Computer Methods in Applied Mechanics and Engineering 428 (2024) 117075

Contents lists available at ScienceDirect

Comput. Methods Appl. Mech. Engrg.

journal homepage: www.elsevier.com/locate/cma

Data-driven physics-informed neural networks: A digital twin perspective

Sunwoong Yang [a], Hojin Kim [b], Yoonpyo Hong [c,1], Kwanjung Yee [b], Romit Maulik [d], Namwoo Kang [a,*]

[a] Cho Chun Shik Graduate School of Mobility, Korea Advanced Institute of Science and Technology, Daejeon 34051, Republic of Korea
[b] Department of Aerospace Engineering, Seoul National University, Seoul 08826, Republic of Korea
[c] Institute of Advanced Machines and Design, Seoul National University, Seoul 08826, Republic of Korea
[d] College of Information Sciences and Technology, The Pennsylvania State University, PA 16802, USA

ARTICLE INFO

Keywords:
Digital twins
Physics-informed neural networks
Adaptive sampling
Data-driven approach
Multi-fidelity modeling
Uncertainty quantification

ABSTRACT

This study explores the potential of physics-informed neural networks (PINNs) for the realization of digital twins (DT) from various perspectives. First, various adaptive sampling approaches for collocation points are investigated to verify their effectiveness in the mesh-free framework of PINNs, which allows automated construction of virtual representation without manual mesh generation. Then, the overall performance of the data-driven PINNs (DD-PINNs) framework is examined, which can utilize the acquired datasets in DT scenarios. Its scalability to more general physics is validated within parametric Navier–Stokes equations, where PINNs do not need to be retrained as the Reynolds number varies. In addition, since datasets can be often collected from different fidelity/sparsity in practice, multi-fidelity DD-PINNs are also proposed and evaluated. They show remarkable prediction performance even in the extrapolation tasks, with 42 ∼ 62% improvement over the single-fidelity approach. Finally, the uncertainty quantification performance of multi-fidelity DD-PINNs is investigated by the ensemble method to verify their potential in DT, where an accurate measure of predictive uncertainty is critical. The DD-PINN frameworks explored in this study are found to be more suitable for DT scenarios than traditional PINNs from the above perspectives, bringing engineers one step closer to seamless DT realization.

1. Introduction

We are entering an era of digital twins (DT), where computational models are integrated with a physical space, creating a system that is dynamically updated through bidirectional interaction between the physical and virtual spaces (Fig. 1) [1–6]. Due to its strong potential in decision-making during the design and development process, DT has been adopted by major companies such as IBM, GE, and Siemens [7]. Especially, its ability to provide real-time guidance plays a crucial role in integrating the physical and digital worlds seamlessly [8–11]. In this context, physics-informed neural networks (PINNs), novel paradigms for solving partial differential equations (PDEs) leveraging deep learning framework [12], have been attracting engineers interested in alternatives

* Corresponding author.
E-mail addresses: sunwoongy@kaist.ac.kr (S. Yang), rhrhak96@snu.ac.kr (H. Kim), hyp1227@snu.ac.kr (Y. Hong), kjyee@snu.ac.kr (K. Yee), rmaulik@psu.edu (R. Maulik), nwkang@kaist.ac.kr (N. Kang).
[1] Current position: Helicopter Department, Institute of Aerodynamics and Flow Technology, German Aerospace Center (DLR, Deutsches Zentrum für Luft- und Raumfahrt).

https://doi.org/10.1016/j.cma.2024.117075
Received 5 January 2024; Received in revised form 16 May 2024; Accepted 17 May 2024
Available online 4 June 2024

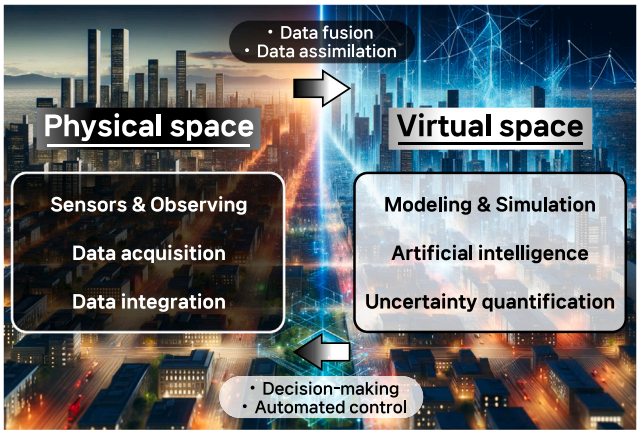


Fig. 1. Ecosystem of the digital twins. This image is generated largely based on the report by National Academies [6], while some parts have been modified. The background image is generated with the assistance of DALL·E 2, an AI model developed by OpenAI.

to resource-intensive numerical simulations. While conventional surrogate models such as Gaussian process regression specialize in scalar output predictions, PINNs can predict high-dimensional flow fields efficiently, improving scalability and efficiency in complex simulations [13,14]. Given that PINNs can replace the demanding physical space of DT with real-time virtual representation processes, their potential seems boundless.

PINNs aim to provide approximate solutions to PDEs at each spatial/temporal collocation point utilizing the architecture of neural networks (NNs). They can be realized by updating model parameters of NNs in both supervised/unsupervised manner: the initial or boundary conditions are leveraged as labels of each point and PDEs residuals are utilized as unsupervised loss functions. These simple but intuitive approaches have advantages over traditional numerical simulations, especially in the field of computational fluid dynamics (CFD) where the following long-standing challenges exist. First, PINNs are theoretically mesh-free techniques. This is because the PDEs are solved directly at each collocation point and there is no need to exploit the connectivity between their points. Their property eliminates the need for expensive mesh generation, which has been a tedious and cumbersome procedure for fluid engineers, and therefore allows PINNs to construct virtual space of the DT without human intervention due to manual mesh generation [15,16]. In addition, once a PINN is trained, the prediction of the flow field based on CFD simulations can be replaced by the real-time inference using the trained model [17]. Moreover, considering that Sun et al. [18] successfully validated PINNs can also solve parametric Navier–Stokes (NS) equations, their deployment can be extended to DT scenarios where the modeling capacity for the general physical system is required. Lastly, the use of PINNs can take advantage of recent developments in GPUs, and it is becoming much easier to implement their algorithms and variants thanks to numerous open-source deep learning libraries such as DeepXDE and Modulus [17,19–21].

However, conventional PINNs without labeled data (data-free PINNs) still have limitations to be applied as predictive models for the realization of DT, especially in the field of fluid dynamics. Although there are numerous previous studies using PINNs in flow problems, most of them are based on the low Reynolds numbers (Re), which limits further investigation of their potential in real-world problems at higher Re. For example, in lid-driven cavity flow, Amalinadhi et al. [22], Jagtap et al. [23], Li and Feng [24], Bai et al. [25], Wang et al. [26] showed that PINNs can work successfully in the forward problem at $Re = 100$. Also, Wong et al. [27] and Chiu et al. [28] tested PINNs at $Re = 400$ in forward problems. However, Krishnapriyan et al. [29] found that PINNs work for simple cases in 1D convection and reaction–diffusion problems, but fail when convection or diffusion coefficients become slightly higher, raising alarms about naive cut-and-paste use of PINNs. In a similar context, Chuang and Barba [30] discovered that while the data-free approach could accurately capture the flow fields around the cylinder at $Re = 40$, it failed at a bit higher Re (200) where vortex shedding exists: the similar failure even at $Re = 100$ was also confirmed by Rohrhofer et al. [31]. From the failures of PINNs that occurred slightly beyond the boundary of simple pedagogical flow problems, additional techniques to mitigate their failures seem necessary.

In this respect, there has been a recent movement towards the use of labeled data for the training of PINNs. Rohrhofer et al. [31] proposed data-driven PINNs (DD-PINNs) using labeled dataset as a training guide and successfully captured the cylinder vortex shedding at $Re = 100$. However, they simply used a high-fidelity CFD dataset, which conflicts with the purpose of training PINNs to replace CFD. Also, they claimed that consideration of this guide dataset leads to tricky optimization problems in terms of the loss landscape. Conversely, Gopakumar et al. [32] argued that DD-PINNs make landscapes have more pronounced valleys, which helps the optimizer converge to an optimal point more easily. In addition, they suggest a more pragmatic scenario for using DD-PINNs, where sparse guide data is generated from the low-fidelity simulation. Nevertheless, they still limited their scope to using only sparse guide data (from 1% to 10% grid points of the entire flow fields obtained by CFD), which can be considered as an inefficient approach as it does not fully exploit the obtained CFD results. Furthermore, they mainly focused on the change in the loss landscape and concluded that the incorporation of guide data helps the optimizer to find an optimal point more easily. However, in their case study of predicting the flow field around a rectangular block, DD-PINN only detected the presence of objects but failed to predict

the detailed flow structure. Finally, Chuang and Barba [30] observed that the DD-PINNs were able to predict the vortex shedding at $Re = 200$, where data-free PINNs failed. In their study, nevertheless, the DD-PINNs exhibited poor predictive performance in extrapolation when applied to non-parametric Navier–Stokes (NS) equations where the Reynolds number is fixed. This observation has necessitated a more in-depth exploration of the extrapolation capability of DD-PINNs, particularly in the context of solving parametric NS equations with different Reynolds numbers.

The objective of this research is to explore the potential of DD-PINNs from the perspective of DT. First, by thoroughly investigating the effectiveness of different adaptive sampling techniques on data-free PINNs, we aim to further consolidate their mesh-free advantage, which promotes automatic interaction between the physical and virtual spaces of the DT without human intervention. Second, given that the DT environment is continuously updated with collected data [6], this study scrutinizes a DD-PINN framework that can fully exploit the dataset assumed to be obtained from the numerical simulation. We then apply it to the higher Re flow problems where the data-free PINNs fail, aiming to bridge the existing gap between low Reynolds toy problems for academic validation and the higher Reynolds problems often encountered in the real-world. However, in practice, there exists a challenge in incorporating obtained data into the predictive models: datasets can be obtained from different sources [33,34]. Thus, we also extend the DD-PINN framework to the multi-fidelity approach, which can efficiently leverage datasets from different sources – even if some of the datasets are assumed to be sparsely collected from physical space – and thus significantly improve its scalability and flexibility. To validate multi-fidelity DD-PINNs in DT scenarios, the forward problem with parametric NS equations is adopted to verify whether they can be used for real-time prediction of general physics under different Reynolds numbers. Furthermore, we attempt to investigate their applicability to the uncertainty quantification (UQ) procedure, which is also a crucial component for the realization of the DT [6]. The main contributions of this paper, paired with its main research questions, are summarized as follows:

1. ***Is the construction of virtual space automated effectively?*** [Sections 2.2 and 3.2]
    - Given that the meshless nature of PINNs is a key ingredient for their automated construction, we aim to investigate the practical effectiveness of their mesh-free property. We propose an adaptive sampling method for collocation points that is specifically tailored to fluid dynamics and comprehensively compare it with existing sampling approaches.
2. ***Is the virtual space ready for data-driven model updating?*** [Section 5]
    - We extensively investigate the performance of DD-PINNs that can fully exploit the guide data set repeatedly acquired in the DT scenario. Then, moving away from the toy problem at low Reynolds numbers, we verify the superiority of the DD-PINNs in higher Reynolds conditions where conventional data-free PINNs fail. Furthermore, by visualizing the loss landscape, the reason for the success of DD-PINN over a data-free approach is demonstrated. See Section 5 for the details.
3. ***Is there any guideline for DD-PINN in DT scenarios?*** [Section 5.2]
    - We discover that random sampling outperforms other adaptive sampling techniques in DD-PINNs. We also conclude that this is because they require uniformly distributed collocation points to compensate for the local regularization effects of the data-driven loss term.
4. ***Is it scalable to general physics with different PDE parameters?*** [Section 6]
    - To validate the scalability of DD-PINNs, we apply them to parametric NS equations and show their advantage over data-free PINNs and data-driven NNs: it highlights the versatility of the proposed DD-PINN framework in DT, where real-time prediction under different PDE parameters is required.
5. ***Is it applicable to datasets of different fidelity/sparsity?*** [Section 6.2]
    - We further extend DD-PINNs to leverage heterogeneous data, taking into account the fact that in physical space the dataset is often collected from different sources. Then, their superiority over data-free PINNs, data-driven NNs, and single-fidelity DD-PINNs is effectively verified. Corresponding multi-fidelity DD-PINNs also show remarkable extrapolation performance even beyond the Reynolds numbers used for the train. It highlights the flexibility of the proposed DD-PINNs when different types of data are given, such as some sparse but high-fidelity data from physical space and others from virtual space.
6. ***Is it able to quantify reasonable predictive uncertainty?*** [Section 6.3]
    - The UQ performance of multi-fidelity DD-PINNs is also evaluated qualitatively by an ensemble approach. Since they are judged to provide reasonable uncertainty according to different Reynolds numbers, their potential in DT scenarios, where an accurate measure of predictive uncertainty is crucial, seems promising.

The rest of this paper is organized as follows. In Section 2, the background on data-free and data-driven PINNs is described along with the adaptive sampling techniques for their efficient training. In Section 3, the effects of adaptive sampling techniques on data-free PINNs in predicting lid-driven flow at $Re = 100$ are investigated. In Section 4, the failure of the data-free PINNs at higher Reynolds numbers is observed, and in Section 5, DD-PINNs are proposed as a remedy and the reason for their successful training is discussed. In Section 6, DD-PINNs are extended to parametric NS equations and further improved with multi-fidelity approaches. Section 7 presents the conclusions and future work of this study.

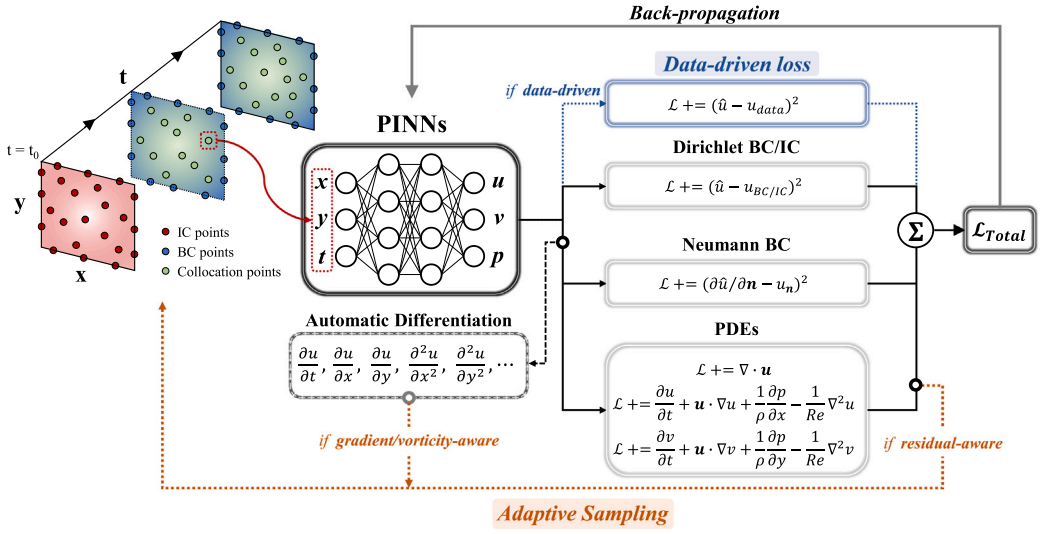


Fig. 2. Overall architectures of PINN. The data-free PINN does not include data-driven loss term whereas data-driven PINN does.

2. Methodologies

2.1. Physics-informed neural networks (PINNs)

PINN, whose algorithm is to constrain NNs by incorporating PDEs as loss functions, is one of the most fascinating concepts for engineers of all disciplines. Its goal is to solve PDEs using NNs, and therefore the loss function of NNs includes PDE losses, which are computed as zero when the PDEs are exactly satisfied. In most cases, PDEs are usually accompanied by boundary conditions (BC) or initial conditions (IC) that make their problem closed and have a unique solution. The corresponding PINN framework is called data-free PINN in that it does not require any labeled data such as velocity fields at specific spatial/temporal locations. There is another PINN framework called data-driven PINN (DD-PINN) [30], which has an additional data-driven loss term. This term takes into account the deviation from the labeled data and the predicted values of PINNs, driving them to predict the specific values (output labels) at specific locations (input labels). Data-free and DD-PINNs are explained in more detail in Sections 2.1.1 and 2.1.2, respectively.

2.1.1. Data-free PINNs

As in Fig. 2, the input to the PINN consists of spatial (x, y for 2D) and temporal (t) coordinates: it has different input and output structures than conventional convolutional neural network-based frameworks, which construct entire flow fields from a single input set at a time [35–37]. PINN outputs the quantities of interest at the corresponding coordinates, and in this study, x-velocity (u), y-velocity (v), and pressure (p) are adopted as output. Based on them, PDE and BC/IC losses can be defined. Since the calculation of these loss terms does not require the labeled dataset, it is referred to as data-free PINN. For incompressible NS equations, PDE losses for PINN are defined as follows:

$$\begin{aligned} \mathcal{L}_{mass} &= \|\nabla \cdot \mathbf{u}\| \\ \mathcal{L}_{x-momentum} &= \|\frac{\partial u}{\partial t} + \mathbf{u} \cdot \nabla u + \frac{1}{\rho}\frac{\partial p}{\partial x} - \frac{1}{Re}\nabla^2 u\| \\ \mathcal{L}_{y-momentum} &= \|\frac{\partial v}{\partial t} + \mathbf{u} \cdot \nabla v + \frac{1}{\rho}\frac{\partial p}{\partial y} - \frac{1}{Re}\nabla^2 v\| \\ & x, y \in \Omega, t \in [0, T] \end{aligned} \tag{1}$$

where $\mathbf{u}$ is velocity vector consists of u, v; and ρ, Re each denotes density and Reynolds number. However, since PDEs alone cannot represent the unique solution of the system, additional information about domain boundaries is needed [18]. BCs and ICs are considered to this end, and assume they are given as follows:

$$\begin{cases} \mathcal{NN}(x, y, t) = \mathcal{F}_{DirichletBC}, & [x, y] \in \Gamma_{DirichletBC}, \ t \in [0, T] \\ \frac{\partial \mathcal{NN}(x,y,t)}{\partial n} = \mathcal{F}_{NeumannBC}, & [x, y] \in \Gamma_{NeumannBC}, \ t \in [0, T] \\ \mathcal{NN}(x, y, 0) = \mathcal{G}_{IC}, & [x, y] \in \Gamma_{IC} \end{cases} \tag{2}$$

Therefore, the loss functions related to them can be written as:

$$\mathcal{L}_{BC} = \|\mathcal{NN}(x,y,t) - \mathcal{F}_{Dirichlet}\|_2 + \|\frac{\partial \mathcal{NN}(x,y,t)}{\partial n} - \mathcal{F}_{Neumann}\|_2$$
$$\mathcal{L}_{IC} = \|\mathcal{NN}(x,y,0) - \mathcal{G}_{IC}\|_2 \tag{3}$$

where $\|\cdot\|_2$ denotes L_2-norm. In Eq. (1) and (3), partial derivative terms such as $\frac{\partial u}{\partial t}$, $\frac{\partial p}{\partial x}$, $\frac{\partial \mathcal{NN}(x,y,t)}{\partial n}$ can be calculated by using automatic differentiation (AD) within NNs [12,38]. AD enables one to compute the partial derivative of the output with respect to the input of the NNs leveraging back-propagation operation. Finally, PINN loss for NS equations is calculated as below:

$$\mathcal{L}_{PINN} = \mathcal{L}_{mass} + \mathcal{L}_{x-momentum} + \mathcal{L}_{y-momentum} + \mathcal{L}_{BC} + \mathcal{L}_{IC} \tag{4}$$

Note that each loss term in Eq. (4) is evaluated using collocation points. Specifically, $\mathcal{L}_{mass}$, $\mathcal{L}_{x-momentum}$, $\mathcal{L}_{y-momentum}$ are evaluated within the collocation points sampled in Ω, and $\mathcal{L}_{BC}$ within points in $\Gamma_{DirichletBC}$ or $\Gamma_{NeumannBC}$, and $\mathcal{L}_{IC}$ within Γ_{IC}. In this context, data-free PINN means that the training of the PINN does not require the labeled dataset with input (x,y,t) and output (u,v,p), while it still requires the locations of the collocation points (x,y,t) to be explored. Based on this fact, one can easily infer that the sampling strategy of these collocation points would greatly affect the performance of PINN, just as CFD results are highly dependent on the mesh quality.

2.1.2. Data-driven PINNs (DD-PINNs)

Though data-free PINN has the decisive advantage that it does not require any labeled dataset for its training, as will be figured out later in this study, the data-free approach can cause training failures in that PDEs themselves are solved directly without any data for guidance. With this regard, another approach to leverage the labeled dataset is also being adopted [30,32], and it is referred to as DD-PINNs in this study. It has been frequently utilized in inverse problems, which aim to predict unknown PDE coefficients or BC/IC from sparse measurements of the PDE solutions [39]. Its approach does not modify the architecture of the data-free PINN; it only adds one more loss term as below:

$$\mathcal{L}_{DD-PINN} = \mathcal{L}_{mass} + \mathcal{L}_{x-momentum} + \mathcal{L}_{y-momentum} + \mathcal{L}_{BC} + \mathcal{L}_{IC} + w_{data} * \mathcal{L}_{data} \tag{5}$$

where $\mathcal{L}_{data}$ denotes the loss term of the labeled guide data (as in Fig. 2, $\mathcal{L}_{data}$ is considered if PINN adopts data-driven concept). For example, when the velocity fields of some locations are given, $\mathcal{L}_{data} = \|\hat{\mathbf{u}} - \mathbf{u}_{data}\|_2$, where $\hat{\mathbf{u}}$ and $\mathbf{u}_{data}$ respectively indicate the predicted velocity by PINNs and the velocity of the labeled data. By including only this $\mathcal{L}_{data}$ additional term in the loss function, Gopakumar et al. [32] argued that DD-PINNs can be biased towards the solution in forward problems, reducing both the approximation error and optimization error. They added that the former error can be reduced due to the regularization effects of the leveraged data points, and the latter due to the morphing effect on the loss landscape towards a simpler optimization task. In this study, to control the effectiveness of this $\mathcal{L}_{data}$ term, the hyperparameter w_{data} is additionally considered: the larger its value, the greater the impact of the data-driven loss term.

The guide data used in the data-driven approach for PINNs can be regarded as the measurements from the sparse sensor data in the experimental domain [40–43]. However, for the simulation domain, using data from numerical simulation in forward problems can be paradoxical [32]; the purpose of PINNs in forward problems is to replace numerical simulation, but for DD-PINNs, we need to perform numerical simulation to obtain data that will be used to train PINNs. In fact, Gopakumar et al. [32] have already raised this point and suggested using part (1%, 5%, and 10%) of the data obtained from the low-fidelity simulation, which was called *coarse regulated PINN* in their work. Their main focus was on the loss landscapes of PINNs: landscapes of the DD-PINNs regulated with sparse simulation data have more pronounced valleys, which helps the optimizer to converge to an optimal point more easily (thus reducing the optimization error). However, they failed to achieve reasonable accuracy when solving 2D NS equations for flow around a rectangular block: DD-PINNs were only able to detect the presence of the block, not the detailed structure of the flow field. In this context, this study aims to verify their practical utility by comprehensively identifying the conditions under which their accuracy improves through a variety of parametric studies, which is one of our main contributions.

2.2. Adaptive sampling techniques

2.2.1. Existing adaptive sampling techniques

As mentioned in Section 2.1.1, the results of PINN strongly depend on the sampling strategy of the collocation points, since the collocation distribution acts like the grid in the CFD simulation. However, the conventional PINN approach often uses the uniform sampling strategies [18,30,44]. Such strategies are easy to implement, but have a critical aspect in that they cannot consider the viscous effects of the flow field in an efficient way as CFD, where the mesh is generated to be dense near the inner objects or wall [45]. In addition, they sample the collocation points only once so that their distribution does not change during the entire training procedure, which leads to the inefficient training of PINN. To overcome this inefficiency, Lu et al. [20] proposed the adaptive sampling strategy where the collocation points are re-sampled at the region of high PDE residual. This approach is intuitive in that since PINN aims to reduce the residuals of the PDEs, additional sample points are added where the residuals are high. Specifically, it first randomly samples M evaluation points to estimate the residuals in the entire region, and then sorts the N largest residual points among them, which will be the newly added collocation points. This deterministic nature makes the points to be added highly dependent on the size of M. More specifically, if M is too large, the adaptive points will be concentrated in a small region, while a small M would lead to even sampling, which is not consistent with its residual-aware concept. Therefore, a stochastic approach

using a probability density function (PDF) has been proposed [46]. It samples the adaptive points based on a PDF proportional to the residuals, thus removing the deterministic nature of the previous approach. There has been another attempt to add new samples where the gradient of the outputs (e.g. u, v, p for flow field prediction) is large [47]. This gradient-aware approach is well suited to the field of fluid dynamics, where the steep gradient of the flow quantities often plays a critical role in the flow phenomena. However, its algorithm was implemented manually with prior knowledge of the solution in the work by Mao et al. [47], intentionally adding points around the discontinuous region.

2.2.2. Vorticity-aware adaptive sampling

Existing adaptive sampling approaches in the previous section can be used universally across all engineering disciplines because they have focused on the general properties of the PINN architecture, namely residuals of PDEs or gradients of outputs. However, when it comes to the specific domain, there may be an alternative way to realize adaptive sampling that works more efficiently than the approaches designed for general domains. To explore this potential possibility, we attempt to develop an adaptive sampling technique specialized for fluid problems. In fact, numerous conventional CFD studies have already attempted to accurately capture the vortices by adding meshes in regions of high vorticity (see Appendix A for more details). To extend these approaches in the CFD community to the PINN architecture filling their academic gap, this study proposes a novel algorithm that adaptively adds new collocation points where high vorticity exists. This method that focuses on the region of high vorticity magnitude is referred to as the "vorticity-aware" approach throughout this manuscript, and its algorithm can be summarized as Algorithm 1. First, sample the locations where the vorticity magnitude is to be evaluated, X_{eval} (line 5). Note that the calculation of the vorticity (ω) requires a partial derivative of velocities with respect to the spatial coordinates ($\frac{\partial u}{\partial y}$, $\frac{\partial v}{\partial x}$), which can be easily performed owing to the AD property of the NNs. Then we define the PDF, $p_V(\mathbf{x})$, which is proportional to the vorticity magnitudes computed at X_{eval} as follows (line 6) [48]:

$$p_V(\mathbf{x}) = \frac{|\omega|^k(\mathbf{x})}{\mathbb{E}[|\omega|^k(\mathbf{x})]} \tag{6}$$

where $\mathbf{x} = (x, y) \in \Omega$ for the 2D spatial domain, and k is a hyperparameter for adjusting the intensity of the correlation between new sample points and $p_V(\mathbf{x})$. More specifically, a large value of k indicates that points will be sampled more intensely according to the $p_V(\mathbf{x})$ distribution (i.e., more on the high vorticity region), while a small value leads to uniform sampling. Therefore, it is referred to as "stochastic intensity" throughout this study. Back to Algorithm 1, as in line 7, X_{adapt} of size N are newly sampled according to $p_V(\mathbf{x})$. Since the original size of the whole collocation points is M, the new points X_{random} of size $(M - N)$ are additionally randomly sampled to keep the original dataset size (line 8). Finally, the next collocation dataset is defined as $X_{adapt} \cup X_{random}$, and retraining of the PINN continues (line 9, 10) — this retraining can be considered as transfer learning, which makes the repetitive training of NNs efficient by using the pre-trained model parameters [49], and it can also be considered as a way to quickly incorporate the data obtained from the physical space into the virtual space model reducing the time delay in DT environment. This process is repeated until the predefined maximum iteration is reached (line 4, 11). Note that the size ratio of the newly sampled points to the total collocation points, $\frac{N}{M}$, is the hyperparameter for the adaptive sampling approaches. The effects of the mentioned hyperparameters, k and $\frac{N}{M}$, will be discussed later in Section 3.2.

Algorithm 1 Vorticity-aware adaptive sampling

1: Prepare the initial collocation points X_{train}, size of M
2: Set N, which is the size of the points to be updated
3: Train PINN with k_{init} epochs
4: **repeat**
5: Randomly sample evaluation points size of M $(=X_{eval})$
6: Calculate PDF $p_V(\mathbf{x})$ that is proportional to the vorticity magnitudes calculated on X_{eval}
7: Sample N points from X_{eval} according to $p_V(\mathbf{x})$ distribution $(=X_{adapt})$
8: Randomly sample points size of $M - N$ $(=X_{random})$
9: Update collocation points ▷ $X_{train} = X_{adapt} \cup X_{random}$
10: Train PINN with k_{adapt} epochs
11: **until** *Max epoch is reached;*

2.2.3. Fusion of all sampling techniques

Though this study proposes the new technique named vorticity-aware adaptive sampling, existing adaptive sampling approaches based on PDE residuals and output gradients cannot simply be considered inefficient for flow problems. Rather, we fuse the proposed vorticity-aware concept (Section 2.2.2) with existing approaches (Section 2.2.1), adopting all their ideas. In particular, the residual-aware concept based on mass conservation PDE, the gradient-aware concept in terms of freestream direction ($\frac{\partial u}{\partial x}$), and the vorticity-aware concept are all fused: the corresponding algorithm is given in Algorithm 2. Note that the only difference with the vorticity-aware algorithm in Algorithm 1 is that it adds $N/3$ points each for all three criteria, whereas vorticity-aware adds the whole N points according to the vorticity magnitude only. Also, $P_R(x)$ and $P_G(x)$, the PDFs used for the residual and gradient criteria in lines 6, 7, can be computed by replacing the vorticity magnitude $|\omega|$ in Eq. (6) by the PDE residual and output gradient, respectively. From now on, "RGV-xyz" refers to the adaptive sampling technique where N new points to be added consist of residual, gradient, and vorticity-aware points with the ratio x:y:z. For example, vorticity-aware approach in Algorithm 1 can be referred to as

"RGV-001" (vorticity-aware only), while Algorithm 2 as "RGV-111" (all three criteria are considered equally). Similarly, "RGV-110" denotes a sampling that adopts residual and gradient-aware sampling without consideration of vorticity, by adding $N/2$ points for each of these two criteria.

Algorithm 2 Residual-Gradient-Vorticity-aware adaptive sampling

Prepare the initial collocation points X_{train}, size of M
Set N, which is the size of the points to be updated
Train PINN with k_{init} epochs
repeat
  Randomly sample evaluation points size of M $(=X_{eval})$
  Calculate PDF $p_R(\mathbf{x})$ based on the residual of PDEs on X_{eval}
  Calculate PDF $p_G(\mathbf{x})$ based on the gradient of outputs on X_{eval}
  Calculate PDF $p_V(\mathbf{x})$ based on the vorticity magnitude on X_{eval}
  Sample $N/3$ points from X_{eval} according to $p_R(\mathbf{x})$ distribution $(=X_{adapt\_R})$
  Sample $N/3$ points from X_{eval} according to $p_G(\mathbf{x})$ distribution $(=X_{adapt\_G})$
  Sample $N/3$ points from X_{eval} according to $p_V(\mathbf{x})$ distribution $(=X_{adapt\_V})$
  Randomly sample points size of $M-N$ $(=X_{random})$
  Update collocation points ▷ $X_{train} = X_{adapt\_R} \cup X_{adapt\_G} \cup X_{adapt\_V} \cup X_{random}$
  Train PINN with k_{adapt} epochs
until *Max epoch is reached;*

3. Preliminary study: lid-driven cavity flow at $Re = 100$

In this section, the lid-driven cavity flow problem at Reynolds number 100 is investigated using data-free PINNs. DNS is also performed using OpenFOAM, the open-source CFD framework with the finite volume method, to compare the results of PINNs with conventional CFD approaches. IcoFoam solver in OpenFOAM is utilized, which is a transient solver for incompressible laminar Newtonian fluid without a turbulence model [50]. Uniform grids with different resolution levels (20×20, 40×40, 80×80, and 160×160) are adopted and their details can be found in Fig. 16, Appendix C. The simulations are run on a single CPU core of the Intel i7-1165G7 processor until the residual dropped below 10^{-6}, and simulation wall times for each grid are 2, 8, 80, and 875 s respectively. The validation of the DNS results is also performed as shown in Fig. 17, Appendix C. To estimate the error of the PINNs compared to DNS, the root mean squared error (RMSE) between PINN prediction ($\hat{u}$) and DNS results from the 160×160 grid (u_{data}) is leveraged as below:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(\hat{u} - u_{data})^2}{N}} \tag{7}$$

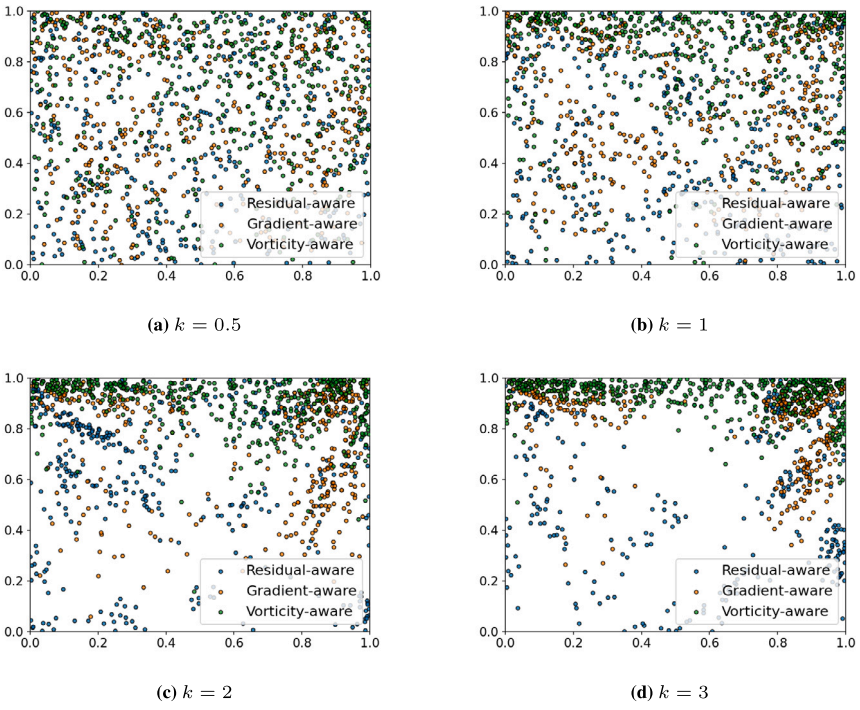
where N is the test dataset size.

3.1. Hyperparameter tuning

Before investigating the overall performance of data-free PINNs at $Re = 100$, hyperparameter tuning is performed to adopt the best model architecture. For this purpose, the number of hidden layers ($N_{layer} \in \{4, 6, 8\}$), the number of nodes at each hidden layer ($N_{node} \in \{32, 64, 128\}$), and the learning rate ($lr \in \{1e-3, 1e-4, 1e-5\}$) are set as hyperparameters. Note that for this tuning, the fixed learning rate is used without the decaying strategy (the decaying technique will be introduced in the higher Reynolds cases, Section 6). For the activation function, a hyperbolic tangent is applied as recommended by Wang et al. [51]: while there are adaptive activation functions proposed by Jagtap et al. [52,53] to improve the training speed, we decided to choose the hyperbolic tangent since the acceleration of PINNs is beyond the scope of this study. Each PINN model is trained for 20,000 iterations with an Adam optimizer, and 2000 collocation points and 500 boundary points are selected. Only vanilla PINNs without special adaptive sampling techniques (hereafter called "Van") are trained to briefly explore the hyperparameter space. Considering the stochastic nature of the training, each PINN is trained at least four times, and their averages are presented throughout all experiments in this study. Finally, errors with respect to the velocity components (u, v) between PINN and DNS are calculated as Table 9 in Appendix B, and they indicate that $N_{layer} = 6, N_{node} = 32, lr = 1e-3$ and $N_{layer} = 8, N_{node} = 32, lr = 1e-3$ cases have the best results. The former one has been selected for further investigation due to its simpler architecture: note that the corresponding model requires 810 s for the training with NVIDIA 3080 GPU, and the training time for each hyperparameter tuning case averages 900 s (therefore about 24,300 s for all cases).

3.2. Parametric studies for adaptive sampling techniques

Based on the PINN architecture selected in Section 3.1, this section aims to scrutinize the practical implications of the various adaptive samplings that promote automatic interaction between the physical and virtual spaces of the DT without human intervention. To this end, effects of the stochastic intensity (k) and the ratio of the newly updated collocation points ($\frac{N}{M}$) are investigated in Sections 3.2.1 and 3.2.2, respectively, effects of the number of collocation points can be found in Section 3.2.3, and the final remarks on their results are given in Section 3.2.4.



(a) $k = 0.5$ **(b)** $k = 1$

(c) $k = 2$ **(d)** $k = 3$

Fig. 3. Newly added adaptive points according to each criterion for different k values. Note that all figures are the results of the last iteration in the RGV-111 model in Section 3.2.1.

3.2.1. *Effects of stochastic intensity (k)*

As mentioned in Section 2.2.2, the hyperparameter k determines the intensity of the correlation between the collocation points to be newly added and residual/gradient/vorticity PDF. When $k \to \infty$, adaptive sampling no longer adopts stochastic but deterministic selection, which means that the top N points with respect to residual/gradient/vorticity values will be selected [48]. On the other hand, when $k = 0$, the PDF of the residual/gradient/vorticity ($p_R(\mathbf{x})/p_G(\mathbf{x})/p_V(\mathbf{x})$) becomes constant over $\mathbf{x}$, indicating that the uniform sampling will be performed.

This section aims to investigate the effects of the above stochastic intensity (k) for various adaptive sampling techniques, and therefore a total of 8 techniques are adopted: Van (random sampling), RGV-100, RGV-010, RGV-001, RGV-110, RGV-101, RGV-011, RGV-111 (see Section 2.2.3 for abbreviations). To focus on analyzing the effects of k, another hyperparameter $\frac{N}{M}$ is set to 0.7, where M is 2000. All PINNs with different adaptive sampling are trained for 20,000 epochs, and the adaptive sampling procedure is repeated every 4000 epochs (i.e., after the initial sampling, the adaptive sampling procedure is performed four times per model). Here, note again that this repetition of training with updated collocation points can be considered as transfer learning approach mentioned in Section 2.2.2.

A parametric study is conducted for $k \in [0.5, 1, 2, 3]$. Fig. 3 shows the N updated collocation points at the last adaptive sampling iteration in RGV-111. There are three labels for the points, and each group shows the newly added points according to the high residual/gradient/vorticity value. As expected, the lower the k, the updated points are evenly distributed as in the random sampling technique, while the higher the k, the more collocation points are added to the specific regions. The quantitative RMSE between DNS from 160×160 grid and PINNs are summarized in Table 1. The average and minimum RMSE values among all trained PINNs within the same k value are also presented: again, since the four models are trained for each case, the minimum RMSE value is derived through all trained models. The interesting point is that although vorticity-aware adaptive sampling is the most indirect algorithm for the training of PINN compared to residual/gradient-aware sampling, it manages to achieve moderate or even better performance. For example, sampling with only vorticity-aware technique (RGV-001) is the best in $k = 0.5$ cases, and the residual-gradient-vorticity-aware technique (RGV-111) is the best in $k = 1$ cases. To further investigate if k affects the accuracy of PINNs, one-way analysis of variance (ANOVA) is performed [54]: within the same sampling approach, 100 y-velocity RMSE values at different locations are extracted for every k value. The results are shown in the right part of Table 1, and it can be confirmed that only the RGV-001 approach has a statistically significant difference (p-value < 0.05) due to varying k — which indicates that the appropriate selection of k value is critical in vorticity-aware sampling. Since the $k = 3$ case has the minimum RMSE and to provide at least some contrast between the different sampling algorithms, it is adopted throughout this paper.

3.2.2. *Effects of the updated collocation sampling ratio ($\frac{N}{M}$)*

With $k = 3$ chosen in Section 3.2.1, this section focuses on the effects of the updated collocation sampling ratio, $\frac{N}{M}$. Parametric studies are conducted for $\frac{N}{M} \in [0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]$. Herein, note that out of 2000 collocation points, $\frac{N}{M} = 0.3$ indicates

Table 1
RMSE of y-velocity for different k values (Section 3.2.1). Four models are trained for each case, therefore mean and minimum values of all trained models are shown in the last two rows. Results of one-way ANOVA are also presented to examine the effect of k in each sampling method.

	k				ANOVA	
	0.5	1	2	3	F statistic	p-value
Van	0.0360	–	–	–	–	–
RGV-100	0.0440	0.0532	0.0646	0.0810	2.3676	0.0704
RGV-010	0.0349	0.0393	0.0308	**0.0273**	1.9975	0.1138
RGV-001	**0.0298**	0.0490	0.0644	0.0388	4.4502	**0.0043**
RGV-110	0.0531	0.0501	**0.0301**	0.0336	2.2290	0.0843
RGV-101	0.0550	0.0424	0.0378	0.0422	1.2376	0.2957
RGV-011	0.0398	0.0441	0.0332	0.0490	1.3829	0.2475
RGV-111	0.0319	**0.0332**	0.0361	0.0422	0.8059	0.4911
Mean	**0.0406**	0.0434	0.0416	0.0437	–	–
Min	0.0186	0.0202	0.0230	**0.0177**	–	–

Table 2
RMSE of y-velocity for different $\frac{N}{M}$ values (Section 3.2.2). Four models are trained for each case, therefore mean and minimum values of all trained models are shown in the last two rows. Results of one-way ANOVA are also presented to examine the effect of $\frac{N}{M}$ in each sampling method.

	N/M							ANOVA	
	0.3	0.4	0.5	0.6	0.7	0.8	0.9	F statistic	p-value
Van	0.0360	–	–	–	–	–	–	–	–
RGV-100	0.0318	0.0563	0.0364	0.0547	0.0810	0.0372	**0.0328**	2.2381	**0.0380**
RGV-010	0.0619	0.0443	**0.0310**	0.0371	**0.0273**	0.0335	0.0391	3.0807	**0.0055**
RGV-001	**0.0287**	0.0354	0.0342	**0.0309**	0.0388	0.0524	0.0378	3.0903	**0.0054**
RGV-110	0.0353	**0.0334**	0.0574	0.0329	0.0336	0.0403	0.0596	3.0241	**0.0063**
RGV-101	0.0488	0.0368	0.0390	0.0421	0.0422	**0.0332**	0.0703	2.8077	**0.0105**
RGV-011	0.0307	0.0489	0.0397	0.0352	0.0490	0.0358	0.0413	1.2672	0.2702
RGV-111	0.0354	0.0650	0.0456	0.0397	0.0422	0.0498	0.0501	3.3599	**0.0029**
Mean	**0.0386**	0.0445	0.0399	**0.0386**	0.0437	0.0398	0.0459	–	–
Min	0.0208	0.0232	0.0243	0.0226	**0.0177**	0.0207	0.0227	–	–

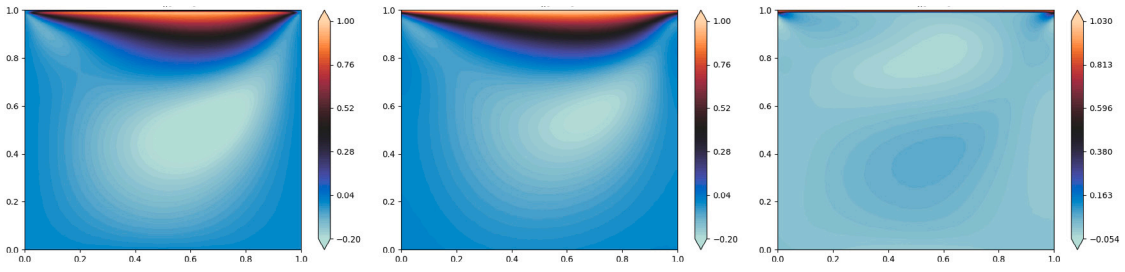
600 points are updated according to the residual/gradient/vorticity criteria, and the other 1400 points are updated randomly. The RMSE results of y-velocity are shown in Table 2. Again, the vorticity-aware approaches (RGV-001 and RGV-101) achieve comparable performance in certain cases: $\frac{N}{M} = 0.3, 0.6, 0.8$. This highlights the potential of user-defined adaptive sampling when the appropriate criterion is chosen, considering that existing approaches such as residual-aware or gradient-aware are much more intuitive and direct ways than focusing on the high vorticity magnitude regions. Throughout all sampling techniques, the mean values of RMSE are best when $\frac{N}{M} = 0.3, 0.6$. A noteworthy point in Tables 1 and 2 is that, depending on the sampling technique and the k and $\frac{N}{M}$ values, adaptive sampling techniques can have worse accuracy than the random sampling technique, Van. These results indicate that the use of adaptive sampling does not always improve PINN performance, but that careful selection of the adaptive sampling type and its hyperparameters is crucial. Again, one-way ANOVA is conducted and except for RGV-011, all sampling approaches are affected by the value of $\frac{N}{M}$ in that they have p-values less than 0.05. Finally, together with the results in Section 3.2.1, hyperparameters of $k = 3$ and $\frac{N}{M} = 0.6$ are adopted for the rest of this study. Before moving on to the next section for higher Reynolds numbers, the flow fields with respect to both velocity components for the RGV-111 model are visualized in Fig. 4: it can be confirmed that the data-free PINNs can predict the overall trends of the flow fields, although some details are captured at low resolutions.

3.2.3. *Effects of the number of collocation points (M)*

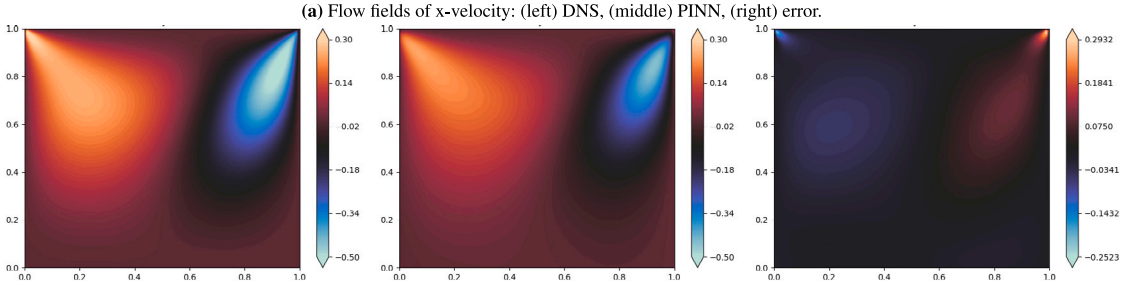
This section investigates the effects of the number of collocation points (M) using the previously selected hyperparameters $k = 3$ and $N/M = 0.6$. The number of points varies from 500 to 3000, with an increment of 500. Only the RGV-001 approach is tested, as it was found to be the best approach when $N/M = 0.6$ in Table 2. The results are presented in Table 3. Contrary to expectations, as the number of points increases, there is no clear trend towards the improvement of results. This inconsistency can be attributed to the stochasticity in the initial sampling and adaptive sampling techniques, which can significantly affect the final accuracy of the PINNs. Also, it is noteworthy that, with an adequate combination of the N/M value, a smaller M value can yield superior results compared to larger values. For example, $M = 2000$ performs better than $M = 2500$ and $M = 3000$ when $N/M = 0.6$. From another perspective, it can also be inferred that $M = 2500$ and $M = 3000$ can be better than $M = 2000$ if the appropriate N/M values are chosen for their cases.

3.2.4. *Remarks on adaptive sampling*

Sections 3.2.1, 3.2.2, and 3.2.3 compare the performances of various sampling techniques in data-free PINNs at low Reynolds number. However, no clear trend is observed throughout parametric studies with respect to the stochastic intensity (k), updated



(a) Flow fields of x-velocity: (left) DNS, (middle) PINN, (right) error.



(b) Flow fields of y-velocity: (left) DNS, (middle) PINN, (right) error.

Fig. 4. Comparison of velocity components at $Re = 100$ between DNS with 160×160 grid and PINN with $k = 3$ and $\frac{N}{M} = 0.6$: (a) x-velocity, (b) y-velocity. The flow fields of PINNs are obtained from the best model among the four trained RGV-111 models.

Table 3
RMSE of y-velocity at $Re = 100$ for different values of M (the number of total collocation points): only RGV-001 is tested.

M	500	1000	1500	2000	2500	3000
RMSE	0.0590	0.0336	0.0487	**0.0309**	0.0430	0.0415

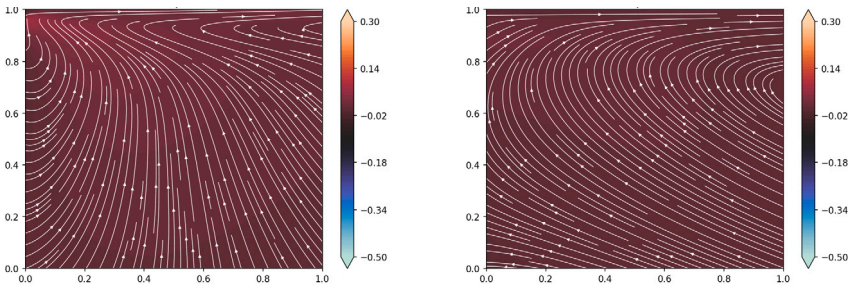
sample ratio ($\frac{N}{M}$), and the number of collocation points (M). Nevertheless, the more comprehensive investigation of the adaptive sampling approaches is judged to be unnecessary for this study for the following reasons: (1) The first purpose of this section was not to find out which sampling is best, but to show how efficiently PINNs' meshless properties allow them to work automatically with different sampling techniques, (2) the second purpose was to shed a light on the potential of user-defined sampling approach tailored for the specific domain, not to exclusively argue that the proposed vorticity-aware approach is the best.

4. Failures of data-free PINNs at higher Reynolds

This paper notes the fact that most of the previous studies on lid-driven cavity flow focused on low Reynolds regions. For example, Amalinadhi et al. [22], Jagtap et al. [23], Li and Feng [24], Bai et al. [25], Wang et al. [26] showed that PINNs successfully worked in the forward problem at $Re = 100$. Also, Wong et al. [27] and Chiu et al. [28] tested PINNs at $Re = 400$ in forward problems. To bridge this gap between toy problems for academic validation and high Reynolds problems encountered in practice, this section investigates the higher Reynolds ($Re = 1000, 3200$) using data-free PINN. Again, to compare the results of PINNs with conventional CFD approaches, DNS is performed by OpenFOAM. The settings for DNS including CFD solver, grids, and CPU core are the same as in Section 3: grids of 20×20, 40×40, 80×80, and 160×160 are utilized and training time for each grid in $Re = 1000$ is 17, 36, 100, and 1853 s, respectively. For $Re = 3200$ case, 20, 98, 473, and 4231 s are taken per each grid. Their validation results are shown in Fig. 17, Appendix C.

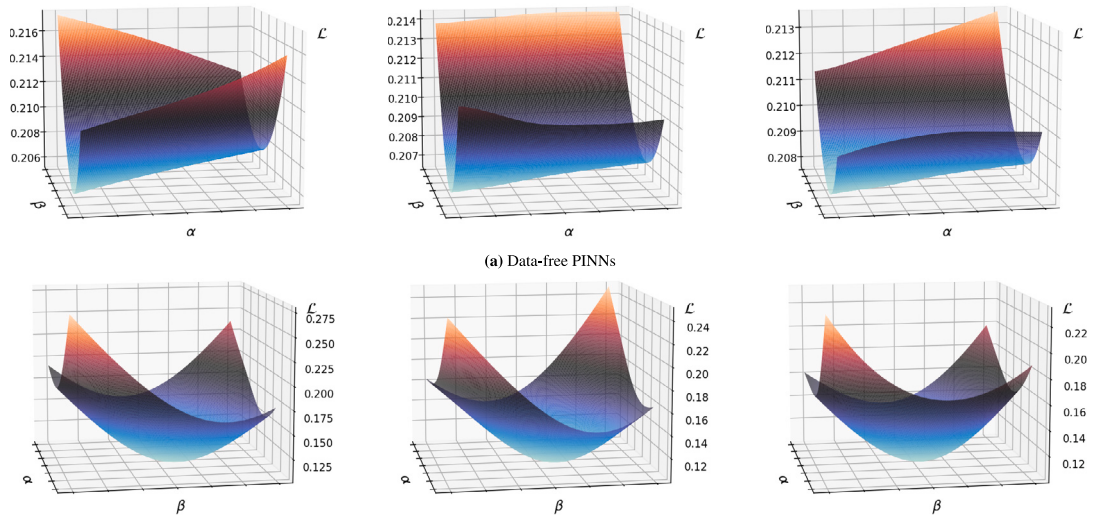
4.1. Experiments at $Re = 1000$ and $Re = 3200$

Data-free PINNs with $N_{layer} = 7, N_{node} = 32, k = 3, \frac{N}{M} = 0.6$ where $M = 5000$ are trained for $Re = 1000$ and $Re = 3200$ cases. As in Section 3.1, only "Van" sampling is applied. Finally, both cases require a training wall time of 900 s: considering that DNS requires 1853 and 4231 s for the 160×160 grid, the computational efficiency of PINNs can be verified. However, they completely failed to predict y-velocity fields along with unrealistic streamlines as shown in Fig. 5. Along with unrealistic streamlines, they completely failed to predict y-velocity fields. More than this case, additional efforts with different hyperparameters ($N_{layer} \in \{5, 7, 9\} \& N_{node} \in \{32, 64, 128\}$) are also tried to find out if the failure is due to insufficient model capacity, but all attempts fail to predict reasonable flow fields.



(a) $Re = 1000$ **(b)** $Re = 3200$

Fig. 5. Prediction of y-velocity flow fields using data-free PINNs: (a) $Re = 1000$, (b) $Re = 3200$. Streamlines are also shown to highlight their failure.



(a) Data-free PINNs

(b) DD-PINNs

Fig. 6. Loss landscape of PINNs at $Re = 3200$: (a) data-free PINNs: from left to right, minimum losses of the trained models are 0.205, 0.207, and 0.208, (b) DD-PINNs: from left to right, minimum losses of the trained models are 0.104, 0.102, and 0.102. Three different models of each type of PINN (with identical settings but trained differently due to the stochastic nature) are visualized.

4.2. Visualization of loss landscapes

Since all data-free PINN models failed at higher Reynolds numbers, we then try to investigate the underlying reason by visualizing the loss landscapes of NNs. The loss landscape in deep learning refers to the representation of loss values ($\mathcal{L}$) with respect to the model parameters of NNs (θ). When there are only two model parameters, one can easily visualize the loss landscape of the model with a 3D contour plot: x, y axes with two model parameters, and z axis with loss values. However, in practice, there are many more model parameters in NNs, and therefore some technique for the projection of high-dimensional model parameters to 2D space is required. To this end, parameters of the already trained model, θ^*, are projected onto the 2D surface consisting of δ and η axes as below [55]:

$$f(\alpha, \beta) = \mathcal{L}(\theta^* + \alpha\delta + \beta\eta) \tag{8}$$

To utilize Eq. (8) to draw high-resolution plots without scale invariance problems, Li et al. [56] proposed the use of filter-wise normalized directions (refer to Ref. [56] for more details).

In fact, Gopakumar et al. [32] already deployed the loss landscapes to argue that data-driven PINNs have sharper minima than data-free PINNs. In a similar context, we visualized the loss landscapes to verify the failure of the data-free PINNs at $Re = 3200$, as shown in Fig. 6(a). As mentioned earlier, all of the experiments in this study train models several times considering the stochastic behavior, and therefore three different data-free PINN models are visualized. Overall, their landscapes seem to have no discernible minima and show flatness (especially along the α axis) compared to the DD-PINNs in Fig. 6(b), which will be discussed in Section 5. Given that non-convexity in the reduced dimensionality indicates that there must be non-convexity in the full dimension [56], one can infer that this is the cause of the failure in training data-free PINNs.

5. Remedy: data-driven PINNs (DD-PINNs)

This section attempts to address the failures of data-free PINNs by leveraging simulation data. To this end, the DD-PINN concept is adopted as a remedy and we focus on $Re = 3200$ case. First, hyperparameter tuning is conducted (Section 5.1). Then, the comprehensive investigations on DD-PINNs in terms of sampling approaches (Section 5.2), the fidelity of guide data (Section 5.3), and number of guide data (Section 5.4) are performed.

5.1. Settings and hyperparameter tuning for DD-PINNs

First, the settings required for DD-PINNs should be discussed. Since they leverage the dataset for guidance, we explored the different datasets obtained from DNS. To be more specific, datasets of different grid densities are obtained: grids of 20 × 20 (guide data size of 400), 40 × 40, 80 × 80, and 160 × 160 at $Re = 3200$. One can easily expect that as the grid becomes denser, the simulation time for preparing the dataset using CFD would increase while the accuracy of DD-PINN would improve. Note that this study made a simplifying assumption by considering the DNS data as a proxy for the experimental sensor data that would be used as data-driven points in the DD-PINN framework. This assumption allows us to focus on the development and evaluation of the DD-PINN methodology. However, it should be noted that in the realization of DT, a near real-time data acquisition/transmission process should be coupled with DD-PINN, including how the experimental data are acquired in real time and how the acquired data are incorporated into the DD-PINN framework, which is considered outside the scope of our work.

Then, as in the $Re = 100$ case study (Section 3.1), hyperparameter tuning is performed for the $Re = 3200$ case. The number of hidden layers ($N_{layer} \in \{5, 7, 9\}$), the number of nodes at each hidden layer ($N_{node} \in \{32, 64, 128\}$), and the weight of the data-driven loss term in Eq. (5) ($w_{data} \in \{1, 5, 10\}$) are set as the hyperparameters to be explored. PINNs are trained for 20,000 iterations with 5000 collocation points and 5000 boundary points. While there are techniques proposed for adaptively adjusting the weights of the loss terms, w_{data} for this study, we decided not to apply them in order to focus on the pure effects of the data-driven loss term without changing its weight during the training procedure [26,57,58]. Among the four grids mentioned above, a 40 × 40 grid is used as a guide dataset for the data-driven approach and the resulting RMSE values are shown in the Table 10 in Appendix B. It indicates that all cases succeed in predicting reasonable flowfields unlike data-free PINNs: among them, $N_{layer} = 7, N_{node} = 32, w_{data} = 10$ with a training time of 960 s is one of the best options considering both velocity components and is therefore utilized for further investigation. Note that since both data-free PINN in Section 4 and DD-PINN here are designed to have the same number of layers and nodes, it can be regarded that the failure of data-free PINN at high Reynolds number is not due to its insufficient model capacity.

To analyze how this DD-PINN model could be well trained compared to data-free PINNs, loss landscapes of DD-PINNs are also shown as Fig. 6(b). One can see that the DD-PINNs have much more convex valleys compared to the data-free PINNs in Fig. 6(a), which can be considered as the reason for their successful training with about half the loss value of the data-free PINNs.

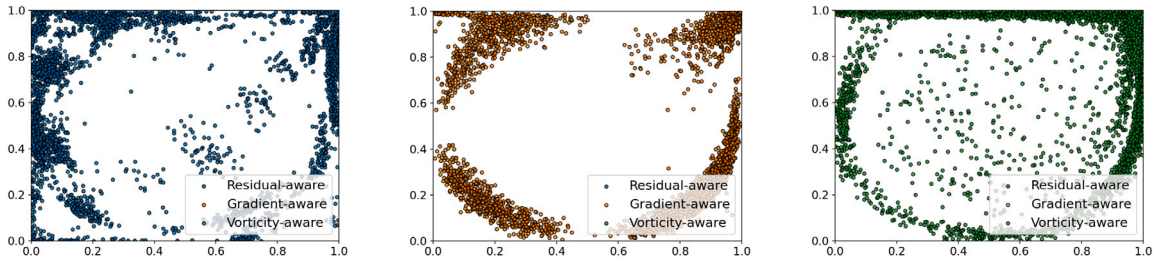
5.2. Effects of sampling techniques

Various adaptive sampling approaches adopted in Section 3.2 are also compared for DD-PINNs with different grids. Table 4 summarizes their results, and the most interesting point is that for all types of grids, the random sampling approach (Van) shows the best performance, although various adaptive sampling techniques still work properly as shown in Fig. 7. Its reason can be inferred that the loss term for the guide data in DD-PINNs ($\mathcal{L}_{data}$) acts as the local regulator, constraining DD-PINNs to predict values similar to the labeled guide data at the specific locations. In this context, PDE loss functions ($\mathcal{L}_{mass} + \mathcal{L}_{x-momentum} + \mathcal{L}_{y-momentum}$) should act as a global calibrator, enabling DD-PINNs to learn the flow fields that globally satisfy the physical conservation laws. However, sampling algorithms working as shown in Fig. 7 are not consistent with this: residual/gradient/vorticity-aware techniques make DD-PINNs rather focus on the specific regions, which is the opposite of the expected role of PDE losses. In contrast, they could have performed better than random sampling in data-free PINNs, since there was no local regulator, $\mathcal{L}_{data}$. Therefore, the PDE losses in data-free PINNs should act as both local and global calibrators: the former is achieved by residual/gradient/vorticity-aware criteria at N points, while the latter is realized by random sampling at $M - N$ points. This can also be deduced from the fact that the error gap between Van and the other adaptive samplings is most pronounced in the results of the 160 × 160 grid, where the role of the global calibrator is most significant due to the strong local regulator with the largest guide dataset.

In this regard, we additionally analyze the effects of N/M again, to verify the trends in adaptive sampling approaches. $N/M \in [0.2, 0.4, 0.6, 0.8]$ are explored and their results are shown in Table 11 in Appendix D: RMSE results between DD-PINNs and DNS with 160 × 160 grid are shown. Since the "Van" method is always the best regardless of the N/M, it can be concluded that for DD-PINNs, local refinement during adaptive sampling techniques (except Van) does not increase the accuracy of the PINNs. They rather decrease the accuracy due to their overlapping role of local refinement with a data-driven loss term. Therefore, only random sampling is used for further investigation on DD-PINNs.

Table 4
RMSE of y-velocity for different guide datasets used in DD-PINNs at $Re = 3200$.

	Grids			
	20 × 20	40 × 40	80 × 80	160 × 160
Van	**0.1032**	**0.0509**	**0.0223**	**0.0120**
RGV-100	0.1056	0.0534	0.0237	0.0152
RGV-010	0.1062	0.0529	0.0237	0.0156
RGV-001	0.1075	0.0536	0.0272	0.0176
RGV-110	0.1061	0.0518	0.0272	0.0226
RGV-101	0.1067	0.0533	0.0280	0.0208
RGV-011	0.1074	0.0560	0.0282	0.0207
RGV-111	0.1079	0.0554	0.0261	0.0195



(a) Residual-aware (RGV-100) **(b)** Gradient-aware (RGV-010) **(c)** Vorticity-aware (RGV-001)

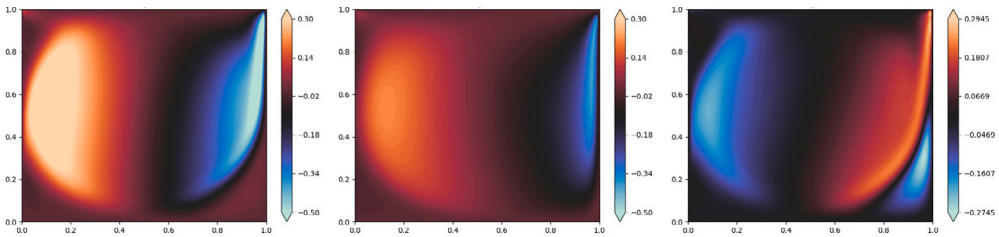
Fig. 7. Adaptive points added by (a) residual-aware, (b) gradient-aware, and (c) vorticity-aware approaches in DD-PINNs with guide data from 40 × 40 grid at $Re = 3200$.

5.3. Effects of guide data fidelity

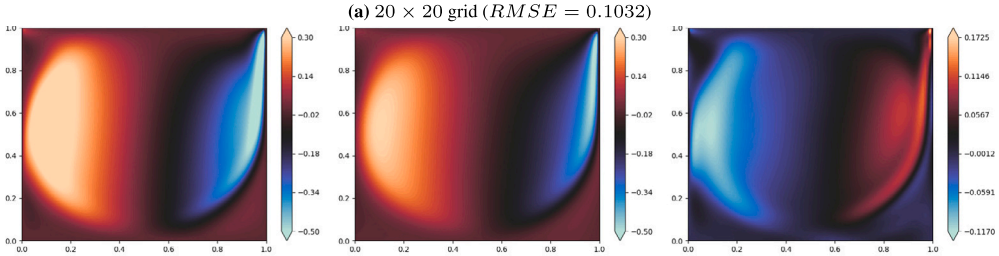
In Table 4, as would be expected, there is a clear trend towards improved accuracies as the guide data is obtained from higher fidelity simulations (i.e., as the mesh becomes denser). However, since the computation time of a denser grid is higher than coarser ones, the fidelity of the data used for the guidance should be selected by the user considering their trade-off between time for obtaining the dataset and the accuracy of DD-PINNs. To see how the accuracies differ between DD-PINNs with various grids, Fig. 8 visualizes the predicted y-velocity flow fields. As shown in Fig. 8(a), DD-PINN fails to reconstruct a realistic flow field with a 20 × 20 grid, while flow fields with reasonable quality are predicted from a 40 × 40 grid. Although the results from 40 × 40 grid are not as accurate as those from 80 × 80, considering that Ghia et al. [59] used 129 × 129 grid for the numerical simulation at $Re = 3200$ for the same domain, the performance of DD-PINN to capture the overall flow trends using only 40 × 40 grid with the help of PINN framework can be considered as remarkable. Fig. 9 shows more quantitative results: DD-PINNs with 80 × 80 and 160 × 160 give almost the same velocity profiles with Ghia et al. [59] and DNS results along the $x = 0.5$ and $y = 0.5$ lines, while the 40 × 40 grid-based DD-PINN only follows the global trends of the flow fields with unsatisfactory local fitting.

5.4. Effects of guide data size

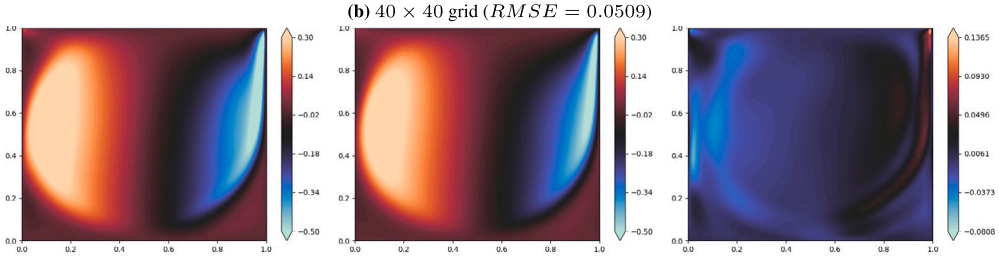
In previous sections, we select DD-PINN with a 40 × 40 grid as the baseline because it is judged to have the potential to train the global trend of the flow fields with a moderate trade-off between accuracy and time required. Although 1600 points can be exploited to guide PINNs in the 40 × 40 grid, we further investigate the effect of the number of guide data, considering that the physical space in DT often only allows the acquisition of sparse sensor data. To this end, several candidates of the guide datasets are defined as follows: first, we divide the range of x and y coordinates of the problem domain into $(n_{divide} + 1)$ equal intervals. This division creates a grid of points so that there are $n_{divide} \times n_{divide}$ points in total, evenly distributed within the domain. The reason for using uniform sampling is that we assume there is no prior knowledge of how to effectively locate the sensors (see Appendix E when the prior knowledge is given). In this study, $n_{divide} \in [2, 5, 10, 20]$ are explored and their RMSE values for y-velocity are summarized in Table 5. It can be confirmed that in the case of $20 * 20$, which uses only a quarter of the guide data of the baseline grid 40 × 40, it can perform as well as the baseline $40 * 40$. Going further, loss landscapes with different n_{divide} values are compared in Fig. 10. The noteworthy point here is that as the number of guide data increases, its regularization effect increases, and thus the convexity of the landscapes increases — as observed in Ref. [32]. This phenomenon supports the points in Section 5.2 that guide data in DD-PINNs act as local regulators, morphing the landscape of the loss function to allow optimizers to traverse it easily [32]. It is also important to note that while the convexity of the $10 * 10$ (Fig. 10(c)) and $20 * 20$ (Fig. 10(d)) cases are similar, their minimum losses can be quite different due to the different absolute locations of the valley, despite the similar convexity in neighboring landscapes.



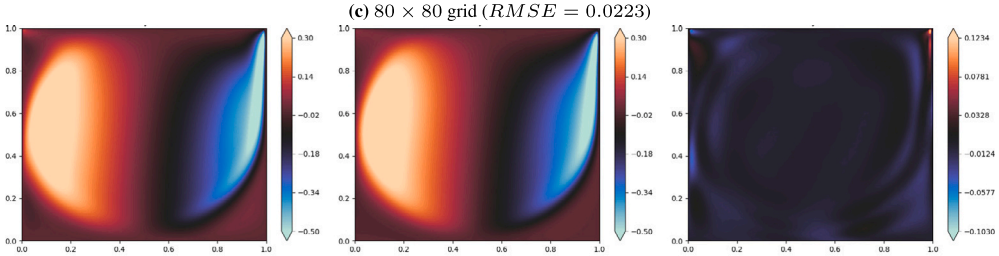
(a) 20×20 grid ($RMSE = 0.1032$)



(b) 40×40 grid ($RMSE = 0.0509$)



(c) 80×80 grid ($RMSE = 0.0223$)



(d) 160×160 grid ($RMSE = 0.0120$)

Fig. 8. Flow fields with respect to y-velocity predicted by DD-PINNs with various fidelities of guide data at $Re = 3200$: (left column) DNS with 160×160 grid, (middle column) PINN, (right column) error.

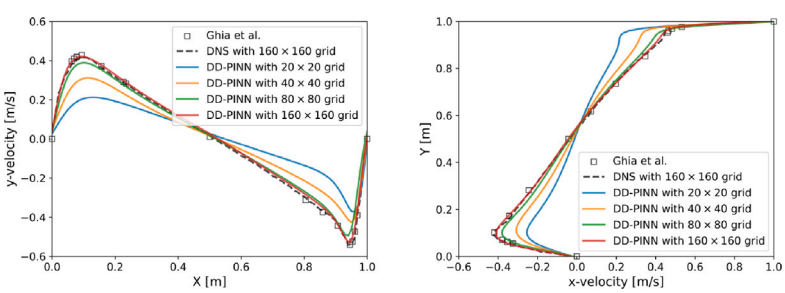
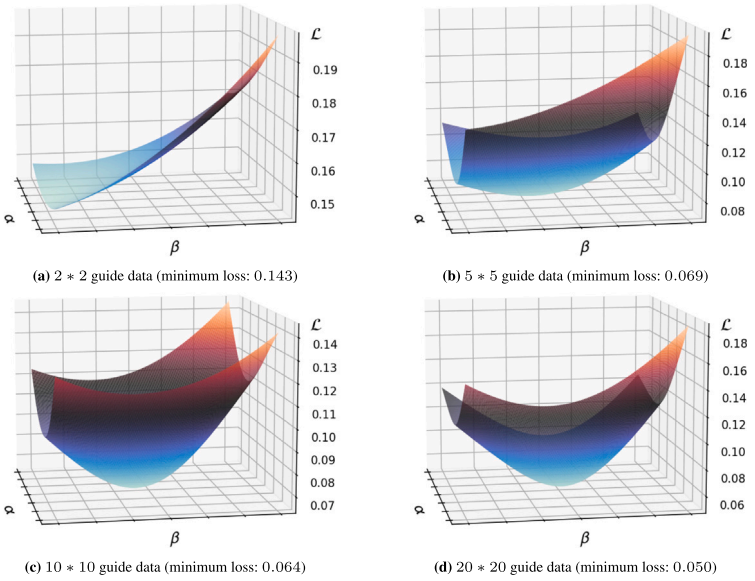


Fig. 9. Comparison of DD-PINNs with results from OpenFOAM DNS and Ghia et al. [59] at $Re = 3200$: (left) y-velocity profile along $y = 0.5$, (right) x-velocity along $x = 0.5$.

Table 5
RMSE of y-velocity predicted by DD-PINNs at $Re = 3200$: different sizes of guide data are extracted from 40×40 grid.

	Guide data for DD-PINN with 40×40 grid				
	$2 * 2$	$5 * 5$	$10 * 10$	$20 * 20$	$40 * 40$
RMSE	0.1574	0.0698	0.0658	**0.0504**	**0.0509**



(a) $2 * 2$ guide data (minimum loss: 0.143)
(b) $5 * 5$ guide data (minimum loss: 0.069)
(c) $10 * 10$ guide data (minimum loss: 0.064)
(d) $20 * 20$ guide data (minimum loss: 0.050)

Fig. 10. Loss landscape of 40×40 grid-based DD-PINNs with different size of guide data at $Re = 3200$: (a) $2 * 2$ guide data, (b) $5 * 5$ guide data, (c) $10 * 10$ guide data, and (d) $20 * 20$ guide data. As the number of guide data increases, there is a clear trend towards convexity in loss landscapes.

6. Extension of DD-PINNs to parametric NS equations and multi-fidelity approach

6.1. DD-PINNs for parametric NS equations

Section 5 successfully showed that the proposed DD-PINNs leveraging CFD guide data can remedy the failure in data-free PINNs. However, there is still a serious limitation: it is not scalable to the parametric NS equations as it is. Since conventional PINN models for NS equations are trained for the specific Re, flow fields with respect to other Reynolds numbers cannot be predicted; new training is required for new Re. To overcome this limitation, Sun et al. [18] proposed a data-free PINN that has modified inputs consisting not only of spatial and temporal coordinates, but also the Re that appears in the NS equations. They also noted that since the PDE parameter, Re in their study, is trained as an input to the NNs, it can be exploited for uncertainty propagation of Re using Monte Carlo simulation. We extended this parametric setting to our DD-PINNs to verify their scalability and robustness in the context of DT, where general physics should be taken into account.

To comprehensively investigate the performance of the DD-PINNs applied to parametric PDEs compared to existing approaches, we train two additional models. First, conventional data-free PINN is trained. Second, as in the study by Sun et al. [18], purely data-driven NNs that utilize only labeled guide data and boundary conditions without PDE information (i.e., without $\mathcal{L}_{mass}$, $\mathcal{L}_{x-momentum}$, $\mathcal{L}_{y-momentum}$ in Eq. (5)) are trained. To train all these models, guide data only at $Re \in [100, 1000, 3200]$ are used. As in Section 5.3, the results of OpenFOAM with a 40×40 grid at these Re are leveraged as guide data. Note that the learning rate is adaptively chosen as [1e−3, 5e−4, 1e−4, 5e−5, 1e−5] for every 8000 epochs (therefore 40,000 epochs in total), with the identical weights on the data-driven loss terms ($w_{data} = 10$) for all $Re \in [100, 1000, 3200]$, and hard boundary conditions are applied for $x = 0$, $x = 1$, $y = 0$ walls [39] (see Appendix E). The model architecture used in Section 5.1 for DD-PINN, with $N_{layer} = 7$ and $N_{node} = 32$, is also applied here. A total of 15,000 collocation points are utilized in the range of $Re \in [50, 4000]$ with random sampling approach: Table 12 in Appendix D shows again that random sampling is still the best among various sampling techniques, especially at high Reynolds number. For training purely data-driven NN, data-free PINN, and DD-PINN models, they require an average of 4020 s, with insignificant differences among them.

Their results are compared with DNS results from 160×160 grids with respect to various Re values, and RMSE values are given in Table 6. Herein, to evaluate both interpolation and extrapolation performance, the trained models are tested at $Re = 500$, 2000 (interpolation), and $Re = 50$, 4000, 5000, 6000 (extrapolation). It can be confirmed that the conventional data-free PINN, which has the largest errors, fails to predict realistic flow fields as Fig. 5. However, conventional data-driven NN and DD-PINN succeed

Table 6
RMSE of y-velocity for different NN architectures. Below the Reynolds numbers, *tr* indicates *Re* used for the training, while *ext* and *int* indicate *Re* values to be tested by extrapolation and interpolation, respectively.

	Reynolds number								
	50 (*ext*)	100 (***tr***)	500 (*int*)	1000 (***tr***)	2000 (*int*)	3200 (***tr***)	4000 (*ext*)	5000 (*ext*)	6000 (*ext*)
Purely data-driven NN	0.0237	**0.0062**	0.0642	**0.0232**	0.0652	0.0490	0.0742	0.1047	0.1327
Data-free PINN	0.0940	0.1060	0.1698	0.1890	0.1984	0.1994	0.1998	0.1979	0.1945
DD-PINN	0.0229	0.0093	0.0367	0.0269	0.0454	0.0530	0.0565	0.0589	0.0623
MF-DD-PINN (case 1)	0.0264	0.0093	0.0363	0.0261	0.0281	0.0280	0.0295	0.0339	0.0429
MF-DD-PINN (case 2)	0.0202	0.0096	**0.0274**	0.0265	0.0271	0.0248	0.0260	0.0332	0.0470
MF-DD-PINN (case 3)	**0.0194**	0.0090	0.0341	0.0258	**0.0241**	0.0219	0.0233	0.0318	0.0497
MF-DD-PINN (case 4)	0.0196	0.0087	0.0306	0.0256	**0.0241**	**0.0217**	**0.0214**	**0.0248**	**0.0364**

Table 7
Details of the guide data used to train multi-fidelity DD-PINN models. Note that the guide data remains the same for $Re = 100$ and 1000, while only the guide data at $Re = 3200$ varies. Additionally, for cases 3 and 4, which use a 160×160 grid, only $5 * 5$ points of it are used, assuming they are obtained from sparse but expensive sensors.

MF-DD-PINN models	Reynolds number		
	100	1000	3200
Case 1	40×40 grid ($w_{data} = 10$)		80×80 grid ($w_{data} = 10$)
Case 2			80×80 grid ($w_{data} = 15$)
Case 3			80×80 grid ($w_{data} = 15$) 160×160 grid ($w_{data} = 15$, only $5 * 5$ points are used)
Case 4			80×80 grid ($w_{data} = 15$) 160×160 grid ($w_{data} = 20$, only $5 * 5$ points are used)

in reconstructing reasonable y-velocity flow fields at various *Re*, not only for the interpolation tasks but also for the extrapolation tasks. In more detail, conventional data-driven NN shows better accuracy at the trained *Re*: 100, 1000, and 3200. On the other hand, DD-PINN shows better performance in interpolation and extrapolation tasks. The reason for this trend is that the conventional data-driven NN focuses only on learning the flow fields at the labeled *Re* without considering the physics contained in the PDEs, and therefore shows better accuracy than DD-PINN only at the *Re* used for training. In contrast, DD-PINN is able to learn physical properties as it considers not only the guide data, but also the physics within PDEs in different *Re* regions, and thus achieves superior generality at *Re* not used during training. For a more intuitive comparison, the flow fields predicted by purely data-driven NN and the proposed DD-PINN are compared in Fig. 11 and Fig. 12, respectively. As discussed, the purely data-driven NN predicts accurate flow fields only at trained *Re*: for example, it interpolates the flow field at $Re = 500$ to be more similar to that at $Re = 100$ than at $Re = 1000$, while DNS indicates that the flow at $Re = 500$ should be more similar to that at $Re = 1000$. Furthermore, especially at the higher Reynolds numbers, it shows much worse quality than DD-PINN: see the predicted flow fields at $Re = 2000$ and $Re = 4000$.

6.2. Further extension: multi-fidelity DD-PINNs

In Section 6.1, OpenFOAM results obtained from 40×40 grids are used as guide data for all *Re* to train DD-PINN. This section focuses on the fact that it is not always the case that the guide datasets are obtained from identical grids (or sensors) in DT scenarios. Similarly, one does not have to use the datasets from identical sources. That is, since prediction is more difficult at higher Reynolds numbers, one can use higher fidelity guide data (denser CFD grid or higher quality experimental sensors) for the higher *Re* values. This is referred to as multi-fidelity DD-PINNs in our study, and this section aims to validate their effectiveness. According to Appendix C, it is shown that 40×40 grids are sufficient for $Re = 100$, 1000, while $Re = 3200$ requires at least 80×80 grid: therefore, multi-fidelity DD-PINNs take corresponding guide data from each grid for each Reynolds number. This framework intends to exploit more accurate guide data at the higher *Re*, expecting superior accuracy, while reducing the computational cost at lower *Re* where the guide data from coarse grid brings sufficient accuracy.

Several multi-fidelity DD-PINNs are trained with different guide data combinations: model architectures of them remain the same as DD-PINN in Section 6.1 and descriptions of their guide data are summarized in Table 7. Case 1 denotes the baseline multi-fidelity DD-PINN (MF-DD-PINN) where for lower Reynolds (100 and 1,000), 40×40 grid is used as guide data and 80×80 grid is used for $Re = 3200$. All of the weight terms for each data-driven loss, w_{data}, are set as 10. In case 2, w_{data} for $Re = 3200$ increases to 15. In case 3, the 160×160 grid is additionally utilized with $w_{data} = 15$, and therefore it can be regarded that three different fidelities are utilized in this case. However, only $5 * 5$ points are leveraged assuming those high-fidelity but expensive data are obtained from the sparse sensors (as in Section 5.4, equal spacing is applied for selecting 25 points). Lastly, case 4 increases w_{data} of $5 * 5$ points to 20 in order to verify the effect of the 160×160 grid's guide data as its impact intensifies.

Training the models from case 1 to case 4 takes about 4000 s for the first two models and about 4800 s for the next two models. Considering that DNS takes 4231 s for the 160×160 grid just for the single case $Re = 3200$, the computational efficiency of

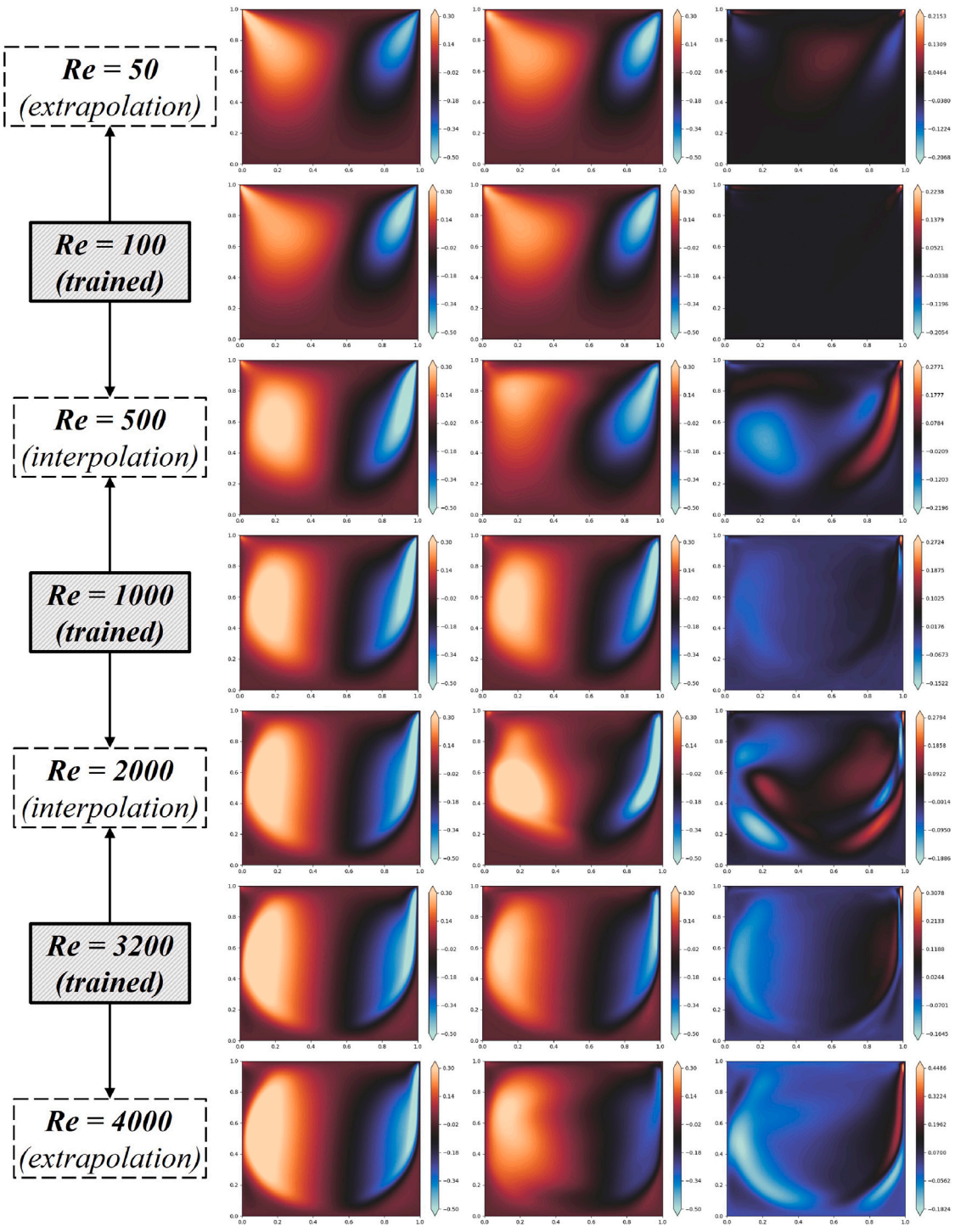


Fig. 11. Predicted y-velocity flow fields by purely data-driven NN for parametric NS equations. Note that only Re = 100, 1000, 3200 are trained and other Reynolds numbers are test cases. For three-column figures: (left column) DNS with 160 × 160 grids, (middle column) data-driven NN, (right column) error.

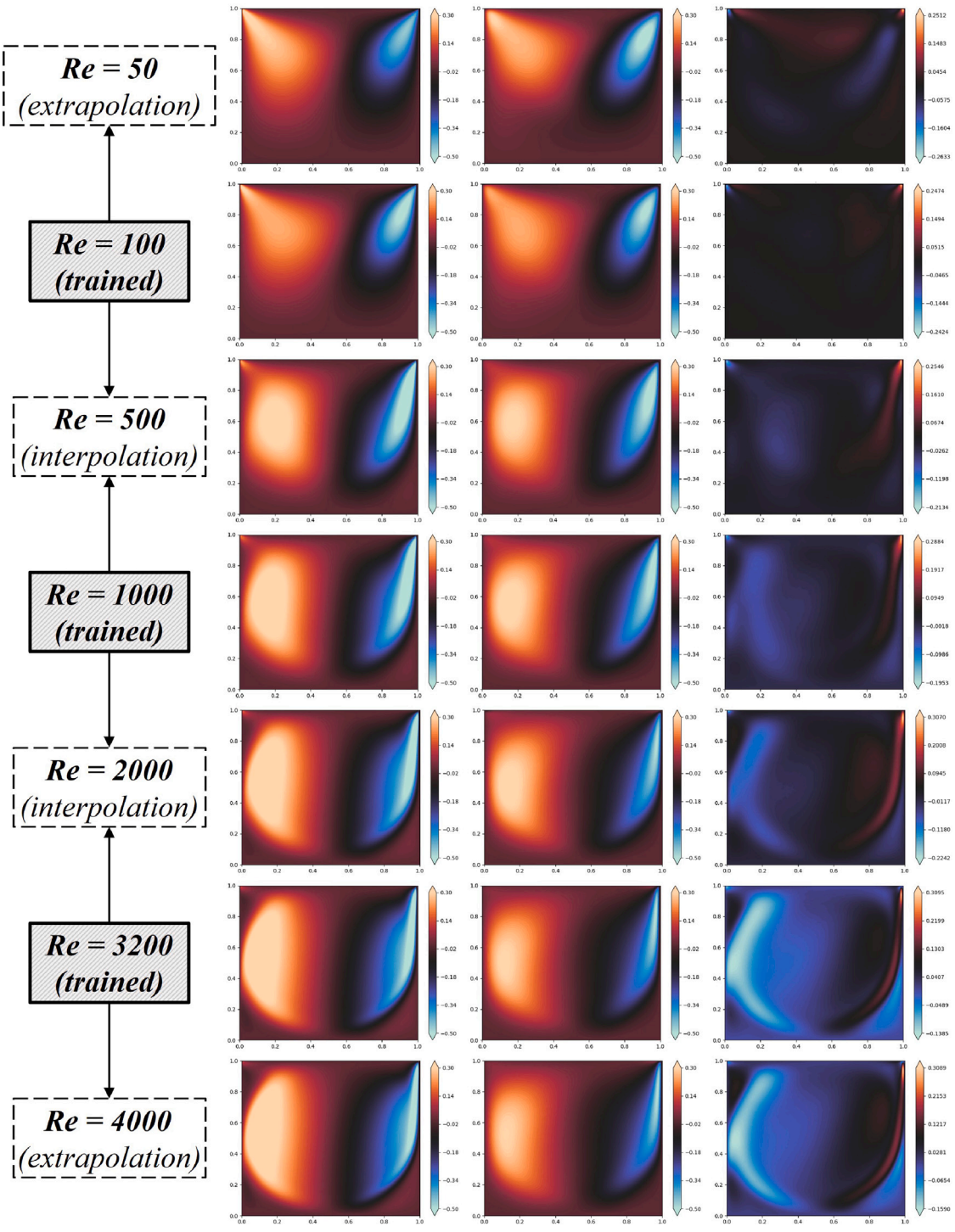


Fig. 12. Predicted y-velocity flow fields by DD-PINN for parametric NS equations. Note that only $Re = 100$, 1000, 3200 are trained and other Reynolds numbers are test cases. For three-column figures: (left column) DNS with 160 × 160 grids, (middle column) DD-PINN, (right column) error.

Table 8
One-way ANOVA results between four MF-DD-PINN cases: only p-values at high Reynolds indicate statistically significant differences among them.

		Reynolds number								
		50 (*ext*)	100 (***tr***)	500 (*int*)	1000 (***tr***)	2000 (*int*)	3200 (***tr***)	4000 (*ext*)	5000 (*ext*)	6000 (*ext*)
ANOVA	F statistic	2.1441	1.1710	0.8682	0.1433	2.4070	5.7041	6.4805	3.6144	0.7275
	p-value	0.0942	0.3205	0.4576	0.9339	0.0668	**0.0008**	**0.0003**	**0.0134**	0.5360

DD-PINNs in parametric NS is remarkable, since parametric DD-PINNs can be used to predict any untrained Re. The final results can be found in Table 6 and they demonstrate that the MF-DD-PINN case 4 achieves the highest accuracy at high Reynolds numbers from 2000 to 6000 among all trained models due to the highest-fidelity guide data at $Re = 3200$: it has half the RMSE compared to the single fidelity DD-PINN. Also, since four cases of MF-DD-PINNs are designed to have higher case numbers as the influence of the high-fidelity guide data at $Re = 3200$ increases, there is a clear decreasing trend of the RMSE as the case number increases at $Re = 3200, 4000, 5000$. The notable point here is that although case 3 and case 4 use only 25 additional guide data from the 160×160 grid, they show an impressive improvement over case 1 and case 2, underscoring the effectiveness of the multi-fidelity approach in DD-PINNs. Even in the low Reynolds region, MF-DD-PINN case 4 shows remarkable performance, indicating its general predictive performance over the different Reynolds numbers, both in interpolation and extrapolation tasks. Note that all MF-DD-PINN models do not show significantly better accuracies than DD-PINN in low Re regions, since their multi-fidelity approaches only aim to increase fidelity in high Re regions. To examine the effects of different multi-fidelity data on prediction accuracy, one-way ANOVA is performed within four cases of MF-DD-PINNs. Table 8 shows its results: the different types of data used for each case obviously affect the accuracy of MF-DD-PINNs in the high Reynolds region ($Re = 3200$, 4000, 5000). However, at $Re = 6000$, the p-value is greater than 0.5, indicating that the extrapolation of Re is too far from the training region in this case, and therefore the meaningful difference between the four cases disappears.

The flow fields of MF-DD-PINN case 4 are visualized in Fig. 13: when compared to DD-PINN in Fig. 12, they clearly show much more similar predictions with DNS at high Re values (since they maintain similar accuracy at low Re with DD-PINN, only high Reynolds regions are shown). The most fascinating point is that although only the range of $Re \in [50, 4000]$ is explored using collocation points for training MF-DD-PINN (as noted in Section 6.1), it successfully extrapolates the flow fields beyond $Re = 4000$, as can be seen in the flow fields at $Re = 5000$, 6000. For more specific investigation, velocity profiles at $x = 0.5$ and $y = 0.5$ from MF-DD-PINN case 4 are compared with DNS results, data-driven NN, and single-fidelity DD-PINN in Fig. 14. Multi-fidelity DD-PINN gives the most accurate results at the training point, $Re = 3200$, even when the velocity changes abruptly. On the other hand, data-driven NN and single-fidelity DD-PINN failed at local fitting even in the case of $Re = 3200$. As the Reynolds number increases towards the extrapolation region, data-driven NN fails to capture the global trend of the velocity profiles, while single-fidelity DD-PINN still follows the global trend of DNS — again highlighting that incorporating physics using PDE losses improves the generalized prediction performance. Meanwhile, MF-DD-PINN shows the best results: its y-velocity profile (left column) shows only slight deviation from DNS even at $Re = 6000$, while the x-velocity profile (right column) shows remarkable accuracy.[2] In summary, the above results demonstrate the scalability of DD-PINN, the framework newly proposed in our study: it can be extended to solve parametric PDEs, and furthermore, the multi-fidelity approach can even allow flexible fusion of guide data from different data sources, with impressive extrapolation performance than other approaches.

6.3. Uncertainty quantification for multi-fidelity DD-PINNs

In this section, we explore the potential of multi-fidelity DD-PINNs for quantifying predictive uncertainty, which is one of the most fundamental requirements for DT. For this purpose, we use the deep ensemble approach [14], which exploits the ensemble of NNs to quantify the predictive uncertainty of NNs: a total of four MF-DD-PINN case 2 models are trained with different random initializations. Then, the predictive value is determined by taking the average of the multiple models, while the epistemic uncertainty is calculated as the standard deviation within the models [60]. It is worth noting that for the efficient UQ of PINN models, the Monte Carlo dropout (MC-dropout) approach can be also easily incorporated [61]. However, due to the ongoing debate surrounding its interpretation as a true Bayesian inference method [14,62–65], we have chosen not to adopt this approach in the present study. Finally, the confidence intervals (CIs) estimated by the ensemble approach are plotted in Fig. 15 and there are two noteworthy points: one is that the uncertainty in terms of both velocity components shows an apparent diverging trend as it moves away from the trained Reynolds numbers (100, 1000, and 3200). However, the region between $Re = 100$ and $Re = 1000$ shows much more uncertainty than the region between $Re = 1000$ and $Re = 3200$: the reason seems to be that the change of the flow fields of the former is much more dramatic than the latter (see DNS results in Fig. 12). The second point is that as Re enters the region with negative values, the CI diverges dramatically compared to the other untrained regions (such as from $Re = 3200$ to $Re = 6000$). Considering that the multi-fidelity DD-PINNs explore only the physically realistic phenomena with the varying magnitude of Re for positive values, but not for negative values, the radically divergent CI trend in the non-physical $Re < 0$ region seems reasonable. In summary, from the qualitative analysis of the UQ performance of multi-fidelity DD-PINNs, we conclude that they have the strong potential to provide realistic predictive uncertainty, which enhances their scalability in DT.

[2] This is the reason why the RMSE of y-velocity is primarily used throughout this study to measure the accuracy of PINNs: y-velocity profiles are more difficult to predict than x-velocity profiles.

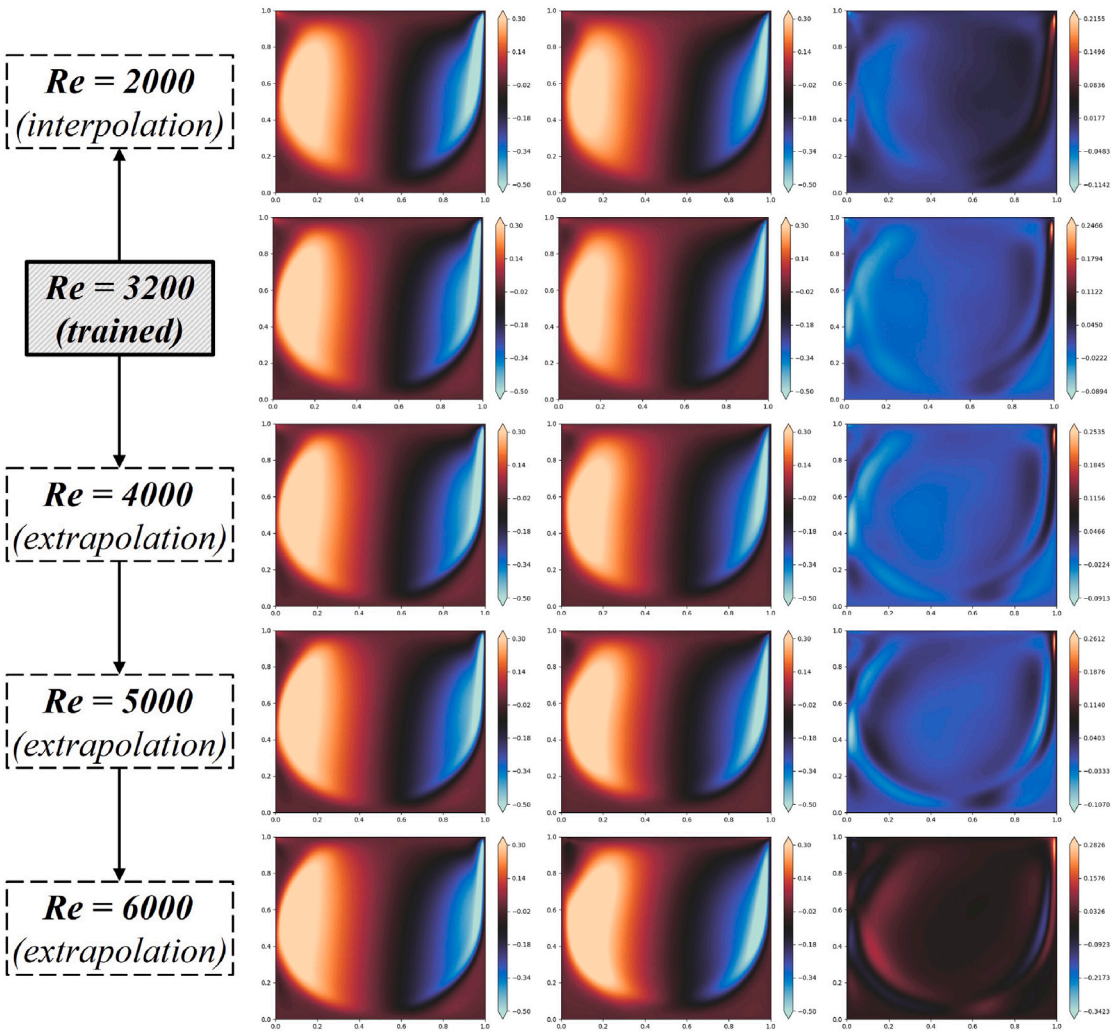
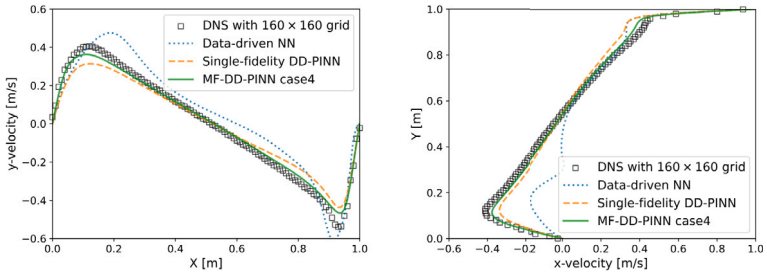


Fig. 13. Predicted y-velocity flow fields by MF-DD-PINN case 4 for parametric NS equations. Note that only Re = 100, 1000, and 3200 are trained and other Reynolds numbers are test cases. For three-column figures: (left column) DNS with 160 × 160 grids, (middle column) MF-DD-PINN case 4, (right column) error.

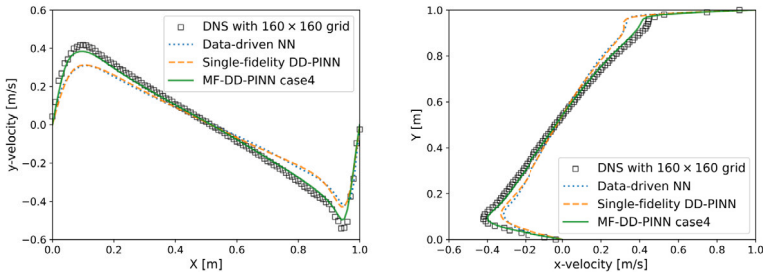
7. Conclusion

This study explored the potential of PINNs for the realization of DT from various perspectives. First, vorticity-aware adaptive sampling strategy tailored to fluid dynamics is proposed and comprehensively compared with existing sampling approaches to investigate the practical effectiveness of PINNs' mesh-free property in automating the construction of virtual space. Then, a data-driven physics-informed neural networks (DD-PINNs) framework, which can update the virtual model in a data-driven manner, is scrutinized as a remedy for the failure of data-free PINNs at high Reynolds in lid-driven cavity flow problem. Then, we extended DD-PINN to parametric PDEs – NS equations with varying Re for this study – and successfully demonstrated its superior performance than other conventional architectures including data-free PINNs and purely data-driven NNs. Moreover, its scalability to multi-fidelity guide data was also analyzed, considering that the datasets in physical space are often collected from different sources. Finally, multi-fidelity DD-PINN is applied to an ensemble-based UQ task in order to validate its applicability to DT, where an accurate measure of predictive uncertainty is necessary. The key findings of our study can be summarized as follows:

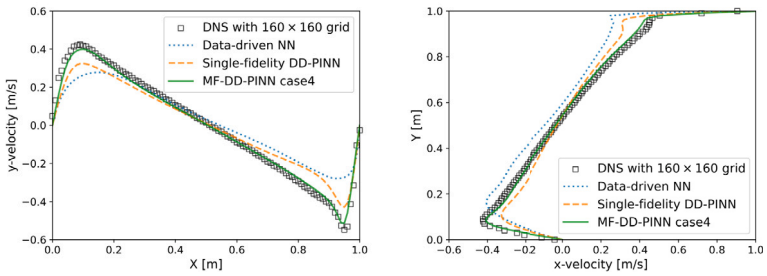
1. In data-free PINNs at Re = 100, vorticity-aware adaptive sampling showed the equivalent or even better performance than other conventional residual or gradient-aware samplings. It indicated that there is room for the development of flow-specific PINNs in that adaptive sampling based on specific flow quantities can be as accurate as existing sampling techniques.
2. However, as Re increased to 1000 and 3200, data-free PINNs failed to predict realistic flow fields, while the proposed DD-PINN framework succeeded. Since both data-free PINN and DD-PINN are designed to have the same model capacity, it can be concluded that the failure of data-free PINN at high Reynolds number is not due to insufficient capacity. Instead, with



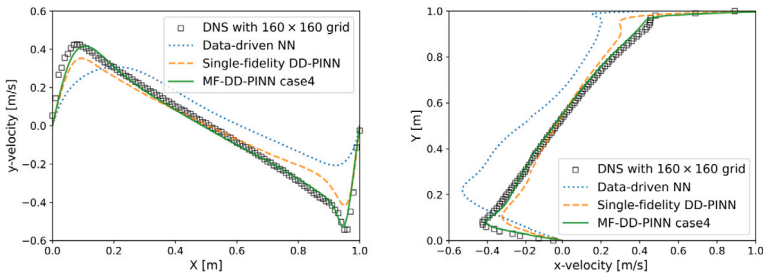
(a) $Re = 2000$ *(int)*



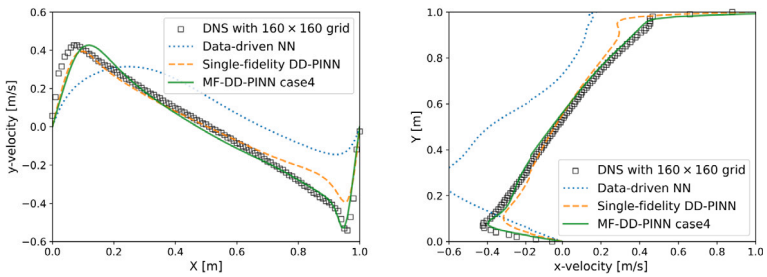
(b) $Re = 3200$ *(tr)*



(c) $Re = 4000$ *(ext)*



(d) $Re = 5000$ *(ext)*



(e) $Re = 6000$ *(ext)*

Fig. 14. Comparison of MF-DD-PINN case 4 with OpenFOAM DNS (160 × 160 grid) results: (left) y-velocity profile at $y = 0.5$, (right) x-velocity profile at $x = 0.5$. The results of data-driven NN and single fidelity DD-PINN in Section 6.1 are also shown.

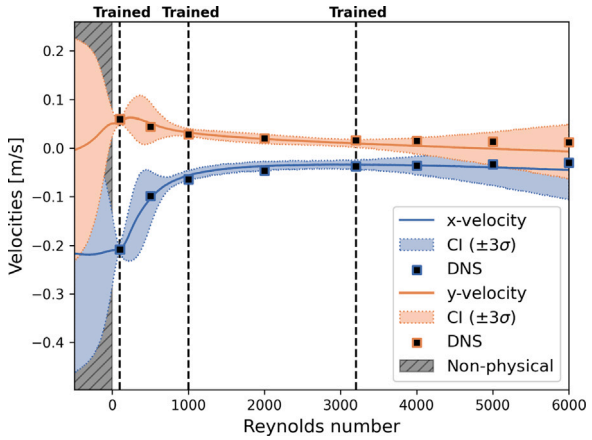


Fig. 15. Confidence intervals (CIs) of velocities at location ($x = 0.5, y = 0.5$) based on MF-DD-PINN case 2 model. At $Re = 100$, 1000, 3200, it shows a narrow CI because these points are used as training data, while the CI diverges in the unexplored Re region due to the absence of guide data. Also, the non-physical region ($Re < 0$) is hatched in gray, showing CIs that diverge much more than the other regions.

respect to the loss landscapes, DD-PINNs showed much sharper valley than data-free PINNs, explaining the reasons for their better performance. The superior performance of DD-PINNs highlights their applicability to DT, where data is consistently fed into the virtual space from the physical space.

3. In the DD-PINN framework, random sampling techniques that promote uniform collocation points outperformed other sampling approaches due to the need for a global calibrator to compensate for the local regularization effects of the guide data.
4. When extended to solve parametric PDEs, DD-PINNs outperformed purely data-driven NNs and data-free PINNs. It indicates that conventional data-free PINNs have obvious limitations in predicting more general physics while highlighting the scalability of the DD-PINNs to parametric problems. In this context, their potential to replace a demanding physical space with a real-time but generalized virtual representation in DT scenarios is verified.
5. The performance of DD-PINNs can be further improved with multi-fidelity guide data, which aims to efficiently train DD-PINNs with guide data from different fidelities. By increasing the weights of the data-driven loss term and increasing the fidelity of the guide data, the multi-fidelity approach achieves greater accuracy than any other compared model, even when high-fidelity guide data is sparse. It highlights its flexibility in utilizing the heterogeneous dataset acquired from physical space in DT.
6. When the multi-fidelity DD-PINN was used for uncertainty quantification, it provided realistic confidence intervals that were narrow in the trained region while diverging in the untrained regions. It also shows a dramatic divergence in the non-physical $Re < 0$ region, highlighting its potential to provide reliable predictive uncertainty over its prediction.

Although this work successfully demonstrated the flexibility of the DD-PINN framework in DT scenarios from multiple perspectives, the scope of the case study was limited to the lid-driven cavity flow problem. Therefore, the proposed DD-PINN would be extended to the unsteady forward problem, such as the flow over the cylinder. Nevertheless, as the concept of DD-PINN does not vary regardless of the flow unsteadiness, we believe that the DD-PINNs and their variants proposed in this study would still perform better than the data-free PINNs and leave it as a future study. Furthermore, the proposed PINNs would be tightly coupled with CFD simulations to collect guide data as in the study by Jeon et al. [66], and finally extended to inverse design optimization applications, where the acceleration of flow analysis is a key issue in terms of computational cost [49]. Lastly, although we have focused on investigating the flexibility and scalability of DD-PINNs from a macroscopic point of view, the next step may be more microscopic investigations on them, such as applying state-of-the-art PINN techniques such as weight adaptation, learning rate scheduling, and adaptive activation functions, etc [26,52,53,57,58].

CRediT authorship contribution statement

Sunwoong Yang: Supervision, Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing – original draft, Writing – review & editing. **Hojin Kim:** Software, Data curation, Writing – review & editing. **Yoonpyo Hong:** Conceptualization, Software, Writing – review & editing. **Kwanjung Yee:** Supervision, Funding acquisition. **Romit Maulik:** Supervision, Writing – review & editing. **Namwoo Kang:** Supervision, Funding acquisition.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Sunwoong Yang reports financial support was provided by National Research Foundation of Korea. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This work was supported by the National Research Foundation of Korea (2018R1A5A7025409), and the Ministry of Science and ICT of Korea (No. 2022-0-00969 and No. RS-2024-00355857).

Appendix A. Importance of capturing vorticity in fluid problems

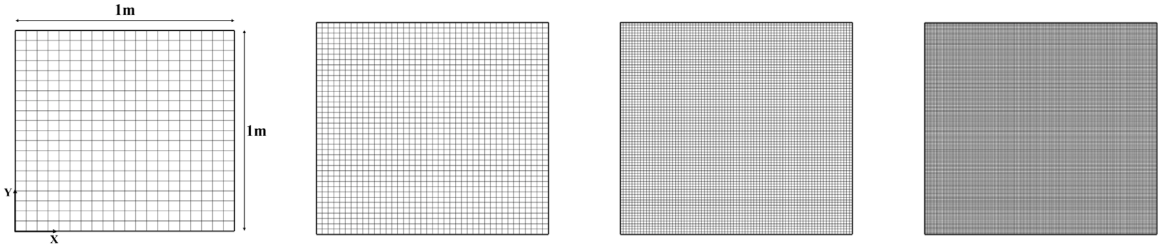
Vorticity is generated in regions where the flow properties change abruptly. To be more specific, it is concentrated in the region where vortex shedding occurs near the high angle of attack airfoil or in the tip vortices generated at the wingtips. These vortex flows generate induced velocity on the object itself so that when they interact with other objects, they not only affect aerodynamic performance but also have significant effects on subsequent structural and acoustic performance. Accordingly, numerous researchers have investigated numerical methods to accurately capture the strength of vortices with conventional CFD approaches by adding meshes in regions of high vorticity. To name a few, there were studies utilizing vorticity-related parameters such as Q-criterion or λ_2 [67–69]. In their work, flow analysis is performed first with a grid constructed without prior knowledge on the flow field. Vorticity-related parameters are then used to identify regions of concentrated vorticity, and new grid points are added to these regions accordingly. Finally, the flow analysis is performed again with the updated grid and this iterative process is repeated until satisfactory results are obtained.

Appendix B. Hyperparameter tuning results at $Re = 100\&3200$

See Table 9 for the results of hyperparameter tuning at $Re = 100$ and Table 10 at $Re = 3200$.

Appendix C. Visualization of grids used in openfoam and their validation

For validation of CFD simulations, results from OpenFOAM are compared with the data from Ghia et al. [59]. Uniform grids with four different densities are used as shown in Fig. 16. Then, comparisons of x-velocity profiles along the vertical center line ($x = 0.5$) and y-velocity profiles along the horizontal center line ($y = 0.5$) can be found in Fig. 17. For $Re = 100$, even the coarsest grid, 20×20, shows a satisfactory result. However, as the Reynolds number increases, results from coarser grids show bad correspondence with the validation data. This indicates that denser grids are required for better prediction of the flow field at higher Reynolds number regions where the thinning of boundary layers near walls becomes more dominant. In addition, at higher Re, a larger number of grids are required to accurately capture the near-linear velocity profile around the cavity center, where a uniform vorticity exists. For $Re = 1000$, 80×80 and 160×160 are indistinguishable from the validation data, while 40×40 shows almost the same trend. And for $Re = 3200$, both 80×80 and 160×160 show good agreement with the validation data.



(a) 20×20 grid **(b)** 40×40 grid **(c)** 80×80 grid **(d)** 160×160 grid

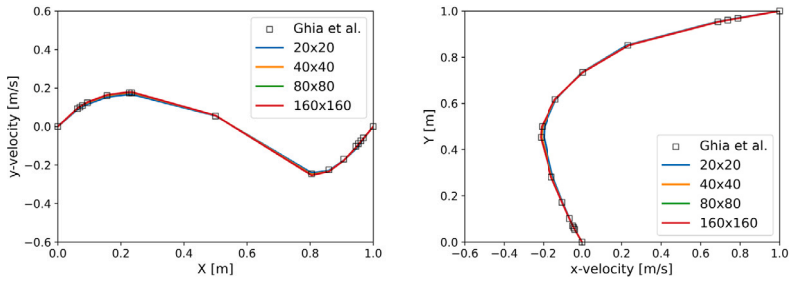
Fig. 16. Visualization of grids used for OpenFOAM analysis. The results from these grids are also utilized for DD-PINNs as guide data.

Table 9
RMSE of hyperparameter tuning in *Re=100* case.

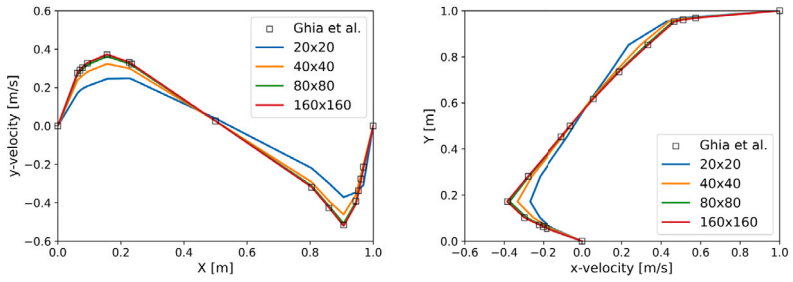
N_{layer}	N_{node}	lr	u	v
4	32	1e−3	1.21e−2	4.60e−3
		1e−4	1.58e−2	1.20e−2
		1e−5	2.02e−2	1.97e−2
	64	1e−3	1.06e−2	1.60e−3
		1e−4	1.37e−2	8.00e−3
		1e−5	1.57e−2	1.16e−2
	128	1e−3	1.08e−2	1.92e−3
		1e−4	1.28e−2	6.22e−3
		1e−5	1.52e−2	1.10e−2
6	**32**	**1e-3**	**1.01e−2**	**5.20e−4**
		1e−4	1.12e−2	2.47e−3
		1e−5	1.78e−2	1.76e−2
	64	1e−3	1.09e−2	2.10e−3
		1e−4	1.16e−2	3.61e−3
		1e−5	1.56e−2	1.16e−2
	128	1e−3	1.05e−2	1.04e−3
		1e−4	1.21e−2	4.79e−3
		1e−5	1.49e−2	1.06e−2
8	32	1e−3	**1.01e−2**	**5.20e−4**
		1e−4	1.12e−2	2.47e−3
		1e−5	1.78e−2	1.76e−2
	64	1e−3	1.09e−2	2.10e−3
		1e−4	1.16e−2	3.61e−3
		1e−5	1.56e−2	1.16e−2
	128	1e−3	1.05e−2	1.04e−3
		1e−4	1.21e−2	4.79e−3
		1e−5	1.49e−2	1.06e−2

Table 10
RMSE of hyperparameter tuning in *Re=3200* case.

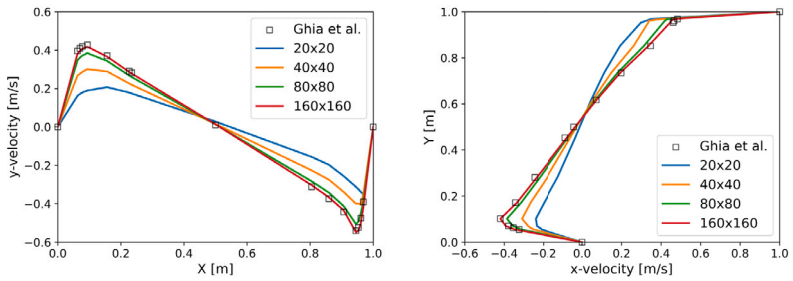
LN_{layer}	LN_{node}	Lw_{data}	u	Lv
5	32	1	1.42e−1	1.16e−1
		5	**1.35e−1**	1.08e−1
		10	1.38e−1	1.06e−1
	64	1	1.41e−1	1.17e−1
		5	1.37e−1	1.06e−1
		10	1.37e−1	1.05e−1
	128	1	1.38e−1	1.15e−1
		5	1.37e−1	1.04e−1
		10	1.37e−1	1.04e−1
7	**32**	1	1.42e−1	1.19e−1
		5	1.38e−1	1.08e−1
		10	1.36e−1	**1.03e−1**
	64	1	1.44e−1	1.25e−1
		5	1.40e−1	1.05e−1
		10	1.37e−1	1.07e−1
	128	1	1.39e−1	1.12e−1
		5	1.39e−1	1.11e−1
		10	1.38e−1	1.05e−1
9	32	1	1.41e−1	1.18e−1
		5	1.36e−1	1.04e−1
		10	1.37e−1	**1.03e−1**
	64	1	1.39e−1	1.13e−1
		5	1.36e−1	1.08e−1
		10	1.35e−1	1.07e−1
	128	1	1.51e−1	1.34e−1
		5	1.36e−1	1.07e−1
		10	1.37e−1	1.06e−1



(a) $Re = 100$



(b) $Re = 1000$



(c) $Re = 3200$

Fig. 17. Comparison of OpenFOAM results with Ghia et al. [59] based on different grid resolutions: (left) y-velocity, (right) x-velocity. Note that each velocity profile is calculated along $y = 0.5$ and $x = 0.5$, respectively.

Appendix D. Additional experiments for explaining the best performance of DD-PINNs with random sampling

See Table 11 to verify the effects of adaptive sampling approaches in DD-PINNs (Section 5.2), and Table 12 in DD-PINNs for parametric NS equations (Section 6.1).

Table 11
RMSE of y-velocity for different N/M values in adaptive sampling: DD-PINNs at $Re = 3200$ (Section 5.2).

	N/M			
	0.2	0.4	0.6	0.8
Van	**0.0509**	**0.0509**	**0.0509**	**0.0509**
RGV-100	0.0530	0.0534	0.0534	0.0516
RGV-010	0.0512	0.0539	0.0529	0.0526
RGV-001	0.0531	0.0528	0.0536	0.0541
RGV-110	0.0525	0.0563	0.0518	0.0520
RGV-101	0.0562	0.0538	0.0533	0.0538
RGV-011	0.0531	0.0544	0.0560	0.0529
RGV-111	0.0540	0.0545	0.0554	0.0527

Table 12
RMSE of y-velocity for different adaptive sampling techniques in DD-PINNs. Below the Reynolds numbers, *tr* indicates Reynolds used for the training, while *ext* and *int* indicate Reynolds to be tested by extrapolation and interpolation, respectively. Again, as in Table 4 and 11, the random sampling approach (Van) shows the best performance.

	Reynolds number								
	50 (*ext*)	100 (***tr***)	500 (*int*)	1000 (***tr***)	2000 (*int*)	3200 (***tr***)	4000 (*ext*)	5000 (*ext*)	6000 (*ext*)
Van	0.0229	**0.0093**	0.0367	**0.0269**	0.0454	**0.0530**	**0.0565**	**0.0589**	**0.0623**
RGV-100	0.0229	0.0096	0.0333	0.0282	0.0533	0.0566	0.0577	0.0611	0.0714
RGV-010	0.0216	**0.0092**	0.0279	0.0289	**0.0446**	0.0550	0.0611	0.0670	0.0724
RGV-001	0.0198	0.0097	0.0400	0.0310	0.0506	0.0579	0.0614	0.0654	0.0708
RGV-110	0.0196	0.0112	0.0297	0.0298	0.0470	0.0578	0.0634	0.0680	0.0722
RGV-101	**0.0185**	0.0098	**0.0270**	0.0317	0.0488	0.0575	0.0612	0.0633	0.0645
RGV-011	0.0189	0.0102	0.0293	0.0307	0.0480	0.0582	0.0630	0.0666	0.0693
RGV-111	0.0216	0.0127	0.0331	0.0344	0.0495	0.0584	0.061	0.0719	0.0759

Appendix E. Incorporating prior knowledge in DD-PINNs

The incorporation of prior knowledge in the construction of the DD-PINN framework is crucial for enhancing the model's efficiency and accuracy and there can be three exemplary situations that embedding prior information into the training of DD-PINN can be realized. (1) If one already knows that there will be a vortex at the corners of the square domain (such as lid-driven cavity flow case in this study), one can strategically place sensors at the corners to obtain new guide datasets that can effectively update the DD-PINNs in terms of vortex phenomena. (2) Knowing the characteristics of the flow to be predicted in advance can allow one to apply the appropriate adaptive sampling technique that is effective in that situation. For example, if we know that the adverse velocity gradient occurs along the y-direction, we can focus on using gradient-aware sampling, which only considers the gradient of the velocity with respect to the y-direction. (3) If BCs and ICs are known before the training of DD-PINNs, their information can be incorporated directly rather than relying solely on a data-driven approach to learn the boundary and initial conditions. In fact, in Section 6.1, we employed hard BCs and have found that the application of hard BCs actually improves the accuracy of DD-PINNs. Leveraging prior information in these three key aspects enables targeted data acquisition, informed sampling strategies, and the integration of physical constraints, ultimately unlocking the full potential of DD-PINNs in the context of DT.

References

[1] Michael G. Kapteyn, David J. Knezevic, Karen Willcox, Toward predictive digital twins via component-based reduced-order models and interpretable machine learning, in: AIAA Scitech 2020 Forum, 2020, p. 0418.
[2] Adil Rasheed, Omer San, Trond Kvamsdal, Digital twin: Values, challenges and enablers from a modeling perspective, IEEE Access 8 (2020) 21980–22012.
[3] Ricardo Vinuesa, Steven L. Brunton, Beverley J. McKeon, The transformative potential of machine learning for experiments in fluid mechanics, Nat. Rev. Phys. 5 (9) (2023) 536–545.
[4] Mohsen Attaran, Bilge Gokhan Celik, Digital twin: Benefits, use cases, challenges, and opportunities, Decis. Anal. J. (2023) 100165, http://dx.doi.org/10.1016/j.dajour.2023.100165.
[5] Fei Tao, He Zhang, Ang Liu, A.Y.C. Nee, Digital twin in industry: State-of-the-art, IEEE Trans. Ind. Inform. 15 (4) (2019) 2405–2415, http://dx.doi.org/10.1109/TII.2018.2873186.
[6] National Academy of Engineering and National Academies of Sciences and Engineering and Medicine, Foundational Research Gaps and Future Directions for Digital Twins, The National Academies Press, Washington, DC, ISBN: 978-0-309-71169-2, 2023, http://dx.doi.org/10.17226/26894, URL https://nap.nationalacademies.org/catalog/26894/foundational-research-gaps-and-future-directions-for-digital-twins.
[7] Luning Li, Sohaib Aslam, Andrew Wileman, Suresh Perinpanayagam, Digital twin in aerospace industry: A gentle introduction, IEEE Access 10 (2022) 9543–9562, http://dx.doi.org/10.1109/ACCESS.2021.3136458.
[8] Steven A. Niederer, Michael S. Sacks, Mark Girolami, Karen Willcox, Scaling digital twins from the artisanal to the industrial, Nat. Comput. Sci. 1 (5) (2021) 313–320.
[9] Omer San, The digital twin revolution, Nat. Comput. Sci. 1 (5) (2021) 307–308.
[10] Hoe-Han Goh, Ricardo Vinuesa, Regulating artificial-intelligence applications to achieve the sustainable development goals, Discov. Sustain. 2 (2021) 1–6.
[11] Rui Zhang, Fang Wang, Jun Cai, Yan Wang, Hongfei Guo, Jingsha Zheng, Digital twin and its applications: A survey, Int. J. Adv. Manuf. Technol. 123 (11–12) (2022) 4123–4136, http://dx.doi.org/10.1007/s00170-022-10445-3.
[12] Maziar Raissi, Paris Perdikaris, George E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707.
[13] Sunwoong Yang, Kwanjung Yee, Design rule extraction using multi-fidelity surrogate model for unmanned combat aerial vehicles, J. Aircr. (2022) 1–15.
[14] Sunwoong Yang, Kwanjung Yee, Towards reliable uncertainty quantification via deep ensemble in multi-output regression task, Engineering Applications of Artificial Intelligence 132 (2024) 107871.
[15] Sifan Wang, Paris Perdikaris, Deep learning of free boundary and stefan problems, J. Comput. Phys. 428 (2021) 109914.
[16] Atakan Aygun, Romit Maulik, Ali Karakus, Physics-informed neural networks for mesh deformation with exact boundary enforcement, 2023, arXiv preprint arXiv:2301.05926.
[17] Stefano Markidis, The old and the new: Can physics-informed deep-learning replace traditional linear solvers? Front. Big Data 4 (2021) 669097.
[18] Luning Sun, Han Gao, Shaowu Pan, Jian-Xun Wang, Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data, Comput. Methods Appl. Mech. Engrg. 361 (2020) 112732.
[19] George Em Karniadakis, Ioannis G. Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, Liu Yang, Physics-informed machine learning, Nat. Rev. Phys. 3 (6) (2021) 422–440.
[20] Lu Lu, Xuhui Meng, Zhiping Mao, George Em Karniadakis, DeepXDE: A deep learning library for solving differential equations, SIAM Rev. 63 (1) (2021) 208–228.

[21] Modulus - A neural network framework — developer.nvidia.com, 2023, https://developer.nvidia.com/modulus. (Accessed 20 December 2023).
[22] Cahya Amalinadhi, Pramudita S. Palar, Rafael Stevenson, Lavi Zuhal, On physics-informed deep learning for solving Navier-Stokes equations, in: AIAA SCITECH 2022 Forum, 2022, p. 1436.
[23] Ameya D. Jagtap, Ehsan Kharazmi, George Em Karniadakis, Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems, Comput. Methods Appl. Mech. Engrg. 365 (2020) 113028.
[24] Shirong Li, Xinlong Feng, Dynamic weight strategy of physics-informed neural networks for the 2D Navier–Stokes equations, Entropy 24 (9) (2022) 1254.
[25] Xiao-dong Bai, Yong Wang, Wei Zhang, Applying physics informed neural network for flow data assimilation, J. Hydrodyn. 32 (2020) 1050–1058.
[26] Sifan Wang, Yujun Teng, Paris Perdikaris, Understanding and mitigating gradient flow pathologies in physics-informed neural networks, SIAM J. Sci. Comput. 43 (5) (2021) A3055–A3081.
[27] Jian Cheng Wong, Chinchun Ooi, Abhishek Gupta, Yew-Soon Ong, Learning in sinusoidal spaces with physics-informed neural networks, IEEE Trans. Artif. Intell. (2022).
[28] Pao-Hsiung Chiu, Jian Cheng Wong, Chinchun Ooi, My Ha Dao, Yew-Soon Ong, CAN-PINN: A fast physics-informed neural network based on coupled-automatic–numerical differentiation method, Comput. Methods Appl. Mech. Engrg. 395 (2022) 114909.
[29] Aditi Krishnapriyan, Amir Gholami, Shandian Zhe, Robert Kirby, Michael W. Mahoney, Characterizing possible failure modes in physics-informed neural networks, Adv. Neural Inf. Process. Syst. 34 (2021) 26548–26560.
[30] Pi-Yueh Chuang, Lorena A. Barba, Predictive limitations of physics-informed neural networks in vortex shedding, 2023, arXiv preprint arXiv:2306.00230.
[31] Franz M. Rohrhofer, Stefan Posch, Clemens Gößnitzer, Bernhard C. Geiger, On the role of fixed points of dynamical systems in training physics-informed neural networks, 2022, arXiv preprint arXiv:2203.13648.
[32] Vignesh Gopakumar, Stanislas Pamela, Debasmita Samaddar, Loss landscape engineering via data regulation on PINNs, Mach. Learn. Appl. 12 (2023) 100464.
[33] Fei Tao, He Zhang, Ang Liu, Andrew Y.C. Nee, Digital twin in industry: State-of-the-art, IEEE Trans. Ind. Inform. 15 (4) (2018) 2405–2415.
[34] Bo Wang, Zengcong Li, Ziyu Xu, Zhiyong Sun, Kuo Tian, Digital twin modeling for structural strength monitoring via transfer learning-based multi-source data fusion, Mech. Syst. Signal Process. 200 (2023) 110625.
[35] Kazuto Hasegawa, Kai Fukami, Takaaki Murata, Koji Fukagata, Machine-learning-based reduced-order modeling for unsteady flows around bluff bodies of various shapes, Theor. Comput. Fluid Dyn. 34 (2020) 367–383.
[36] Romit Maulik, Bethany Lusch, Prasanna Balaprakash, Reduced-order modeling of advection-dominated systems with recurrent neural networks and convolutional autoencoders, Phys. Fluids 33 (3) (2021).
[37] Yu-Eop Kang, Sunwoong Yang, Kwanjung Yee, Physics-aware reduced-order modeling of transonic flow via β-variational autoencoder, Phys. Fluids 34 (7) (2022) 076103.
[38] Atilim Gunes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, Jeffrey Mark Siskind, Automatic differentiation in machine learning: a survey, J. Marchine Learn. Res. 18 (2018) 1–43.
[39] Lu Lu, Raphael Pestourie, Wenjie Yao, Zhicheng Wang, Francesc Verdugo, Steven G. Johnson, Physics-informed neural networks with hard constraints for inverse design, SIAM J. Sci. Comput. 43 (6) (2021) B1105–B1132.
[40] Myeong-Seok Go, Jae Hyuk Lim, Seungchul Lee, Physics-informed neural network-based surrogate model for a virtual thermal sensor with real-time simulation, Int. J. Heat Mass Transfer 214 (2023) 124392.
[41] Hamidreza Eivazi, Ricardo Vinuesa, Physics-informed deep-learning applications to experimental fluid mechanics, 2022, arXiv preprint arXiv:2203.15402.
[42] Hongping Wang, Yi Liu, Shizhao Wang, Dense velocity reconstruction from particle image velocimetry/particle tracking velocimetry using a physics-informed neural network, Phys. Fluids 34 (1) (2022).
[43] Gazi Hasanuzzaman, Hamidreza Eivazi, Sebastian Merbold, Christoph Egbers, Ricardo Vinuesa, Enhancement of PIV measurements via physics-informed neural networks, Meas. Sci. Technol. 34 (4) (2023) 044002.
[44] Elijah Ang, Bing Feng Ng, Physics-informed neural networks for flow around airfoil, in: AIAA SCITECH 2022 Forum, 2022, p. 0187.
[45] Salim M. Salim, S. Cheah, Wall y strategy for dealing with wall-bounded turbulent flows, in: Proceedings of the International Multiconference of Engineers and Computer Scientists, Vol. 2, 2009, pp. 2165–2170.
[46] Mohammad Amin Nabian, Rini Jasmine Gladstone, Hadi Meidani, Efficient training of physics-informed neural networks via importance sampling, Comput.-Aided Civ. Infrastruct. Eng. 36 (8) (2021) 962–977.
[47] Zhiping Mao, Ameya D. Jagtap, George Em Karniadakis, Physics-informed neural networks for high-speed flows, Comput. Methods Appl. Mech. Engrg. 360 (2020) 112789.
[48] Chenxi Wu, Min Zhu, Qinyang Tan, Yadhu Kartha, Lu Lu, A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks, Comput. Methods Appl. Mech. Engrg. 403 (2023) 115671.
[49] Sunwoong Yang, Sanga Lee, Kwanjung Yee, Inverse design optimization framework via a two-step deep learning approach: application to a wind turbine airfoil, Engineering with Computers 39 (3) (2023) 2239–2255.
[50] Qiong Liu, F. Gómez, J.M. Perez, Vassilios Theofilis, Instability and sensitivity analysis of flows using OpenFOAM®, Chin. J. Aeronaut. 29 (2) (2016) 316–325.
[51] Sifan Wang, Shyam Sankaran, Hanwen Wang, Paris Perdikaris, An expert's guide to training physics-informed neural networks, 2023, arXiv preprint arXiv:2308.08468.
[52] Ameya D. Jagtap, Kenji Kawaguchi, George Em Karniadakis, Locally adaptive activation functions with slope recovery for deep and physics-informed neural networks, Proc. R. Soc. Lond. Ser. A Math. Phys. Eng. Sci. 476 (2239) (2020) 20200334.
[53] Ameya D. Jagtap, Kenji Kawaguchi, George Em Karniadakis, Adaptive activation functions accelerate convergence in deep and physics-informed neural networks, J. Comput. Phys. 404 (2020) 109136.
[54] Gary W. Heiman, Understanding Research Methods and Statistics: An Integrated Introduction for Psychology, Houghton, Mifflin and Company, 2001.
[55] Ian J. Goodfellow, Oriol Vinyals, Andrew M. Saxe, Qualitatively characterizing neural network optimization problems, 2014, arXiv preprint arXiv:1412.6544.
[56] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, Tom Goldstein, Visualizing the loss landscape of neural nets, Adv. Neural Inf. Process. Syst. 31 (2018).
[57] Levi McClenny, Ulisses Braga-Neto, Self-adaptive physics-informed neural networks using a soft attention mechanism, 2020, arXiv preprint arXiv:2009.04544.
[58] Colby L. Wight, Jia Zhao, Solving allen-cahn and cahn-hilliard equations using the adaptive physics informed neural networks, 2020, arXiv preprint arXiv:2007.04542.
[59] UK.N.G. Ghia, Kirti N. Ghia, C.T. Shin, High-re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method, J. Comput. Phys. 48 (3) (1982) 387–411.
[60] Masaki Morimoto, Kai Fukami, Romit Maulik, Ricardo Vinuesa, Koji Fukagata, Assessments of epistemic uncertainty using Gaussian stochastic weight averaging for fluid-flow regression, Physica D 440 (2022) 133454.
[61] Yarin Gal, Zoubin Ghahramani, Dropout as a bayesian approximation: Representing model uncertainty in deep learning, in: International Conference on Machine Learning, PMLR, 2016, pp. 1050–1059.
[62] Ian Osband, Risk versus uncertainty in deep learning: Bayes, bootstrap and the dangers of dropout, in: NIPS Workshop on Bayesian Deep Learning, Vol. 192, 2016.
[63] Jiri Hron, Alexander G. de G. Matthews, Zoubin Ghahramani, Variational Gaussian dropout is not Bayesian, 2017, arXiv preprint arXiv:1711.02989.

[64] Jiri Hron, Alex Matthews, Zoubin Ghahramani, Variational Bayesian dropout: pitfalls and fixes, in: International Conference on Machine Learning, PMLR, 2018, pp. 2019–2028.
[65] Loic Le Folgoc, Vasileios Baltatzis, Sujal Desai, Anand Devaraj, Sam Ellis, Octavio E Martinez Manzanera, Arjun Nair, Huaqi Qiu, Julia Schnabel, Ben Glocker, Is MC dropout bayesian? 2021, arXiv preprint arXiv:2110.04286.
[66] Joongoo Jeon, Juhyeong Lee, Ricardo Vinuesa, Sung Joong Kim, Residual-based physics-informed transfer learning: A hybrid method for accelerating long-term CFD simulations via deep learning, Int. J. Heat Mass Transfer 220 (2024) 124900.
[67] S.J. Kamkar, Andrew M. Wissink, Venkateswaran Sankaran, Antony Jameson, Feature-driven cartesian adaptive mesh refinement for vortex-dominated flows, J. Comput. Phys. 230 (16) (2011) 6271–6298.
[68] Mitsuhiro Murayama, Kazuhiro Nakahashi, Keisuke Sawada, Simulation of vortex breakdown using adaptive grid refinement with vortex-center identification, AIAA J. 39 (7) (2001) 1305–1312.
[69] Min Kyu Jung, Je Young Hwang, Oh Joon Kwon, Assessment of rotor aerodynamic performances in hover using an unstructured mixed mesh method, in: 52nd AIAA Aerospace Sciences Meeting, 2014, p. 0042.