (x,y)-128-64-64-64-($\hat{u}$) |
| Training collocation points | 64+32 | 6670+1770+435 |
| Initial learning rate | 1e-3 | 1e-3 |
| Max Training epochs | 30,000 | 50,000 |

Above table shows model setting by default, which varies for different tests with details shown in Appendix (sec.V).

TABLE I: Experiments and default setups for 1D ODE and 2D Smith-Hutton problems

## B. 1D high frequency ODE

In order to demonstrate the effectiveness of our method in addressing the aforementioned spectral bias issue, we consider the following 1D sine function:

$$f(x) = sin(10x), x \in [0, 2\pi] \tag{8}$$

which represents a high frequency ordinary differential equation (ODE), expressed as:

$$g(x) = \frac{\partial f(x)}{\partial x} = 10sin(10x) \tag{9}$$

| Problem | Lid-driven cavity (sec.III D) Re=400 | Lid-driven cavity (sec.III D) Re=1000 | Lid-driven cavity (sec.III D) Re=5000 |
|---|---|---|---|
| Govern eq. | eq.16 | eq.16 | eq.16 |
| Neural Network structure | (x,y)-64-20-20-[(20-20-($\hat{\psi}$))-(20-20-($\hat{p}$))] | (x,y)-256-64-64-[(64-64-($\hat{\psi}$))-(64-64-($\hat{p}$))] | (x,y)-256-64-64-[(64-64-($\hat{\psi}$))-(64-64-($\hat{p}$))] |
| Training collocation points | 6565+1685+445 | 6565+1685+445 | 14645 +3725+965 |
| Initial learning rate | 1e-3 | 1e-3 | 1e-3 |
| Max Training epochs | 200,000 /300,000(w-cycle) /200,000(V-cycle) | 212,000(V-cycle) | 1,207,000 (curriculum) |

Above table shows model setting by default, which varies for different tests with details shown in Appendix (sec.V).

TABLE II: Experiments and default setups for Lid-driven cavity problem

where $g(x)$ represents the residual equation for the differential equation loss ($\mathscr{L}_{DE}$), and $f(0) = 0, f(2\pi) = 0$ denote boundary conditions for the boundary loss ($\mathscr{L}_{BC}$).

In this experiment, we set the weighting to unity for clarity of observation. The dataset used in this section, denoted as ($D_1$), which is main collocation in this section , consists of 64 randomly sampled collocation points and is primarily used for training to satisfy the equation. Additionally, a testing set containing 256 randomly sampled points is employed to evaluate the model's accuracy against ground truth over 30,000 Adam epoch. The model structure consists of a 4-layer MLP with a width of 64. Activation functions are expressed as follows:

$$\sigma_l(x) = \begin{cases} sin(2\pi x), & l = 1 \\ sin(x), & 2 \leq l \leq 4 \end{cases} \tag{10}$$

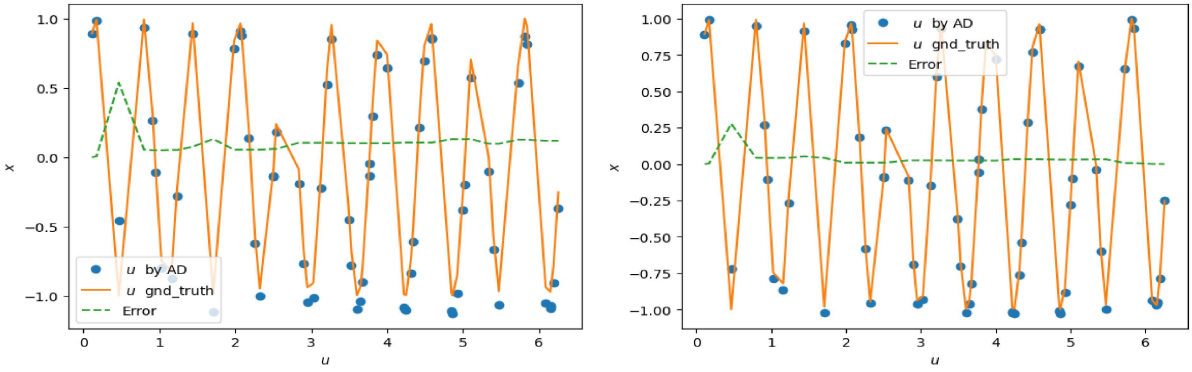
where $l$ is the layer number. The He uniform initialization is employed, while the biases for this problem is set as zero.

To compare training with only main dataset($D_1$) for 30,000 epochs, we introduce an additional auxiliary dataset ($D_2$ in this section) with 32 random samples, to assist $D_1$ in converge. Training of the additional dataset will interrupt the training of $D_1$, which result in following training sequence (Table. I).

| Level | Dataset | training epochs |
|---|---|---|
| 1 | $D_1$ | $1 \leq n \leq 5{,}000$ |
| 2 | $D_2$ | $5{,}001 \leq n \leq 15{,}000$ |
| 3 | $D_1$ | $15{,}001 \leq n \leq 30{,}000$ |

TABLE III: ODE training sequence

During the training of $D_2$, we employ different optimizers and schedulers to facilitate the model's learning of additional frequencies within a shorter training period. This training sequence is structured with the following objectives: the first level establishes the model's recognition of the target dataset, the second level focuses on learning new frequencies, and the final level integrates both learned frequencies. The detailed setting is shown in Appendix.(secV A).




(a) PINN w/o proposed method

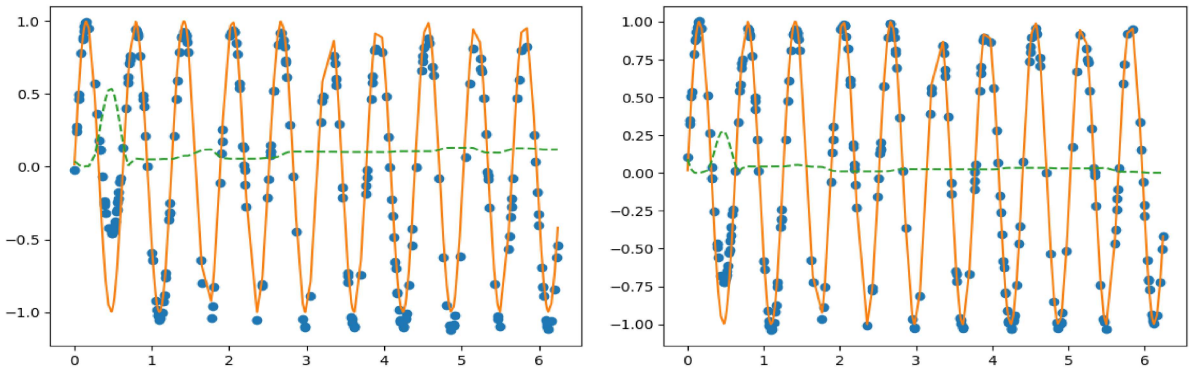
(b) PINN w/ proposed method

FIG. 1: Training result (blue dot), Training ground truth (solid line) and Absolute error (dashed line) after end training for 30,000 epochs

| Model | $\mathscr{L}_{DE}$ | $\mathscr{L}_{BC}$ | Time cost(sec) | Test error(Rel. $L_2$) |
|---|---|---|---|---|
| PINN w/o proposed method | 5.518e-07 | 2.906e-12 | 164 | 1.15e-01 |
| PINN w/ proposed method | 1.863e-06 | 1.120e-10 | 169 | 4.16e-02 |

TABLE IV: ODE(sec.III B) Loss values comparison after 30,000 epochs of training

As shown in Table. IV and Figures 1 and 2, PINNs often struggle to solve high-frequency ODEs despite achieving low loss values. As our method resolve this issue by incorporating additional



(a) PINN w/o proposed method

(b) PINN w/ proposed method

FIG. 2: Test result (blue dot), Test ground truth (solid line) and Absolute error (dashed line) after end training for 30,000 epochs

smaller datasets within the training sequence, which incurs a modest increase in training time but improves performance.

### C. 2D Smith-Hutton problem

To further demonstrate the applicability and efficiency , another benchmark problem, the Smith-Hutton problem, is investigated. In this study, the scalar field is governed by the 2D convection diffusion equation, and driven by the following velocity field in the domain of $(x,y) = (-1 \sim 1, 0 \sim 1)$[55]:

$$\frac{\partial \phi}{\partial x} + v\frac{\partial \phi}{\partial y} - \frac{1}{Pe}\left(\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial x^y}\right) = 0 \tag{11a}$$

$$u = 2y(1-x^2) \tag{11b}$$

$$v = -2x(1-y^2) \tag{11c}$$

where the $Pe$ is the Pelect number and chosen as 1000 in this study. The top, left and right boundary are set as $\phi = 1 - tanh(10)$, and the bottom boundary is set as:

$$\phi = 1 + tanh(10(2x+1)), \quad x \leq 0 \tag{12a}$$

$$\frac{\partial \phi}{\partial y} = 0, \quad x > 0 \tag{12b}$$

The loss function for the current investigated problem is defined as follows:

$$\mathscr{L}_{DE} = u\frac{\partial \phi}{\partial x} + v\frac{\partial \phi}{\partial y} - \frac{1}{Pe}(\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial x^y}) \tag{13}$$

$$\mathscr{L}_{DBC} = \phi - (1 + tanh(10(2x+1))) \tag{14}$$

$$\mathscr{L}_{NBC} = \frac{\partial \phi}{\partial y} \tag{15}$$

in the above, $\mathscr{L}_{DE}$, $\mathscr{L}_{DBC}$ and $\mathscr{L}_{NBC}$ are the loss components for the governing equation, Dirichlet boundary condition and Neumann boundary condition. The weight for each component is chosen as 1 in this investigated problem.

To demonstrate the efficiency of the current method, the training is performed with conventional training and V cycle training procedure. for the conventional training, the uniform sample points of 6670 is employed for the training with 50,000 epochs, while for the V cycle method, the three level of sample points, 6670 ($D_1$),1770 ($D_2$)and 435 ($D_3$) is employed and trained with 10,000 epochs at each level, which lead to 50,000 epochs in total. For the first hidden layer, sine activation function is employed, while for other hidden-layers we choose tanh as activation function. The details of the training procedure for V cycle is tabulated in table.V.

| Level | Dataset | training epochs |
|---|---|---|
| 1 | $D_1$ | 1$\leq$ n $\leq$ 10,000 |
| 2 | $D_2$ | 10,001$\leq$ n $\leq$ 20,000 |
| 3 | $D_3$ | 20,001$\leq$ n $\leq$ 30,000 |
| 4 | $D_2$ | 30,001$\leq$ n $\leq$ 40,000 |
| 5 | $D_1$ | 40,001$\leq$ n $\leq$ 50,000 |

TABLE V: V cycle training sequence for Smith-Hutton problem

From Fig. 3 and Table VI, it can be seen that the current proposed method improve the accuracy of the training results. By comparing the the relative L2-norm with the CFD reference solutions as ground truth, the proposed method can improve the accuracy of 20%. The applicability and accuracy of the current method is confirmed for 2D Smith-Hutton problem.

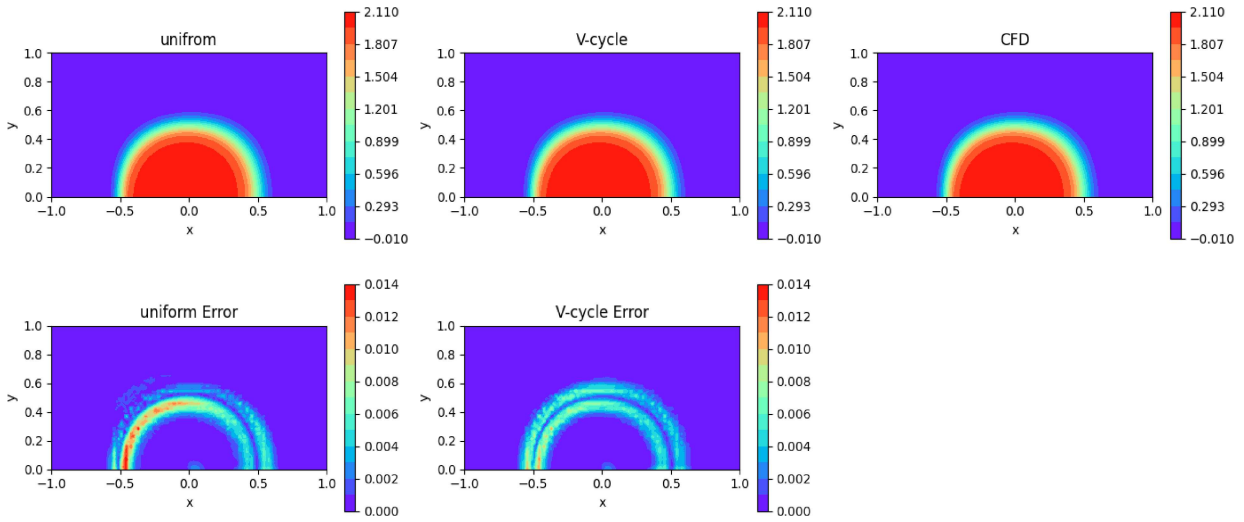



FIG. 3: Comparisons of training result (top) and absolute error (bottom) for different training settings.

| Model | $\mathscr{L}_{DE}$ | $\mathscr{L}_{DBC}$ | $\mathscr{L}_{NBC}$ | Test error(Rel. $L_2$) |
|---|---|---|---|---|
| uniform | 3.54e-06 | 6.33e-07 | 1.49e-08 | 2.54e-03 |
| V cycle | 5.10e-06 | 7.87e-09 | 3.70e-08 | 2.10e-03 |

TABLE VI: Loss values comparison after 50,000 epochs of training for the Smith-Hutton problem

### D. 2D steady Lid-driven cavity

Finally, we investigate the Lid-driven cavity problem, a classic benchmark used to assess the accuracy of methodologies in Computational Fluid Dynamics (CFD). In this study, the effect of grid sequence on model accuracy, evaluating Relative $L_2$ error(Rel. $L_2$) and Mean Absolute Error (MAE) against the reference solution by Ghia et al.[56] will be studied.

**Problem description:** Under the isothermal and incompressible assumptions, the flow field in the Lid-driven cavity is governed by the incompressible Navier-Stokes equations, i.e., momentum equations (denotes $NS_x$ and $NS_y$) and Continuity equation (denotes $Con$):

$$\begin{cases} f_{NS_x} = u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + \frac{1}{\rho}[\frac{\partial p}{\partial x} - \mu(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2})] \\ f_{NS_y} = u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + \frac{1}{\rho}[\frac{\partial p}{\partial y} - \mu(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2})] \\ f_{Con} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \end{cases} \tag{16}$$

With top boundary $U_{lid} = 1$, density $\rho = 1$ and domain $\Omega(x,y) \in [0,1] \times [0,1]$, the Reynolds number (Re) is then be defined as $Re = \frac{\rho U_{lid}}{\mu}$.

This is problem is well-known to be difficult when higher $Re$ is used. To model this challenging problem, Wang et al.[30] addressed gradient pathology issues and employing adaptive loss weights. Li et al.[57] proposed an adaptive loss weighting approach optimized through neural networks. Karniadakis et al.[58] introduced a two-network structure incorporating parameters for eddy viscosity and entropy equations. Jiang et al.[59] utilized finite difference methods for accurate gradient calculations and rapid convergence. Wang et al.[29] provided a comprehensive review, highlighting advancements such as Fourier features[35], Neural-Tangent-Kernel based Weighting[32], and curriculum training[28]. Chen et al.[60] proposed a citation block-based structure using Convolutional Neural Networks and sample weighting mechanisms. Lastly, Chiu et al.[11] proposed a gradient calculation method combining automatic and numerical differentiation. In this study, we will show the by using a simple multilevel idea can make the training feasible.

To better provide the training weight for each individual loss component, in this study we employ the dimensional-analyzed weighting scheme.

Beginning with dimensional analysis of model losses, each loss component can be expressed with dimensions as follows:

$$\begin{cases} \mathscr{L}_{NS_x} \sim (\mu(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}))^2 \sim \frac{[\mu]^2[U]^2}{[h]^4} \\ \mathscr{L}_{NS_y} \sim (\mu(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}))^2 \sim \frac{[\mu]^2[U]^2}{[h]^4} \\ \mathscr{L}_{Con} \sim (\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y})^2 \sim \frac{[U]^2}{[h]^2} \\ \mathscr{L}_{BC_u} \sim (u - g_u)^2 \sim [U]^2 \\ \mathscr{L}_{BC_v} \sim (v - g_v)^2 \sim [U]^2 \end{cases} \tag{17}$$

where $BC$ represents boundary condition in u- and v- components, and $h$ is the specific length to be defined.

To balance the losses, the corresponding weightings will have reciprocal dimensions of eq.(17)

after the equation eliminates repeating elements:

$$\begin{cases} \lambda_{NS_x} = \frac{[h]^4}{[\mu]^2} \\ \lambda_{NS_y} = \frac{[h]^4}{[\mu]^2} \\ \lambda_{Con} = [h]^2 \\ \lambda_{BC_u} = 1 \\ \lambda_{BC_v} = 1 \end{cases} \quad Rootscheme \rightarrow \begin{cases} \lambda_{NS_x} = \frac{[h]^2}{[\mu]} \\ \lambda_{NS_y} = \frac{[h]^2}{[\mu]} \\ \lambda_{Con} = [h] \\ \lambda_{BC_u} = 1 \\ \lambda_{BC_v} = 1 \end{cases} \tag{18}$$

In this study, $h$ is chosen as $\frac{1}{N}$ when a uniform $N \times N$ samples are employed, while in the random collocation scenario, $h$ is determined by the closest number to square root of the number of main collocations. For example, with 6565 main collocations, we can get $h = \frac{1}{Round(\sqrt{6565})} = \frac{1}{81}$.

Our neural network is structured into two sub-networks for predicting two distinct values, which this structure is inspired by Wong et al.[11,48] with a decent accuracy in original structure in Lid-driven cavity, as illustrated in the figure(Fig.4) and tabulated in Table. I, where $\hat{\psi}$ and $\hat{p}$ each utilize a sub-network with 2 layers and 20 units. For the first hidden layer, sine activation function is employed, while for other hidden-layers we choose Swish as activation function:

$$\sigma_l(x) = \begin{cases} sin(2\pi x), & l = 1 \\ Swish(x) = x * sigmoid(x), & l > 1 \end{cases} \tag{19}$$

To ease the training, the vorticity-velocity (VV) formulation[61] is also chosen:

$$\hat{u} = \frac{\partial \hat{\psi}}{\partial y}, \hat{v} = -\frac{\partial \hat{\psi}}{\partial x} \tag{20}$$

In every training conducted in this subsection, two convergence criteria have been employed: the Total Loss Criterion and the Loss Plateau Criterion. For the Total Loss Criterion, the following equation is employed:

$$f_i^{crit} = |log_{10}(\mathscr{L}_i^n) - log_{10}(\mathscr{L}_i^{n-1000})| \tag{21}$$

where $i$ is the loss components and $n$ is current epoch. For the Loss Plateau Criterion, we use the following equation (Eq. 22) to halt training if the losses remain unchanged for a specified duration

$$Crit_{Plat} := \frac{\sum_i^{N_{comp}} f_i^{crit}}{N_{comp}} \tag{22}$$

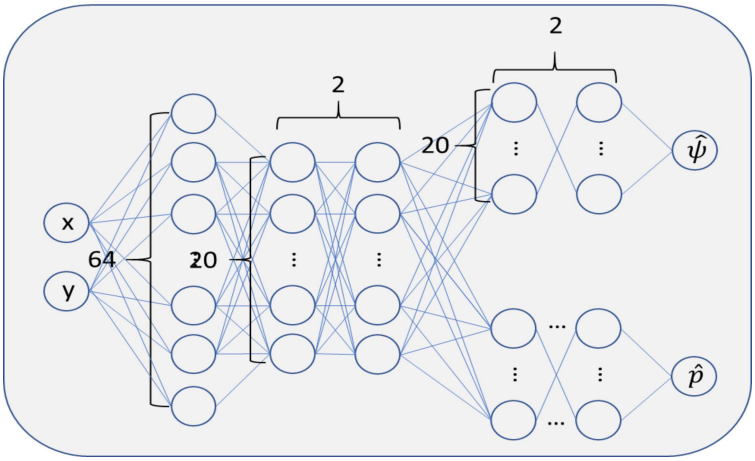



FIG. 4: NN structure for Lid-driven cavity

By default, the model stops training when either the total loss drops below $10^{-6}$ or $Crit_{Plat} \leq 10^{-4}$. If neither condition is met, training stops at the maximal epochs defined in Table.I.

Similar to previous experiments, several datasets are used in training. We aim to investigate the effectiveness of different dataset sequences and structures, as outlined in the Table.VII.

| Dataset | Collocations | Dataset | Collocations |
|---|---|---|---|
| $D_1, D_{1a}$(main) | 6565 | $D_{avg1}$ | 2900 |
| $D_2, D_{2a}$ | 1685 | $D_{avg2}$ | 2900 |
| $D_3$ | 445 | $D_{avg3}$ | 2900 |

TABLE VII: Datasets detail in Lid-driven cavity

Where $D_1 - D_3$(Random) are datasets with randomly distributed collocation points. $D_{1a}, D_{2a}$(Inherit) are datasets where the collocation points inherit coordinates from a preceding smaller dataset level. $D_{avg1} - D_{avg3}$(Average) are three datasets with the same number of collocations but different coordinates. Then, we perform tests with the following sequence and composition as shown in Table. VIII:

Table. IX presents our training results for Re=400 ($\mu = \frac{1}{400}$) using the default setting alongside a reference solution. In addition to the aforementioned datasets, we also compare the accuracy achieved with a uniform (81×81) mesh dataset which is obtained by finite difference method(FDM) with central difference scheme (CDS), and a single random dataset ($D_1$) to demonstrate the efficacy of our method. Across all tests, significant improvements in accuracy are observed, with the random V-cycle showing the largest improvement by reducing the error by 64%.

| Level | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| Duration(W-cycle) | 20,000 | 20,000 | 20,000 | 20,000 | 20,000 | 20,000 | 20,000 | 20,000 | To converge |
| Duration(V-cycle) | 20,000 | 20,000 | 20,000 | 20,000 | To converge | | | | |
| Duration(3-level) | 40,000 | 40,000 | To converge | | | | | | |
| Random, 3 levels | $D_3$ | $D_2$ | $D_1$ | | | | | | |
| Inherit, 3 levels | $D_3$ | $D_{2a}$ | $D_{1a}$ | | | | | | |
| Random, V-cycle | $D_1$ | $D_2$ | $D_3$ | $D_2$ | $D_1$ | | | | |
| Inherit, V-cycle | $D_{1a}$ | $D_{2a}$ | $D_3$ | $D_{2a}$ | $D_{1a}$ | | | | |
| Random, W-cycle | $D_1$ | $D_2$ | $D_3$ | $D_2$ | $D_1$ | $D_2$ | $D_3$ | $D_2$ | $D_1$ |
| Average, V-cycle | $D_{avg1}$ | $D_{avg2}$ | $D_{avg3}$ | $D_{avg2}$ | $D_{avg1}$ | | | | |

TABLE VIII: Test detail in Lid-driven cavity

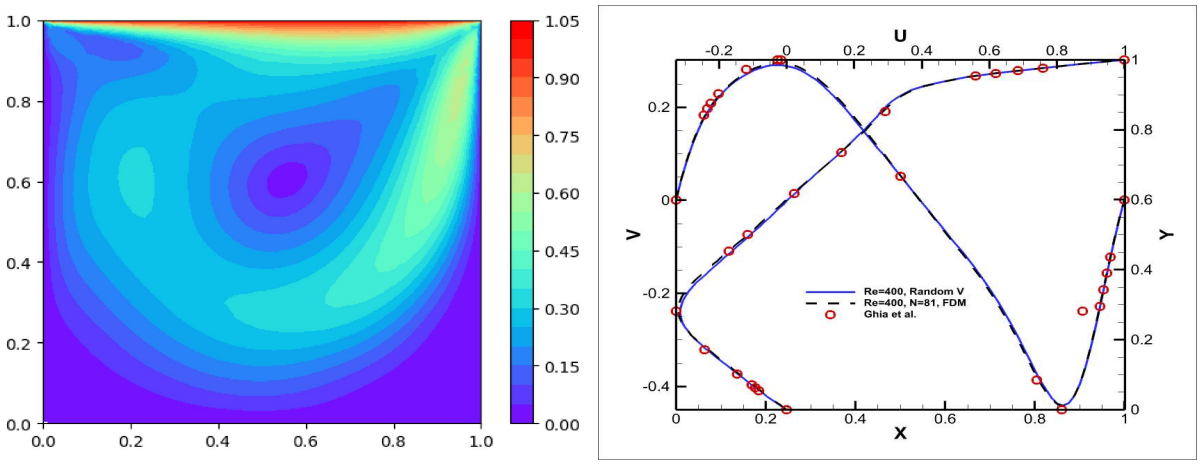
| Test | $Error_u$ (MAE) | $Error_v$ (MAE) | Training epochs | Error decrease |
|---|---|---|---|---|
| Uniform | 4.48e-02 | 5.05e-02 | 100,000 | |
| Random, 1 level | 1.75e-02 | 2.90e-02 | 104,000 | baseline |
| Random, 3 levels | 1.10e-02 | 2.40e-02 | 180,000 | -27.9% |
| Inherit, 3 levels | 8.3e-03 | 1.74e-02 | 182,000 | -46.3% |
| **Random, V-cycle** | **4.25E-03** | **1.38E-02** | **163,000** | **-64.1%** |
| Inherit, V-cycle | 8.69e-03 | 1.97E-02 | 163,000 | -41.2% |
| Random, W-cycle | 1.04E-02 | 2.12E-02 | 162,000 | -33.7% |
| Average, V-cycle | 7.91E-03 | 1.75E-02 | 184,000 | -47.2% |

$$\text{Error decrease} = \frac{1}{2} \times \left(\frac{Error_{u,model} - Error_{u,base}}{Error_{u,base}} + \frac{Error_{v,model} - Error_{v,base}}{Error_{v,base}}\right)$$

TABLE IX: Best Test Result MAE compare with reference solution, $Re = 400$

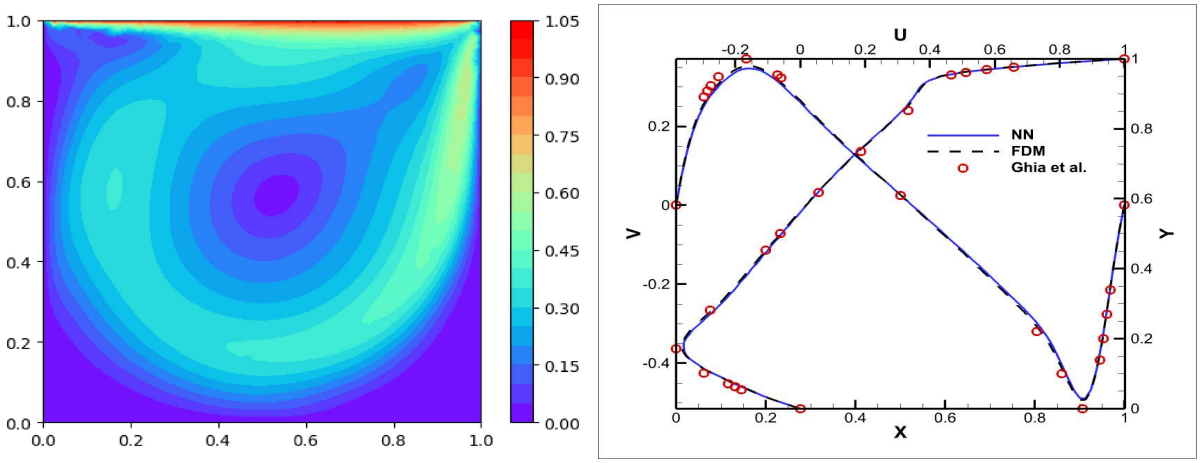
As the Reynolds number increases, the model becomes infeasible to converge. To address this, we scaled up our PINN model by making it four times wider with hidden layers. Specifically, the first layer now contains 256 neurons, while subsequent layers have 64 neurons each. This width was chosen considering computational time constraints and for benchmarking against other studies[29]. Without curriculum training, our model predicts Re=1000 with an accuracy of 5.85e-02 using a random V-cycle structure, as illustrated in Figure 6.

However, achieving an MAE accuracy under $5 \times 10^{-1}$ has proven significantly challenging in our model for $Re > 1000$ even for the larger neural network. To alleviate the training difficulty,




(a) (b)

FIG. 5: (a)Re=400 Model training result
(b)Re=400 test result(Blue solid line), FDM(81 × 81 CDS)(Black dashed line),
and Ghia et al solution(Red circle) after training for 163,000 epochs with Random, V-cycle




(a) (b)

FIG. 6: (a)Re=1000 Model training result
(b)Re=1000 test result(Blue solid line), FDM(81 × 81 CDS)(Black dashed line),
and Ghia et al solution(Red circle) after training for 212,000 epochs

we implemented a curriculum training approach for improving accuracy at higher *Re* values. In this study, we conducted predictions for $Re = 5000$ using a random V-cycle structure with denser sample points, as detailed in Table. X and XI. Additional training settings are provided in the Appendix(sec.V C). With the current approach, our model achieved a prediction accuracy of ap-

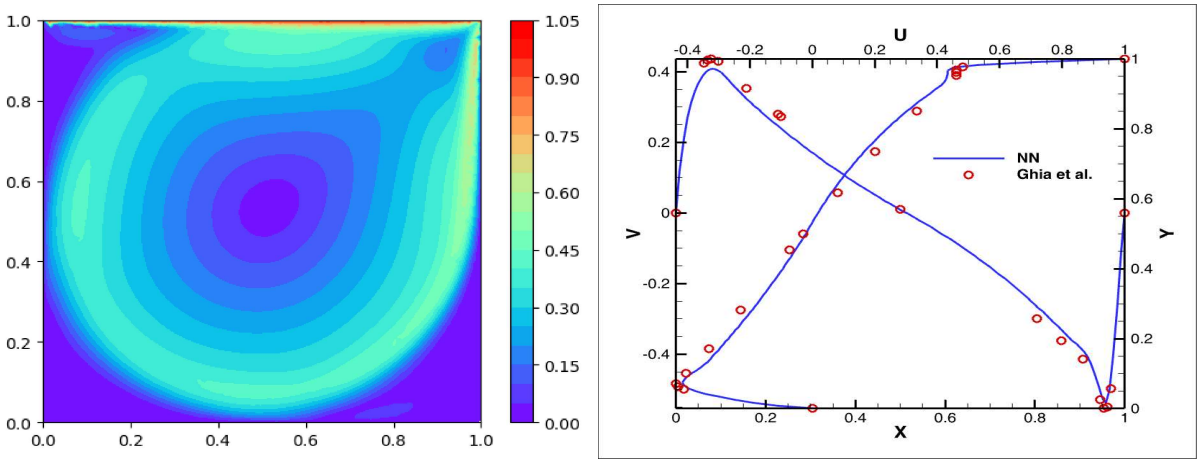
proximately $7 \times 10^{-2}$(Fig.7).

| Training sequence | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Reynolds number | 400 | 1000 | 2000 | 3200 | 5000 |

TABLE X: Curriculum training sequence in Lid-driven cavity

| Dataset | Collocations |
|---|---|
| $D_1$(main) | 14645 |
| $D_2$ | 3725 |
| $D_3$ | 965 |

TABLE XI: Datasets for $Re$ = 5000




(a) (b)

FIG. 7: (a)Re=5000 Model training result
(b)Re=5000 test result(Blue solid line), and Ghia et al solution(Red circle) after curriculum training for total around 1,207,000 epochs

Since the lid-driven cavity case is a well-documented benchmark in the literature, we further compare our results with selected papers from various perspectives to ensure our model achieves acceptable accuracy.

As shown in Table XII, achieving high accuracy in PINNs for high Reynolds numbers (Re > 1000) typically requires extensive training epochs and a large set of collocation points unless

| Paper | Re cases | Method[61] | Input | Training epochs | Collocation points | Accuracy for highest Re |
|---|---|---|---|---|---|---|
| Wang et al.[30] | [100] | DNS(V-V) | (x,y) | unknown | unknown | 3.42e-02 |
| Li et al[57] | [100] | DNS(V-P) | (x,y) | 32,000 | 13,000 | 6.70e-02 |
| Karniadakis et al.[58] | [2000,3000,5000] | DNS(V-P) | (x,y) | 3,000,000 | 120,000 | 7.00e-02 |
| Jiang et al.[59] | [100,1000,5000,10000] (1-100 samples) | DNS(V-P) **FD** | (x,y) | 200,000 | 40401 | 9.30e-03 (Re=1000, 50 samples) |
| Wang et al.[29] | [100,400,1000,3200] | DNS(V-P) | (x,y) | 700,000 | 8192 | 1.58e-01 |
| Chen et al.[60] | [100,300,600] | DNS(V-V) | (x,y) | 40,000 | 1280 | 7.25e-02 |
| Chiu et al.[11] | [400] | DNS(V-P) | (x,y) | 20,000 | 40401 | 3.16e-03 |
| Proposed model | [400, 1000, 2000, 3200, 5000] | DNS(V-V) | (x,y) | 1,207,000 | 19335 | 7.0e-02 |

In method we use abbreviations provided in Jin et al.[61], where **V-P** stands for velocity-pressure formulation and **V-V** stands for vorticity-velocity formulation. As **FD** shown in Jiang et al.[59] means using finite difference to calculate gradients instead of automatic differentiation. As for accuracy column, the accuracy is calculated in Relative L2.

TABLE XII: Test detail in Lid-driven cavity

ground truth data points are included in training loss. Comparing the results obtained by Karniadakis et al.[58], our model achieved an error of 7e-02, which is slightly better than the accuracy reported by Karniadakis et al.[58], while requiring significantly fewer epochs and collocation points. It is noted that as the Reynolds number increases, the rate of error reduction gradually decreases. Additionally, accurately predicting higher Reynolds numbers becomes more challenging due to the increased numerical stiffness of the governing equations, making precise adjustments to the learning rate increasingly critical. It is also noted that although training epochs may not directly reflect time costs, this comparison indicates that our method requires fewer parameter updates and achieves higher accuracy compared to the referenced paper. Importantly, our proposed method can complement theirs without interference in implementations.

## IV. CONCLUSIONS

In this paper, we have demonstrated the effectiveness of Physics-Informed Neural Networks (PINNs) with our proposed method, utilizing a multi-dataset training strategy and dimensional analysis weighting. We have successfully shown that employing multiple datasets, which including at least two sizes of dataset for transfer learning, enhances the neural network's capability to fit higher frequency functions more accurately. This approach aligns with similar approach validated in MscaleDNNs[39,45]. Additionally, we have highlighted the model's compatibility with curriculum training[28], comparing favorably against conventional models in solving high Reynolds number steady Lid-driven cavity problems.

While the current work is focus on fluid dynamic problem, the current method is generic and neural network independent. Our future investigations will focus on tackling more complex problems involving higher-dimensional domains and transient tasks. Furthermore, we are keen to conduct numerical analyses on training frequencies and investigate on the effectiveness within different collocation size relationship, which potentially further optimize the performance of our method.


## ACKNOWLEDGMENTS

Chao-An Lin would like to acknowledge support by Taiwan National Science and Technology Council under project No. NSTC 113-2221-E-007-129.


## DATA AVAILABILITY STATEMENT

The data supporting this study's findings are available from the corresponding author upon reasonable request.

## CONFLICT OF INTEREST

The authors have no conflicts to disclose.

## REFERENCES


[1]H. Lee and I. S. Kang, "Neural algorithm for solving differential equations," Journal of Computational Physics **91**, 110–131 (1990).

[2]A. Meade and A. Fernandez, "Solution of nonlinear ordinary differential equations by feedforward neural networks," Mathematical and Computer Modelling **20**, 19–44 (1994).

[3]M. Perdikaris P. Karniadakis G. E. Raissi, "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).

[4]M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics informed deep learning (part I): data-driven solutions of nonlinear partial differential equations," CoRR **abs/1711.10561** (2017), 1711.10561.

[5]M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics informed deep learning (part II): data-driven discovery of nonlinear partial differential equations," CoRR **abs/1711.10566** (2017), 1711.10566.

[6]A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, "Automatic differentiation in machine learning: a survey," Journal of Machine Learning Research **18**, 1–43 (2018).

[7]S. J. W. Suvrit Sra, Sebastian Nowozin, *Optimization for Machine Learning* (The MIT Press, 2011).

[8]Y. B. Ian Goodfellow and A. Courville, *Deep Learning* (The MIT Press, 2016).

[9]H. Jeong, J. Bai, C. Batuwatta-Gamage, C. Rathnayaka, Y. Zhou, and Y. Gu, "A physics-informed neural network-based topology optimization (pinnto) framework for structural optimization," Engineering Structures **278**, 115484 (2023).

[10]H. Kervadec, J. Bouchtiba, C. Desrosiers, E. Granger, J. Dolz, and I. Ben Ayed, "Boundary loss for highly unbalanced segmentation," Medical Image Analysis **67**, 101851 (2021).

[11]P.-H. Chiu, J. C. Wong, C. Ooi, M. H. Dao, and Y.-S. Ong, "Can-pinn: A fast physics-informed neural network based on coupled-automatic–numerical differentiation method," Computer Methods in Applied Mechanics and Engineering **395**, 114909 (2022).

[12]P. Zhang, Y. Hu, Y. Jin, S. Deng, X. Wu, and J. Chen, "A maxwell's equations based deep learning method for time domain electromagnetic simulations," IEEE Journal on Multiscale and Multiphysics Computational Techniques **6**, 35–40 (2021).

[13]P. Sharma, W. T. Chung, B. Akoush, and M. Ihme, "A review of physics-informed machine learning in fluid mechanics," Energies **16** (2023), 10.3390/en16052343.

[14]G. Pang, L. Lu, and G. E. Karniadakis, "fpinns: Fractional physics-informed neural networks," SIAM Journal on Scientific Computing **41**, A2603–A2626 (2019), https://doi.org/10.1137/18M1229845.

[15]Y.-T. Liao and C.-A. Lin, "Accelerating fractional step simulation with autoencoder," Master thesis, Department of Power Mechanical Engineering, National Tsing Hua University (2021).

[16]A. D. Jagtap, E. Kharazmi, and G. E. Karniadakis, "Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems," Computer Methods in Applied Mechanics and Engineering **365**, 113028 (2020).

[17]A. D. Jagtap and G. E. Karniadakis, "Extended physics-informed neural networks (xpinns): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations," Communications in Computational Physics **28** (2020).

[18]J. Sirignano, J. F. MacArt, and J. B. Freund, "Dpm: A deep learning pde augmentation method with application to large-eddy simulation," Journal of Computational Physics **423**, 109811 (2020).

[19]D. Kochkov, J. A. Smith, A. Alieva, Q. Wang, M. P. Brenner, and S. Hoyer, "Machine learning–accelerated computational fluid dynamics," Proceedings of the National Academy of Sciences **118**, e2101784118 (2021), https://www.pnas.org/doi/pdf/10.1073/pnas.2101784118.

[20]R. Maulik, O. San, A. Rasheed, and P. Vedula, "Subgrid modelling for two-dimensional turbulence using neural networks," Journal of Fluid Mechanics **858**, 122–144 (2019).

[21]Z. Shi, N. S. Gulgec, A. S. Berahas, S. N. Pakzad, and M. Takáč, "Finite difference neural networks: Fast prediction of partial differential equations," (2020), arXiv:2006.01892.

[22]N. Sukumar and A. Srivastava, "Exact imposition of boundary conditions with distance functions in physics-informed deep neural networks," Computer Methods in Applied Mechanics and Engineering **389**, 114333 (2022).

[23]J. Wang, Y. Mo, B. Izzuddin, and C.-W. Kim, "Exact dirichlet boundary physics-informed neural network epinn for solid mechanics," Computer Methods in Applied Mechanics and Engineering **414**, 116184 (2023).

[24]J. C. Wong, P.-H. Chiu, C. Ooi, M. H. Dao, and Y.-S. Ong, "Lsa-pinn: Linear boundary connectivity loss for solving pdes on complex geometry," *2023 International Joint Conference on Neural Networks (IJCNN)*, (2023), 10.1109/ijcnn54540.2023.10191236.

[25]S. Goswami, C. Anitescu, S. Chakraborty, and T. Rabczuk, "Transfer learning enhanced physics informed neural network for phase-field modeling of fracture," Theoretical and Applied Fracture Mechanics **106**, 102447 (2020).

[26]S. Chakraborty, "Transfer learning based multi-fidelity physics informed deep neural network," Journal of Computational Physics **426**, 109942 (2021).

[27]M. Penwarden, S. Zhe, A. Narayan, and R. M. Kirby, "A metalearning approach for physics-informed neural networks (pinns): Application to parameterized pdes," Journal of Computational Physics **477**, 111912 (2023).

[28]A. Krishnapriyan, A. Gholami, S. Zhe, R. Kirby, and M. W. Mahoney, "Characterizing possible failure modes in physics-informed neural networks," Advances in Neural Information Processing Systems **34**, 26548–26560 (2021).

[29]S. Wang, S. Sankaran, H. Wang, and P. Perdikaris, "An expert's guide to training physics-informed neural networks," (2023), arXiv:2308.08468.

[30]S. Wang, Y. Teng, and P. Perdikaris, "Understanding and mitigating gradient flow pathologies in physics-informed neural networks," SIAM Journal on Scientific Computing **43**, A3055–A3081 (2021), https://doi.org/10.1137/20M1318043.

[31]N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. Hamprecht, Y. Bengio, and A. Courville, "On the spectral bias of neural networks," Proceedings of the 36th International Conference on Machine Learning (2019).

[32]S. Wang, X. Yu, and P. Perdikaris, "When and why pinns fail to train: A neural tangent kernel perspective," Journal of Computational Physics **449**, 110768 (2022).

[33]A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," (2018), arXiv:1705.07115 [cs.CV].

[34]Z. Xiang, W. Peng, X. Liu, and W. Yao, "Self-adaptive loss balanced physics-informed neural networks," Neurocomputing **496**, 11–34 (2022).

[35]M. Tancik, P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. Barron, and R. Ng, "Fourier features let networks learn high frequency functions in low dimensional domains," Advances in neural information processing systems **33**, 7537–7547 (2020).

[36]S. Wang, H. Wang, and P. Perdikaris, "On the eigenvector bias of fourier feature networks: From regression to solving multi-scale pdes with physics-informed neural networks," Computer Methods in Applied Mechanics and Engineering **384**, 113938 (2021).

[37]A. Jacot, F. Gabriel, and C. Hongler, "Neural tangent kernel: Convergence and generalization in neural networks," Advances in neural information processing systems **31** (2018).

[38]R. Mojgani, M. Balajewicz, and P. Hassanzadeh, "Kolmogorov n–width and lagrangian physics-informed neural networks: A causality-conforming manifold for convection-dominated pdes," Computer Methods in Applied Mechanics and Engineering **404**, 115810 (2023).

[39]Z. Liu, "Multi-scale deep neural network (mscalednn) for solving poisson-boltzmann equation in complex domains," Communications in Computational Physics **28**, 1970–2001 (2020).

[40]R. Wienands and W. Joppich, *Practical Fourier Analysis for Multigrid Methods*, Numerical Insights (Taylor & Francis, 2005).

[41]N. Margenberg, D. Hartmann, C. Lessig, and T. Richter, "A neural network multigrid solver for the navier-stokes equations," Journal of Computational Physics **460**, 110983 (2022).

[42]N. Margenberg, R. Jendersie, C. Lessig, and T. Richter, "Dnn-mg: A hybrid neural network/finite element method with applications to 3d simulations of the navier–stokes equations," Computer Methods in Applied Mechanics and Engineering **420**, 116692 (2024).

[43]M. A. Nabian, R. J. Gladstone, and H. Meidani, "Efficient training of physics-informed neural networks via importance sampling," Computer-Aided Civil and Infrastructure Engineering **36**, 962–977 (2021), https://onlinelibrary.wiley.com/doi/pdf/10.1111/mice.12685.

[44]C. Wu, M. Zhu, Q. Tan, Y. Kartha, and L. Lu, "A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks," Computer Methods in Applied Mechanics and Engineering **403**, 115671 (2023).

[45]E. Riccietti, V. Mercier, S. Gratton, and P. Boudier, "Multilevel physics informed neural networks (MPINNs)," (2022).

[46]S. J. Pan and Q. Yang, "A survey on transfer learning," IEEE Transactions on Knowledge and Data Engineering **22**, 1345–1359 (2010).

[47]L. Torrey and J. Shavlik, "Transfer learning," *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, , 242–264 (2010).

[48]J. Cheng Wong, C. Ooi, A. Gupta, and Y.-S. Ong, "Learning in sinusoidal spaces with physics-informed neural networks," IEEE Transactions on Artificial Intelligence , 1–15 (2022).

[49]P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," arXiv preprint arXiv:1710.05941 (2017).

[50]K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," *2015 IEEE International Conference on Computer Vision*

*(ICCV)*, , 1026–1034 (2015).

[51]V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein, "Implicit neural representations with periodic activation functions," *Advances in Neural Information Processing Systems*, **33**, 7462–7473 (2020).

[52]A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, and L. Antiga, "Automatic differentiation in pytorch." In NIPS 2017 Autodi Workshop: The Future of Gradient-based Machine Learning Software and Techniques (2017).

[53]D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," (2017), arXiv:1412.6980 [cs.LG].

[54]Y.-E. Chou and C.-A. Lin, "Impact of loss weight and model complexity on physics-informed neural networks for computational fluid dynamics," Master thesis, Department of Power Mechanical Engineering, National Tsing Hua University (2023).

[55]R. Smith and A. Hutton, "The numerical treatment of advection: A performance comparison," Numerical Heat Transfer **5**, 439–46 (1982).

[56]U. Ghia, K. Ghia, and C. Shin, "High-re solutions for incompressible flow using the navier-stokes equations and a multigrid method," Journal of Computational Physics **48**, 387–411 (1982).

[57]S. Li and X. Feng, "Dynamic weight strategy of physics-informed neural networks for the 2d navier–stokes equations," Entropy **24**, 1254 (2022).

[58]G. Karniadakis, Z. Wang, and X. Meng, "Solution multiplicity and effects of data and eddy viscosity on navier-stokes solutions inferred by physics-informed neural networks," Bulletin of the American Physical Society (2023).

[59]Q. Jiang, C. Shu, L. Zhu, L. Yang, Y. Liu, and Z. Zhang, "Applications of finite difference-based physics-informed neural networks to steady incompressible isothermal and thermal flows," International Journal for Numerical Methods in Fluids **95**, 1565–1597 (2023).

[60]X. Chen, R. Chen, Q. Wan, R. Xu, and J. Liu, "An improved data-free surrogate model for solving partial differential equations using deep neural networks," Scientific reports **11**, 19507 (2021).

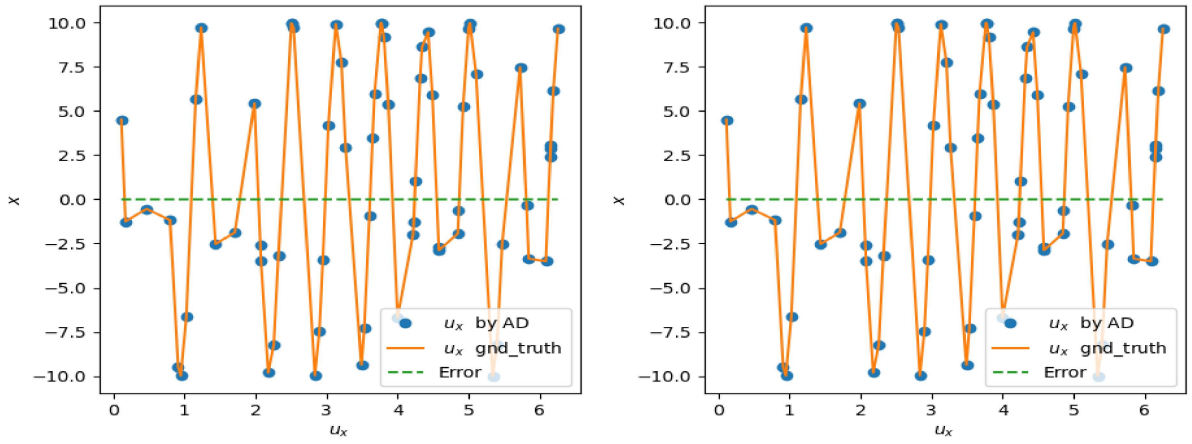
[61]X. Jin, S. Cai, H. Li, and G. E. Karniadakis, "Nsfnets (navier-stokes flow nets): Physics-informed neural networks for the incompressible navier-stokes equations," Journal of Computational Physics **426**, 109951 (2021).

# V. APPENDIX: DETAIL SETTING OF EXPERIMENTS

## A. 1D high frequency ODE

| Dataset | $D_1$ | $D_2$ |
|---|---|---|
| Collocation points | 64 | 32 |
| Optimizer | Adam(lr=1e-03) | Adam(lr=1e-03) |
| Scheduler (ReduceLROnPlateau) | factor=0.65, patience=100, min_lr=1e-06 | factor=0.8, patience=100, min_lr=1e-06 |
| Loss weighting ($\lambda_{DE}$:$\lambda_{BC}$) | 1:1 | 1:1 |

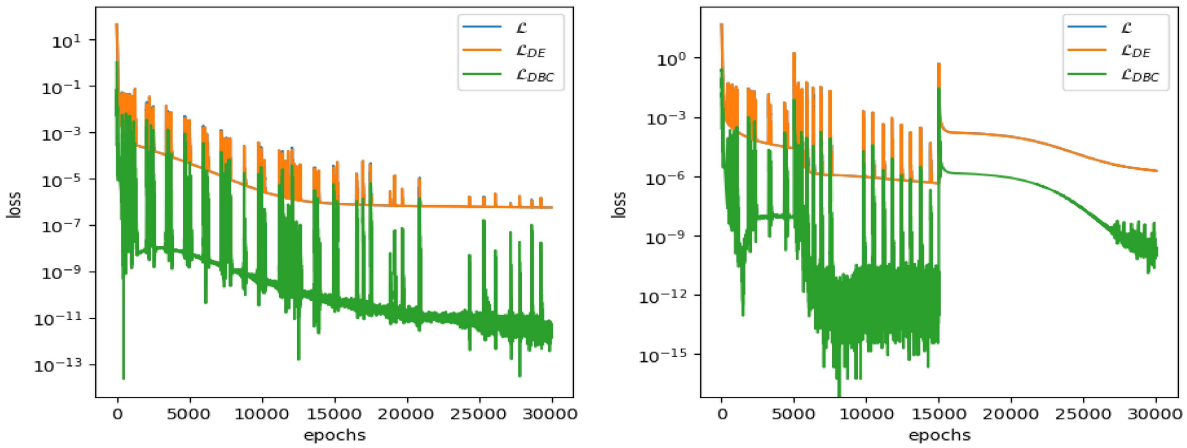
TABLE XIII: 1D high frequency ODE detail setup of fig.1




(a) PINN w/o proposed method

(b) PINN w/ proposed method

FIG. 8: Gradients value graphs after training for 30,000 epochs




(a) PINN w/o proposed method

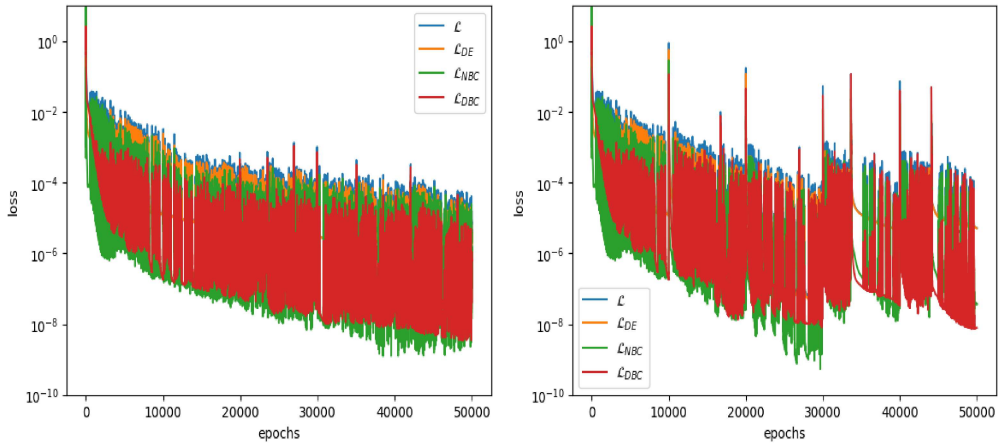
(b) PINN w/ proposed method

FIG. 9: Loss history graphs after training for 30,000 epochs

## B. 2D Smith-Hutton problem

| Dataset | $D_1$ | $D_2$ | $D_3$ |
|---|---|---|---|
| Collocation points | 6670 | 1770 | 435 |
| Optimizer | Adam(lr=5e-03) | Adam(lr=5e-03) | Adam(lr=5e-03) |
| Scheduler (ReduceLROnPlateau) | factor=0.8, patience=1000, min_lr=1e-10 | factor=0.8, patience=1000, min_lr=1e-10 | factor=0.8, patience=1000, min_lr=1e-10 |
| Loss weighting ($\lambda_{DE}$:$\lambda_{NBC}$:$\lambda_{DBC}$) | 1:1:1 | 1:1:1 | 1:1:1 |

TABLE XIV: 2D Smith-Hutton problem detail setup of fig.3



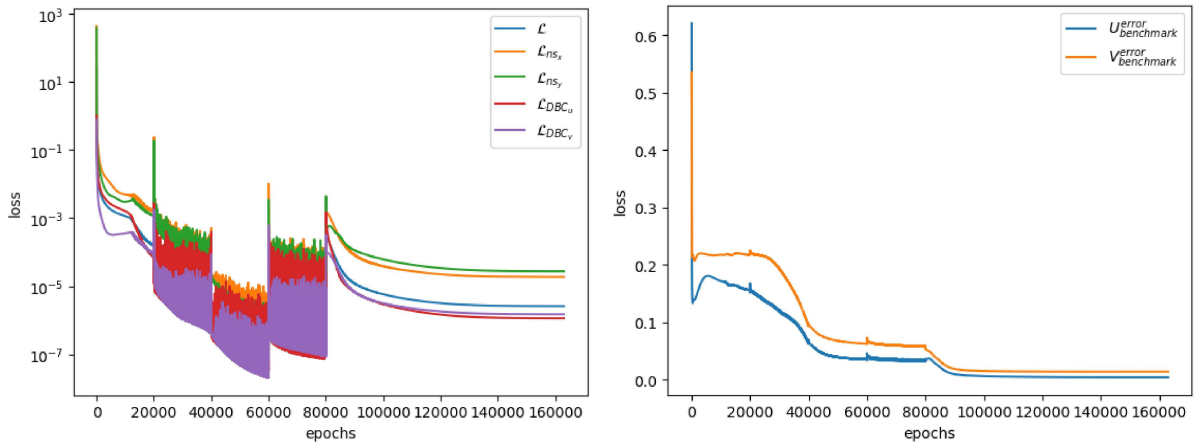

(a) uniform (b) V cycle

FIG. 10: plot of training history for 2D Smith-Hutton problem with different training settings.

## C. 2D steady Lid-driven cavity

| Dataset | $D_1$ | $D_2$ | $D_3$ |
|---|---|---|---|
| Collocation points | 6565 | 1685 | 445 |
| Optimizer | Adam(lr=1e-03) | Adam(lr=1e-03) | Adam(lr=1e-03) |
| Scheduler (StepLR) | factor=0.89, step_size=1000 | factor=0.93, step_size=1000 | factor=0.97, step_size=1000 |
| Loss weighting ($\lambda_{DE}$:$\lambda_{BC}$) | $DW_{root}$(eq.18) | $DW_{root}$ | $DW_{root}$ |

TABLE XV: 2D Lid-driven cavity detail setup of Re = 400(fig.5)

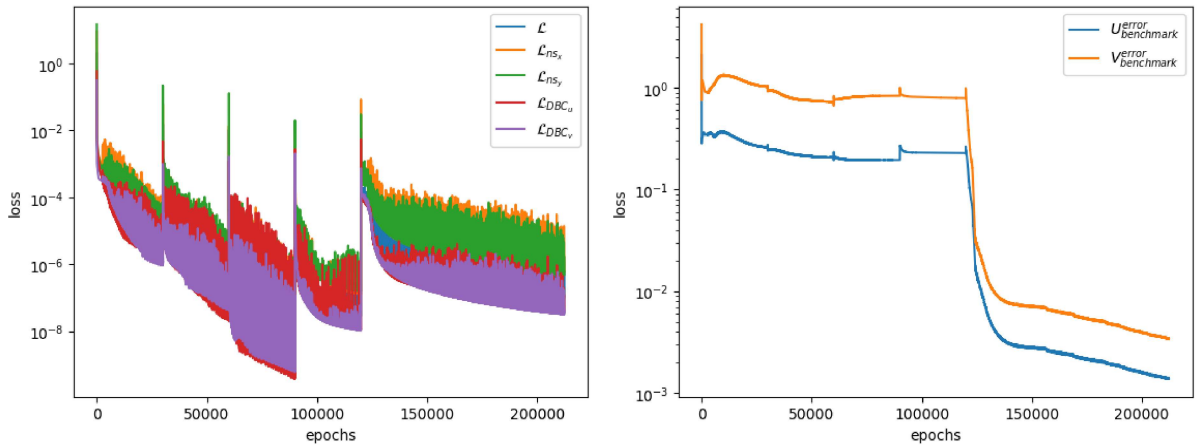



(a) (b)

FIG. 11: (a)Loss history and (b)Error history for Re=400 (fig.5)

| Dataset | $D_1$ | $D_2$ | $D_3$ |
|---|---|---|---|
| Collocation points | 6565 | 1685 | 445 |
| Optimizer | Adam(lr=1e-03) | Adam(lr=1e-03) | Adam(lr=1e-03) |
| Scheduler (ReduceLROnPlateau) | factor=0.78, patience=1000, min_lr=1e-08 | factor=0.82, patience=1000, min_lr=1e-08 | factor=0.90, patience=1000, min_lr=1e-08 |
| Loss weighting ($\lambda_{DE}$:$\lambda_{BC}$) | $DW_{root}$(eq.18) | $DW_{root}$ | $DW_{root}$ |

TABLE XVI: 2D Lid-driven cavity detail setup of Re = 1000(fig.6)




(a) (b)

FIG. 12: (a)Loss history and (b)Error history for Re=1000 (fig.6)

| Reynolds number | 400 | 1000 | 2000 | 3200 | 5000 |
|---|---|---|---|---|---|
| Level Duration | 20,000 | 20,000 | 20,000 | 30,000 | 30,000 |
| Initial LR | 1e-3 | 1e-3 | 1e-4 | 1e-4 | 4e-5/4e-5/1e-4 |
| Scheduler factor (ReduceLROnPlateau) | 0.73/0.80/0.88 | 0.68/0.75/0.83 | 0.63/0.70/0.81 | 0.58/0.65/0.79 | 0.58/0.6/0.80 |
| Loss weighting ($\lambda_{DE}$:$\lambda_{BC}$) | $DW_{root}$(eq.18) | $DW_{root}$ | $DW_{root}$ | $DW_{root}$ | $DW_{root}$ |
| Total loss criterion | 1e-07 | 5e-08 | 5e-07 | 5e-08 | 1e-08 |
| Error value (u/v) | 1.30e-02/1.43e-01 | 1.36e-02/2.3e-02 | no reference values | 5.75e-02/6.69e-02 | 5.86e-02/6.45e-02 |
| Training epochs | 111,297 | 143,225 | 82,016 | 370,844 | 500,000 |
| Total epochs | 111,297 | 254,522 | 336,538 | 707,382 | 1,207,382 |

The setting is similar to fig.7 and Table.XVI

TABLE XVII: 2D Lid-driven cavity curriculum detail setup of Re = 5000 (Table. XI)