On the use of Fourier Features-Physics Informed Neural Networks (FF-PINN) for forward and inverse fluid mechanics problems

Article *in* Proceedings of the Institution of Mechanical Engineers Part M Journal of Engineering for the Maritime Environment · April 2023

DOI: 10.1177/14750902231166424

CITATIONS
16

READS
1,547

2 authors, including:

Omar Sallam
Texas A&M University
20 PUBLICATIONS **89** CITATIONS

SEE PROFILE

Original Article

Institution of MECHANICAL ENGINEERS

Journal of ENGINEERING FOR THE MARITIME ENVIRONMENT

Proc IMechE Part M:
J Engineering for the Maritime Environment
2023, Vol. 237(4) 846–866

DOI: 10.1177/14750902231166424
journals.sagepub.com/home/pim
SAGE

On the use of Fourier Features-Physics Informed Neural Networks (FF-PINN) for forward and inverse fluid mechanics problems

Omar Sallam and Mirjam Fürth

Abstract

Physics Informed Neural Networks (PINN), a deep learning tool, has recently become an effective method for solving inverse Partial Differential Equations (PDEs) where the boundary/initial conditions are not well defined and only noisy sparse measurements sampled in the domain exist. PINN, and other Neural Networks, tends to converge to the low frequency solution in a field that has multiple frequency scales, this is known as spectral bias. For PINN this happens when solving PDEs that exhibit periodic behavior spatially and temporally with multi frequency scales. Previous studies suggested that Fourier Features-Neural Networks (FF-NN) can be used to overcome the spectral bias problem. They proposed the Multi Scale-Spatio Temporal-Fourier Features-Physics Informed Neural Networks (MS-ST-FF-PINN) to overcome the spectral bias problem in PDEs solved by PINN. This has been evaluated on basic PDEs such as Poisson, wave and Gray-Scott equations. In this paper we take MS-ST-FF-PINN a step further by applying it to the incompressible Navier-Stokes equations. Furthermore, a comparative analysis between the PINN and the MS-ST-FF-PINN architectures solution accuracy, the learnt frequency components and the rate of convergence to the correct solution is included. To show this three test cases are shown (a)-Forward time independent double-lid-driven cavity, (b)-Inverse time independent free surface estimation of Kelvin wave pattern, and (c)-Inverse 2D time-dependent turbulent Von Karman vortex shedding interaction downstream of multiple cylinders. The results show that MS-ST-FF-PINN is better at learning low and high frequency components synchronously at early training iterations compared to the PINN architecture that does not learn the high frequency components even after multiple iteration numbers such as the Kelvin wave pattern and the Karman vortex shedding cases. However, for the third test case, the MS-ST-FF-PINN architecture showed a discontinuity for the temporal prediction of the pressure field due to over-fitting.

Keywords

PINN, Fourier Features Neural Network, incompressible flow, spectral bias, deep learning, FEM, CFD, free surface, OpenFOAM, Volume of Fluid

Date received: 1 September 2022; accepted: 8 March 2023

Introduction

Raissi et al.[1] introduced Physics Informed Neural Network (PINN) as a new deep learning framework to solve supervised learning tasks while respecting any given law of physics described by general nonlinear Partial Differential Equations (PDEs). The PINN model can be used to solve forward problems, where the boundary/initial conditions of the given PDE are well defined, or can be used to solve inverse problems (data driven discovery problem) where sparse measurement of the solution scatter in the spatio temporal domain and some of the boundary/initial conditions are not well defined or some of the PDE parameters are unknown. As an example of using PINN for solving inverse problems, Raissi[2] showed the ability of the PINN model to evaluate the fluid flow density and dynamic viscosity for vortex shedding of

Ocean Engineering Department, Texas A&M University, College Station, TX, USA

Corresponding author:
Omar Sallam, Ocean Engineering Department, Texas A&M University, 400 Bizzell St, College Station, TX 77843, USA.
Email: osallam@tamu.edu

incompressible transient flow downstream of cylinder based on sparse measurement of the velocity components in the spatio temporal domain. The model was able to infer the pressure field from the measured velocity field.

One of the strengths of this new paradigm is the ability of the Neural Network to infer the solution of multiple physical variables based on sparse measurements of other variable/s encapsulated in the set of PDEs. Raissi et al.[3] used PINN to infer the solution of the velocity components and the pressure for incompressible flow over cylinder from numerical measurements of a scalar quantity (die) injected from the numerical tunnel, by integration of the inferred pressure field the lift and drag forces and validated with Direct Numerical Simulations (DNS). Hence, this paradigm can be a substitute to the invasive lift and drag measurement devices in wind/water tunnels. The authors also tested the PINN model on a 3D internal flow problem in the intracranial aneurysm, where the PINN model was trained on the injected scalar quantity (die) for only two orthogonal planes; the model successfully predicted the scalar quantity, velocity components and the pressure field in the rest of the complex intracranial aneurysm geometry in addition to the wall shear stresses. The study comprehensively presented the effect of the sampling points density and the sampling frequency on the Neural Network prediction accuracy for the flow fields.

Dealing with problems that include periodic behaviors of multiple frequencies is challenging for Neural Networks due to the spectral bias phenomenon.[4] Spectral bias is the tendency of the Neural Network to converge toward the low frequency solution components faster than to the higher frequency solution components, in addition, convergence to the high frequency solution components is not guaranteed even with a large number of iterations.[5] For problems that exhibit periodic behaviors with multi scale frequency components in spatio temporal domains, such as vortex induced vibration problems[6,7] and floating structures subjected to irregular waves,[8] PINN architecture faces the spectral bias problem. Tancik et al.[5] proposed the use of Fourier Features-Neural Network (FF-NN) to tackle the spectral bias problem for any general Neural Network. In FF-NN, the inputs of the network are mapped/transformed to a vector of periodic functions (sine and cosine). This mapping/transformation layer is called Fourier Feature embeddings. The frequency vector elements of the Fourier Feature embedding layer were sampled from a normal distribution. The authors successfully applied the Fourier Features embedding hypothesis on high frequency colored images reconstruction Neural Networks that was able to reconstruct the high frequency component in the images such as hair and fur, while the conventional Neural Network failed to do even with large iteration numbers. Built on this, Wang et al.[9] proposed the Multi Scale-spatio temporal-Fourier Feature-Physics Informed Neural Network (MS-ST-FF-PINN) to overcome the spectral bias problem in PDEs. The authors tested the MS-ST-FF-PINN on basic PDEs such as Poisson, wave and Gray Scott equations and showed the Network successfully learned the multi scale frequency solution components of the mentioned PDEs. However, MS-ST-FF-PINN has not been tested yet on more complex and nonlinear PDEs such as the Navier-Stokes equations. Hence, using the Fourier Features embedding hypothesis in the PINN architecture could be an improvement, especially for turbulent flow problems where spatial and temporal multi frequency scales exist.

In this paper, we apply the MS-ST-FF-PINN architecture proposed by Wang et al.[9] on fluid mechanics applications that exhibit periodic behavior temporally or spatially, we focus on the incompressible fluid flow equation set of continuity and momentum equations. The mathematical model behind the PINN and the MS-ST-FF-PINN are presented in the methodology section. Then three test cases are presented to test the behavior of PINN and MS-ST-FF-PINN architectures, the three test cases are: (a)-Forward time independent double-lid-driven cavity, (b)-Inverse time independent free surface estimation of Kelvin wave pattern, and (c)-Inverse 2D time dependent turbulent Von Karman vortex shedding interaction downstream of multiple cylinders. Deepxde python library[10] is used for these deep learning simulations. All Neural Network training are implemented on a Nvidia Quadro RTX 4000 GPU with 8 GB dedicated memory.

Methodology

Physics Informed Neural Network (PINN)

Physics Informed Neural Network (PINN) is a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations.[11] PINNs are trained to solve supervised learning tasks while respecting given laws of physics described by general nonlinear partial differential equations such as mass conservation, energy conservation or any other physical laws.[1,2]

Suppose a physical problem described by a nonlinear Partial Differential Equation (PDE) shown in equation (1) with exact solution $u(x, t)$. $N[\cdot]$ is a nonlinear differential operator. The PDE is bounded in the temporal domain $[0, T]$ and spatial domain Ω with spatial domain boundary $\partial\Omega$, initial condition u_i and boundary condition u_b. The Neural Network predicted (approximated) solution $u_net(t, x.\theta)$ is achieved by training the Neural Network to optimize the Neural Network trainable parameters $\theta(\mathbf{W}, \mathbf{b})$ to satisfy the PDE at residual points in the spatio temporal domain, satisfy the boundary/initial conditions and satisfy any extra sampled solution in the spatio temporal domain. Equation (2) shows the total weighted loss function (objective function) to be minimized by the Neural Network, the subscripts r, i, b, and s stands for residual, initial, boundary and sampled points respectively.

PINN is a meshless technique since there is no need for computational mesh (grid) to compute the temporal derivative $\frac{\partial}{\partial t}$ or the spatial derivatives of the nonlinear operator $N[\cdot]$ but it uses the automatic differentiation (AD)[12] to compute these spatio temporal derivatives in the domain at the residual points, these residual points densely span the spatio temporal domain to satisfy the governing PDE (minimize the PDEs residuals). Automatic Differentiation (AD)[12] is available in the well known deep learning open source codes such as TensorFlow[13] and PyTorch.[14]

$$\begin{aligned} \tfrac{\partial u}{\partial t} + N[u] &= 0; \quad x \in \Omega, t \in [0, T] \\ u(t = 0, x) &= u_0; \quad \forall x \in \Omega \\ u(t, x) &= u_b; \quad \forall x \in \partial\Omega, t \in [0, T] \end{aligned} \tag{1}$$

$$\begin{aligned} Total\ loss = \quad & W_r || \tfrac{\partial u_{net}}{\partial t} + N[u_{net}] ||^2_{t\in[0,\,T],\ x\in\Omega} \\ + \quad & W_i || u_{net} - u_i ||^2_{t\,=\,0,\,x\in\Omega} \\ + \quad & W_b || u_{net} - u_b ||^2_{t\in[0,\,T],\ x\in\partial\Omega} \\ + \quad & W_s || u_{net} - u_s ||^2_{t\in[0,\,T],\ x\in\Omega} \end{aligned} \tag{2}$$

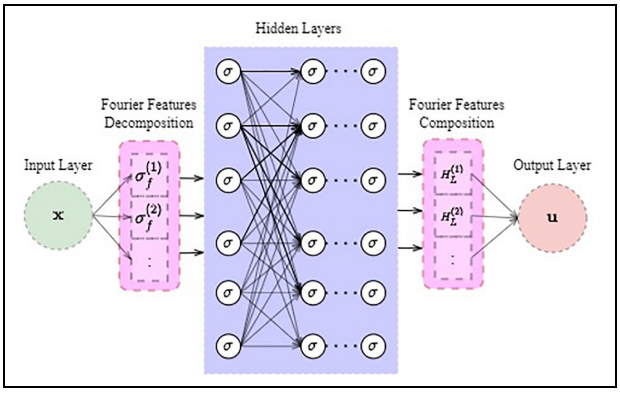

Figure 1. Multi Scale-Fourier Feature-Neural Network (MS-FF-NN) architecture. The inputs **x** to the NN is transformed to a series of sinusoidal functions with frequencies sampled from normal distribution by the Fourier Features decomposition layer. In the Fourier Features composition layer, signals superposition is implemented to obtain the NN output **u**. MS-FF-NN architecture helps to avoid the spectral bias problem in NNs.

Multi scale-Fourier Feature-Neural Network (MS-FF-NN)

Spectral bias phenomenon is the tendency of Neural Networks to converge toward the low frequency solutions.[4] As a results of this phenomenon, if a problem has a solution composed of multiple modulated signals, the NN is converging faster toward the low frequency signal at the early training iterations, then at later iterations the NN is slowly converging to the modulated solution (low + high frequency signals). It should also be mentioned that convergence to the higher frequency signals is not guaranteed even with large training iterations.[5] One approach to overcome the spectral bias phenomenon is applying linear transformation to the Neural Network input layer, the aim of this input transformation is to transform the NN input to higher frequency functions that can speed up the NN convergence toward the high frequency solutions.[5] The Neural Networks with the input transformation layer are called Fourier Feature-Neural Networks (FF-NN), the FF-NN architecture is shown in Figure 1.

The input transformation layer $\phi_E^{(i)}$ transforms the input layer **x** to a higher frequency function as shown in equation (3), where $\mathbf{f^{(i)}}$ is the frequency vector of the transformed input layer and its values are sampled from the normal distribution $N(0, \sigma_f^{(i)})$, for $i > 1$ the architecture is called Multi scale-Fourier Feature-Neural Network (MS-FF-NN) since the frequency vector components are sampled from multiple normal distribution.[9] $\sigma_f^{(i)}$ is not a trainable parameter, but its selection is problem dependent,[5] for problems that exhibit high frequency solution, the inputs should be encoded in the Fourier Feature layer with frequency vector $\mathbf{f^i}$ sampled from normal distributions with high standard deviation values $\sigma_f^{(i)}$ and vise versa, the derivation of the relation between the $\sigma_f^{(i)}$ and the solution dominant frequency is presented in Wang et al.[9]

The hidden layers of the Fourier Feature-Neural Network (FF-NN) have the same structure as the conventional NN as shown in Figure 1 and equation (5) except for the first hidden layer $H_1^{(i)}$ where the nonlinear activation function σ is applied to the weighted Fourier Feature transformed input $\phi_E^{(i)}$ instead of directly applying to the weighted input **x** in the conventional Neural Network as shown in equation (4). **W** and **b** are the Neural Network weights and biases.

The Neural Network output (predicted solution) $\mathbf{u_{net}(x, \theta)}$ is the linear combination of the last hidden layer for all frequency bands as shown in the Fourier Feature composition layer in Figure 1 and in equation (6), where θ is the Neural Network trainable parameters including the weights **W** and the biases **b**.

$$\phi_E^{(i)}(\mathbf{x}) = [sin(2\pi \mathbf{f^{(i)}} \times \mathbf{x}); cos(2\pi \mathbf{f^{(i)}}) \times \mathbf{x})]^T \tag{3}$$

$$\mathbf{H}_1^{(i)} = \sigma(\mathbf{W_1} \cdot \phi_E^{(i)}(\mathbf{x}) + \mathbf{b_1}); i = 1, 2, ..., M \tag{4}$$

$$\mathbf{H}_l^{(i)} = \sigma(\mathbf{W}_l \cdot \mathbf{H}_{l-1}^{(i)} + \mathbf{b}_l); l = 1, 2, ..., L;\ i = 1, 2, ..., M \tag{5}$$

$$\mathbf{u}_{net}(\mathbf{x}, \theta) = \mathbf{W}_{L+1} \cdot [\mathbf{H}_L^{(1)}, \mathbf{H}_L^{(2)}, ..., \mathbf{H}_L^{(M)}] + \mathbf{b}_{L+1} \tag{6}$$

Illustrative example: Function approximation

To illustrate the effect of the Fourier Feature embedding on the learnt solution's frequency; an illustrative example is presented here for a simple function approximation problem. The function $y(x)$ is a superposition of two sinusoidal signals with wave numbers $\kappa_1 = 1, \kappa_2 = 10$: $y(x) = sin(\kappa_1 x) + sin(\kappa_2 x)$; $x \in \mathbb{R}$. The

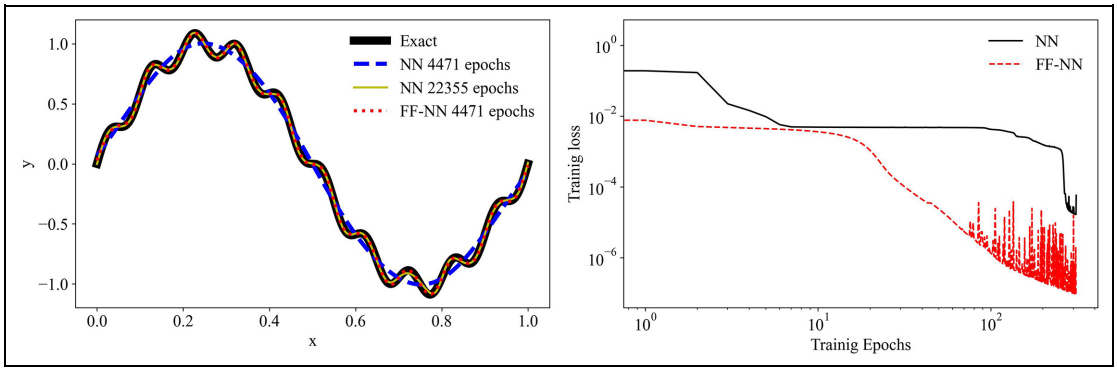

Figure 2. Left: Function approximation using Neural Network (NN) and Fourier Feature-Neural Network (FF-NN), both architectures trained on sparse points of a two frequency component signal. The results show that the FF-NN architecture converged faster to the exact function and learnt the low and high frequency components synchronously. Right: Relative absolute error for NN and FF-NN predictions.

exact form of $y(x)$ is plotted in the left panel in Figure 2. Two Neural Network architectures are tested to perform function approximation (prediction) based on training samples from the exact function $y(x)$. The first Neural Network is a conventional feed forward fully connected Neural Network (NN),[15] while the second Neural Network is a Multi Scale-Fourier Feature-Neural Network.[5] The FF-NN includes the Fourier Feature embedding layer as shown Figure 1 and equations (3)–(6) with spatial frequency (wave number κ) sampled from the normal distributions $N(0, \sigma_\kappa^{(1)} = 1)$ and $N(0, \sigma_\kappa^{(2)} = 10)$.

The NN and FF-NN have three hidden layers with 20 neuron each and *tanh* activation function.[16] The trainable parameters $\theta(\mathbf{W}, \mathbf{b})$ are initialized by the Glorot initializer[17] while ADAMS algorithm[18] with learning rate 10^{-3} is used as an optimizer.

For both architectures (NN and FF-NN) the networks are trained for31,000 iterations and the networks predictions are saved every 4471 iterations (one training cycle). The left panel of Figure 2 shows that the NN prediction after a single training cycle (4471 epochs) is just a pure sinusoidal wave with the lower wave number value $\kappa = 1$ due to the spectral bias, however the FF-NN architecture after the same number of epochs (4471) is able to accurately fit the function exact form. The right panel of Figure 2 shows the absolute relative error (training data loss) between the NN/FF-NN predictions and the training dataset. It is obvious that through the whole training period, the FF-NN architecture loss is lower than the NN's, and a better function approximation is achieved by the FF-NN compared to the NN even after large number of iterations.

To give more visual representation of the spectral bias phenomenon, the predictions of the NN and the FF-NN architectures after each training cycle are horizontally stacked together. A Wavelet transform is applied to each horizontally stacked series (NN and FF-NN separately) as shown in Figure 3. It can be seen that in the Wavelet transform of the NN prediction, only the low wave number $\kappa = 1$ is detected at the early training cycles until the 4^{th} cycle where the higher wave

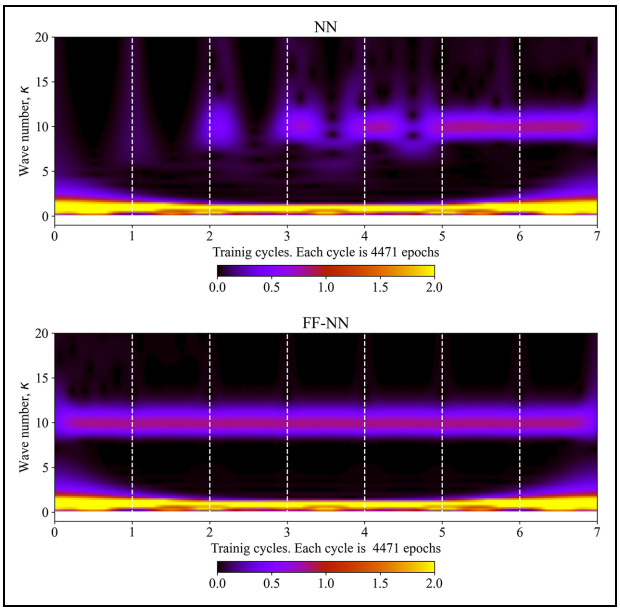

Figure 3. Wavelet transform evolution for the NN and the FF-NN signal approximation prediction during training cycles. The results show the NN architecture learnt only the low frequency solution component during the first 5 training cycles then learnt the high frequency component, the FF-NN architecture learnt the low and high frequency solution components synchronously during the first training cycle.

number $\kappa = 10$ appears. On the other hand, for the Wavelet transform of the FF-NN both wave numbers $\kappa = 1$ and $\kappa = 10$ appear after the first training cycle.

Although the presented illustrative example of function approximation is quite simple, it gives guidance on how the effect of using the Fourier Feature embedding in Neural Network can improve the prediction accuracy of real life physical problems that exhibits periodic behaviors.

Multi Scale-Spatio Temporal-Fourier Feature-Physics Informed Neural Network (MS-ST-FF-PINN) for incompressible fluid flow problems

For PDEs with solutions that exhibit periodic behavior spatially and temporally, the conventional structure of

the PINN is found to be inaccurate or computationally expensive due to the spectral bias problem.[9] To tackle the spectral bias of the PINN for spatio temporal PDEs[9] proposed the Multi Scale-Spatio Temporal-Fourier Feature-Physics Informed Neural Network (MS-ST-FF-PINN) in the feed-forward pass of the Neural Network. In MS-ST-FF-PINN the Fourier Feature embeddings (transformation) are applied to the Neural Network spatial and temporal inputs separately.

$\phi_E^{(x_i)}(x_i)$ and $\phi_E^{(t)}(t)$ in equations (7) and (8) are the input Fourier Features embeddings spatially and temporally respectively. Similar to the FF-NN mentioned earlier, the spatial and temporal frequencies $\mathbf{f}^{(x_i)}$, $\mathbf{f}^{(t)}$ are sampled from normal distributions $N(0, \sigma_{f_x}^{(i)})$ and $N(0, \sigma_{f_t}^{(i)})$ respectively. As shown in equations (9)–(12), each hidden layer has components for the spatial Fourier Feature embeddings $\mathbb{H} < .^{(x_i)}$ and a component for the temporal Fourier Feature embeddings $\mathbb{H} < .^{(t)}$ which increase the computational cost of the FF-PINN architecture compared to the PINN architecture. The final hidden layer $\mathbf{H_L}$ is computed in the Fourier Feature composition layer by a pointwise multiplication of its spatial and temporal components as shown in equation (13).

For the incompressible fluid flow, the input to the PINN and MS-ST-FF-PINN are the spatial Cartesian coordinates (x_1, x_2, x_3) and time t, while the outputs are the velocity in three components (u_1, u_2, u_3) and the pressure p. Other vectorial or scalar quantities can be added to the Networks output layer, such as temperature, salinity, phase fraction, free surface water elevation or any other fluid dynamic quantity related to the physical laws governing the fluid flow application.

The PINN and MS-ST-FF-PINN architectures for the incompressible fluid flow are summarized in the diagram in Figure 4. It is obvious that two new layers for the MS-ST-FF-PINN exist (Fourier Feature decomposition/ composition), but the loss functions and automatic differentiation are same for PINN and the MS-ST-FF-PINN.

$$\phi_E^{(x_i)}(x_i) = [sin(2\pi \mathbf{f}^{(x_i)} \times x_i); cos(2\pi \mathbf{f}^{(x_i)} \times x_i)]^T \tag{7}$$

$$\phi_E^{(t)}(t) = [sin(2\pi \mathbf{f}^{(t)} \times t); cos(2\pi \mathbf{f}^{(t)} \times t)]^T \tag{8}$$

$$\mathbf{H}_1^{(x_i)} = \sigma(\mathbf{W_1} \cdot \phi_E^{(x_i)}(x_i) + \mathbf{b_1}); i = 1, 2, ..., d \tag{9}$$

$$\mathbf{H}_1^{(t)} = \sigma(\mathbf{W_1} \cdot \phi_E^{(t)}(t) + \mathbf{b_1}) \tag{10}$$

$$\mathbf{H}_l^{(x_i)} = \sigma(\mathbf{W_l} \cdot \mathbf{H}_{l-1}^{(x_i)} + \mathbf{b_l}); \; l = 2, 3, .., L. \;\; i = 1, 2, ..., d \tag{11}$$

$$\mathbf{H}_l^{(t)} = \sigma(\mathbf{W_l} \cdot \mathbf{H}_{l-1}^{(t)} + \mathbf{b_l}) \quad l = 2, 3, .., L \tag{12}$$

$$\mathbf{H}_L = \prod_{i=1}^{d} \mathbf{H}_L^{(x_i)} \cdot \mathbf{H}_L^{(t)} \tag{13}$$

$$[u, v, w, p]_{net}^T = \mathbf{W}_{L+1} \cdot \mathbf{H}_L + \mathbf{b}_{L+1} \tag{14}$$

Results and discussion

Forward time independent example, double-lid-driven cavity problem

The first test case for comparison of the PINN and the FF-PINN architectures is a 2D double-lid-driven cavity problem at $Re = 10$; the double-lid-driven cavity has an upper and lower moving velocity boundaries while the side boundaries are nonslip walls as shown in the left panel in Figure 5.

Unlike the well known lid-driven cavity problem that has a single central vortex, the double-lid-driven cavity problem solution consists of two counter rotating upper and lower vortices as shown in the streamlines plot in Figure 5.

As a reference solution, the incompressible fluid flow equation set (15) and (16) are solved using NGSolve (a high performance finite element method python library).[19] The computational grid consists of 23×10^3 triangle elements generated by Netgen (Mesh generator). The velocity pressure coupling is implemented using the Taylor-Hood algorithm of 3^{rd} degree.[20] The time step is 0.01 s and the convergence tolerance is 10^{-7}; the problem converged quickly since a steady state solution exists for this small Reynolds number. The tangential velocity u_1 profiles at the top and bottom boundaries is parabolic ($u_{top} = u_{bottom} = Cx_1(1 - x_1)$) to avoid the singularity near the domain boundary corners, $C = 6$ to ensure average upper and bottom velocity is in agreement; the average velocity for the upper and bottom lids are ($\overline{u_{upper}} = \overline{u_{bottom}} = \int_0^1 6x_1(1 - x_1) = 1$). Figure 5 shows the finite element solution for the double-lid-driven cavity problem.

$$\frac{\partial u_j}{\partial x_j} = 0; x_j \in \mathbb{R}^2 \tag{15}$$

$$\frac{\partial(\rho u_i)}{\partial t} + \frac{\partial(\rho u_j u_i)}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \mu \frac{\partial^2 u_i}{\partial x_j \partial x_j} \tag{16}$$

The problem is solved in a forward scheme where the Neural Network has no information about the solution fields in the domain. Two architectures are tested, a PINN and an FF-PINN architectures. Since the problem has a steady state solution, the temporal acceleration term $\frac{\partial u_i}{\partial t}$ is eliminated form the momentum equation in the loss function. Hence the optimization functions become a time independent

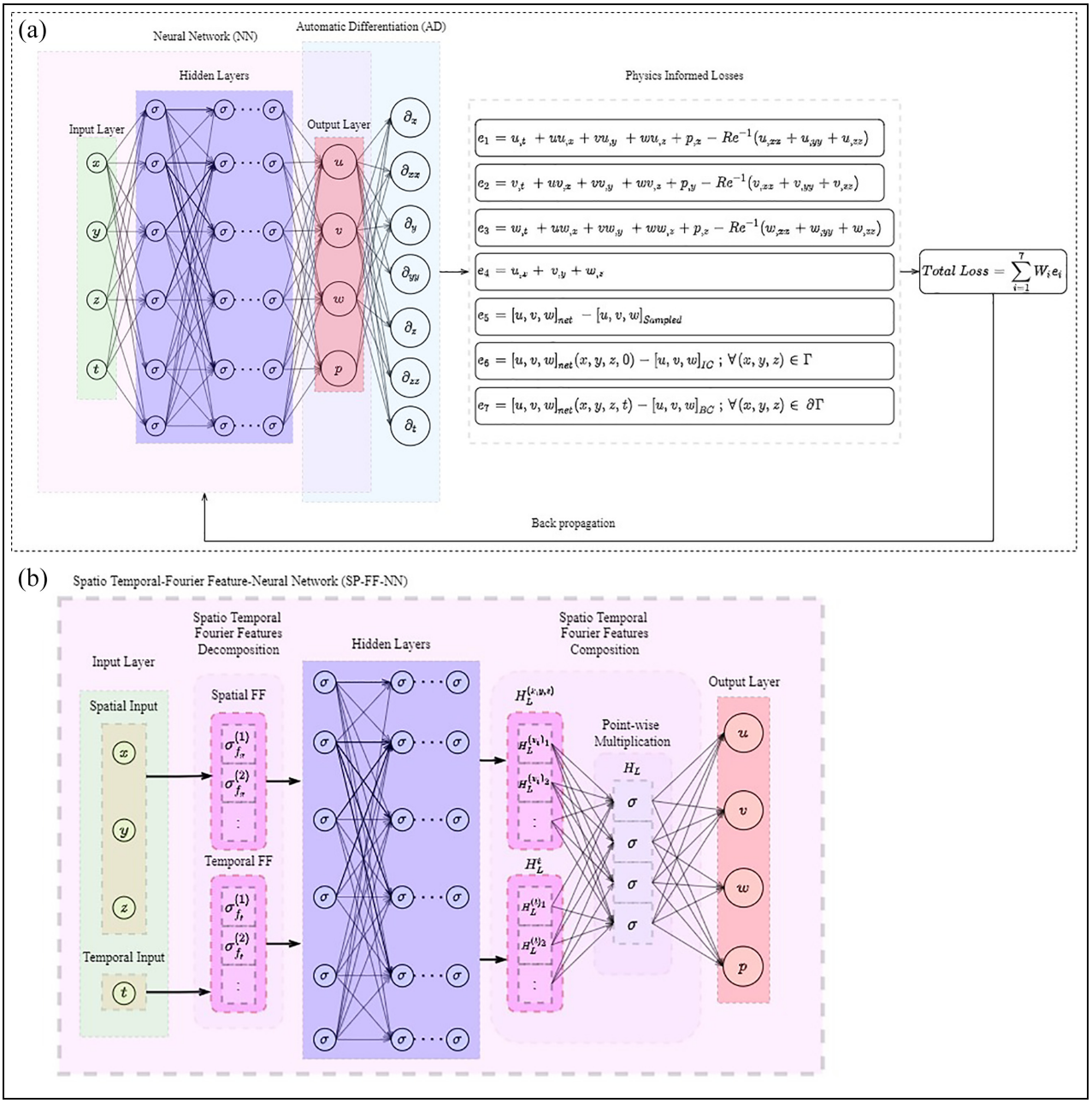

Figure 4. PINN and MS-ST-FF-PINN architectures for incompressible fluid flow: (a) Incompressible Flow PINN Architecture, including the feed forward Neural Network and (b) Incompressible Flow Spatio Temporal-Fourier Features PINN architecture. AD and Losses are same the same as PINN architecture shown above. The MS-ST-FF-PINN architecture has extra two layers (Spatio Temporal-Fourier Features Decomposition and the Spatio Temporal-Fourier Features composition layers). In MS-ST-FF-PINN, the spatio temporal $\mathbf{u}(\mathbf{x}, t)$ inputs are transformed to a series of sinusoidal functions with frequencies sampled from normal distribution. In the spatio temporal Fourier features composition layer, superposition is implemented to the decomposed signals to predict the NN outputs.

problem and the spatio temporal Fourier Features decomposition layer (Figure 4(b)) only include the spatial frequency standard deviations $\sigma_{f_x}^{(i)}$. For both architectures (PINN and FF-PINN) the Neural Networks consist of 2 inputs (x_1 and x_2), 3 outputs (u_1, u_2 and p), and 6 hidden layers with 50 neurons each with *tanh* activation function. In addition, the weights and the biases of the Network neuron are initialized by the Glorot initializer.[17]

ADAMS optimizer[18] with 10^{-3} learning rate is used to minimize loss functions (PDEs residuals and the boundary conditions) with $2{\times}10^4$ iterations. 1000 collocation points are uniformly randomly distributed in the domain, in addition to 1000 points for the domain boundaries.

Figure 6 shows the prediction of the PINN and the FF-PINN architectures for velocity u_1 as a function of training epochs. The figure shows that the FF-PINN

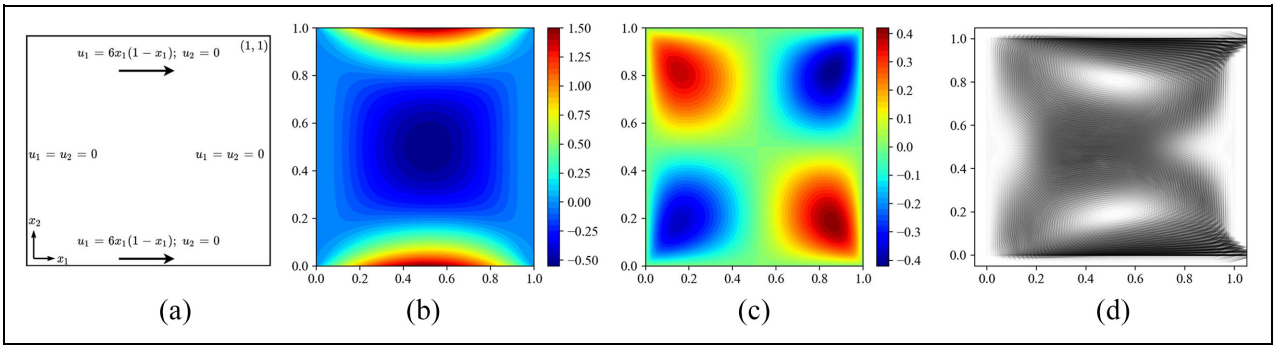

Figure 5. (a) Double-lid-driven cavity problem description at Re = 10.0, top and bottom boundaries have same velocity profile, vertical boundaries have no-slip condition, (b) Horizontal velocity u_1 FEM, (c) Vertical velocity u_2 FEM, and (d) Streamlines FEM.

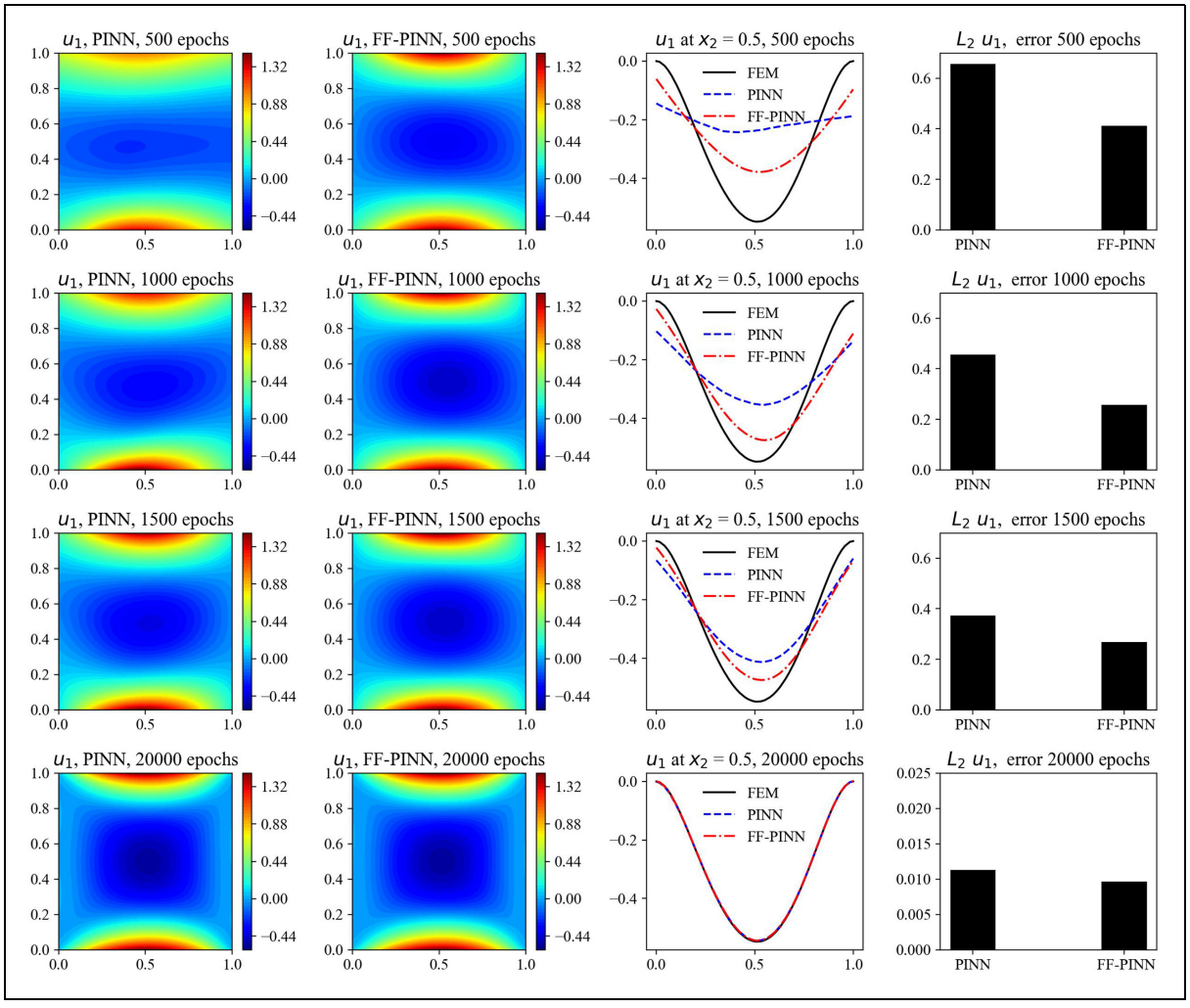

Figure 6. Horizontal velocity u_1 prediction for PINN and FF-PINN for double-lid-driven cavity problem. The reference FEM solution is shown in Figure 5. u_1 velocity profile at x_2 shows that FF-PINN architecture is able to converge faster than the PINN to the FEM solution. The right panel shows the second norm relative error for PINN and FF-PINN architectures with respect to the FEM solution.

architecture after 500 training epochs predicted the central circular shape for velocity u_1, while the PINN architecture after 500 and 1000 epochs predicted the central shape more flattened axially in x_1 direction; this axial flattening in early training epochs shows that the PINN architecture tried to converge to a single frequency sinusoidal velocity profile for all vertical sections of the square domain, see top left contour plot in Figure 6. The third panel shows the evolution of the u_1 velocity profile at $x_2 = 0.5$ during the training cycles,

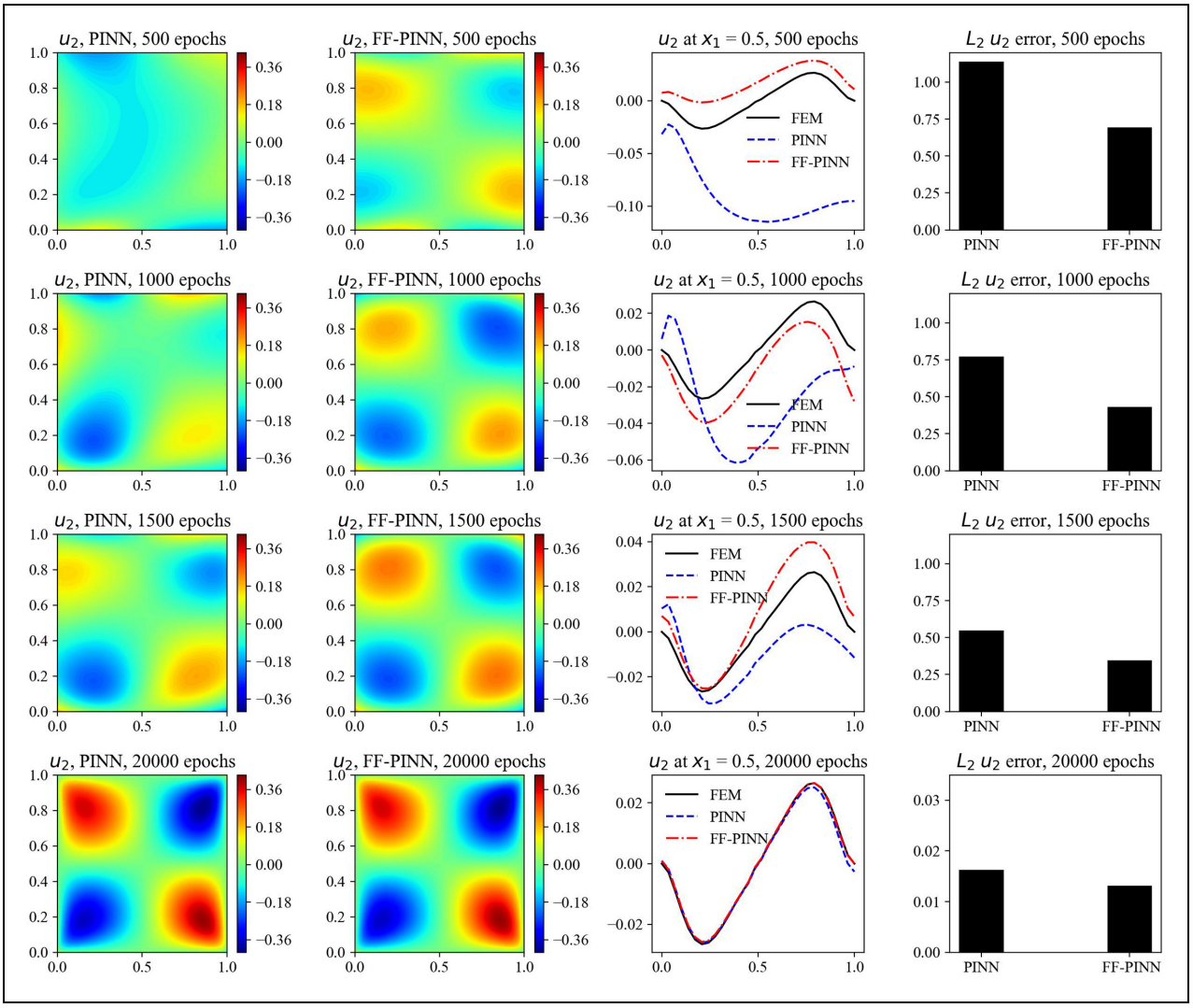

Figure 7. Vertical velocity u_2 prediction for PINN and FF-PINN for double-lid-driven cavity problem. The reference FEM solution is shown in Figure 5. FF-PINN architecture is able to converge faster than the PINN to the correct solution. The PINN architecture struggles to learn the high frequency solution at early training epochs due to the spectral bias. The right panel shows the second norm relative error for PINN and FF-PINN architectures with respect to the FEM solution.

the FF-PINN predicted the bell shape velocity profile after only 500 epochs while the PINN architecture prediction profile is almost flat (low frequency) due to spectral bias. The right panel of Figure 6 shows the evolution of the error second order norm for the velocity u_1 with respect to the FEM solution, during all training cycles the L_2 u_1 error norm for the FF-PINN architecture compared to the FEM solution is lower than the PINN architecture.

Figure 7 shows the prediction of the PINN and the FF-PINN architectures for the velocity u_2 as a function of epochs. The figure shows that the FF-PINN architecture after 500 training epochs predicted the shape for the four corners vertical velocity u_2, while the PINN architecture predicted the vertical velocity u_2 shape flattened laterally in x_2 direction. The PINN architecture is not able to predict the four corners shape until it reaches 1500 epochs. The third panel shows the u_2 velocity component predictions for PINN and FF-PINN at $x_1 = 0.5$; after 500 epochs the FF-PINN architecture was able to predict the full sinusoidal velocity profile, while the PINN architecture predicted half sinusoidal velocity profile due to spectral bias. The right panel of Figure 7 shows the evolution of the error second order norm for the velocity u_2 with respect to the FEM solution, during all training cycles the L_2 u_2 error norm for the FF-PINN architecture is lower than the PINN architecture

Figure 8 shows the mean absolute relative error for the planar velocities u_1 and u_2 during the training cycles shaded with the standard deviation of the error in the domain. It is obvious that for the early training cycles the relative error for the FF-PINN architecture is lower than the PINN architecture, then at later training cycles, the relative error converges for later training cycles but still the FF-PINN architecture has lower

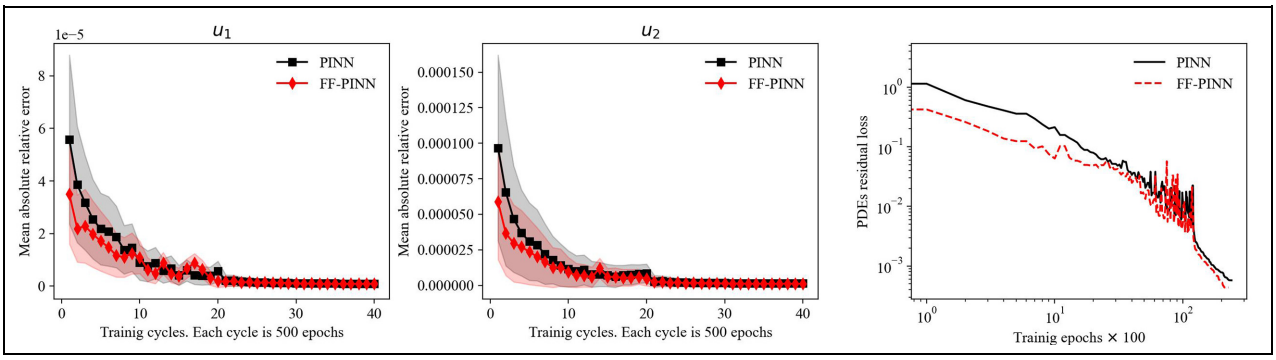

Figure 8. Mean absolute relative error per training cycle for horizontal velocity u_1 (left) and the vertical velocity u_2 (middle) for PINN and FF-PINN architectures, the shaded area is the error standard deviation from the mean error. PDEs (continuity and momentum equations) residuals per training epochs (right). The FF-PINN architecture is faster than PINN architecture to converge toward the correct solution and satisfy the governing PDEs.

relative error values. The right panel of Figure 8 shows the residual loss for the PDEs (continuity equation (15) and momentum equation (16)). It shows that the PDEs residual loss is lower for the FF-PINN architecture through all training epochs, and the slope of the FF-PINN is steeper at the later training epochs, meaning that with more training epochs the FF-PINN architecture will continue minimizing the governing equations residuals.

Inverse time independent example, Kelvin wave pattern

The second example is an inverse problem to infer the water free surface elevation η and the vertical free surface velocity u_3 from the horizontal free surface velocities u_1 and u_2. This is a novel approach contributing to remote sensing techniques for wave elevation measurement. This allows the water wave surface elevation to be determined using a single calibrated camera, instead of a stereo vision[21] or radar systems.[22] The model consists of a modified version of the momentum equation and the Kinematic Free Surface boundary conditions (KFS) as shown in equations 17 and 18 respectively. The derivation of equation (17) is presented in Salmon[23] where the problem dimension is reduced from 3D to 2D. The PINN total loss functions consists of the residuals of momentum equation (17) and the kinematic free surface PDE 18, in addition to satisfying the sampled horizontal velocities at the free surface u_1 and u_2.

$$\frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j} = g_3 \frac{\partial \eta}{\partial x_i}; i \;\; j \in \{1, 2\} \tag{17}$$

$$u_3 = \frac{\partial \eta}{\partial t} + u_j \frac{\partial \eta}{\partial x_j}; j \in \{1, 2\} \tag{18}$$

The model is tested on a numerically generated free surface Kelvin wave pattern down stream of a 1/31.6 scaled KRISO Container Ship (KCS)[24,25] as shown in Figure 9, the KCS model parameters are shown in Table 1. The free surface numerical solution of the KCS model is implemented using the Interfoam library[26] available in OpenFOAM,[27] the CFD simulation is done for half of the domain (port side). The free surface is modeled using single fluid mixture assumption Volume of Fluid method that solves for the volume phase fraction α transport equation (21) of the primary fluid represented by the water, alongside with the continuity and the momentum equations shown in equations (19) and (20). InterFoam library uses the PIMPLE algorithm for velocity pressure coupling.[28] The mixture fluid properties such as density ρ and dynamic viscosity μ are computed based on the mixture phase fraction as shown in equations (22) and (23), where the subscript a or w represents air or water respectively. $f_{\sigma i}$ represents the surface tension force.

Table 1. KCS model parameters.

Scale ratio	1/31.6
Speed [m/sec]	2.1964
Length Over All (LOA) [m]	7.72
Length Water Line (LWL) [m]	7.28
Breadth [m]	1.0190
Depth [m]	0.6013
Draft [m]	0.3418
Fr	0.26

$$\frac{\partial u_j}{\partial x_j} = 0 \tag{19}$$

$$\frac{\partial (\rho u_i)}{\partial t} + \frac{\partial}{\partial x_j}(\rho u_i u_j) = -\frac{\partial p}{\partial x_i} + \mu \frac{\partial}{\partial x_j}\left(\frac{\partial u_i}{\partial x_j}\right) + \rho g_i + f_{\sigma i} \tag{20}$$

$$\frac{\partial \alpha}{\partial t} + \frac{\partial (\alpha u_j)}{\partial x_j} = 0 \tag{21}$$

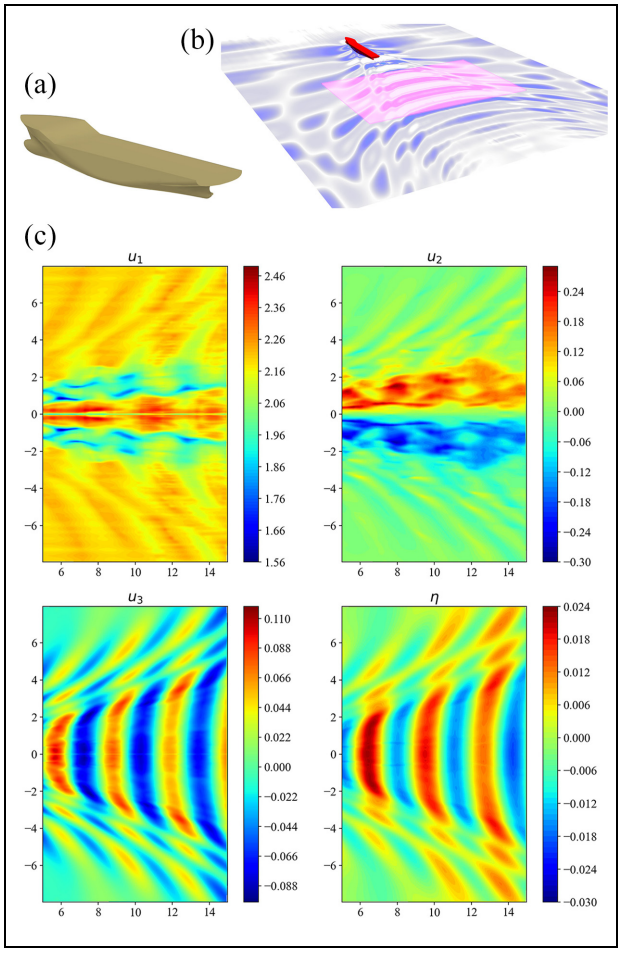

Figure 9. OpenFOAM solution for the free surface two phase Volume of Fluid simulation of a KCS model at $F_n = 0.26$: (a) KCS model, (b) Kelvin wave pattern, and (c) OpenFOAMCFD solution for the shaded area.

$$\rho = \alpha\rho_w + (1-\alpha)\rho_a \tag{22}$$

$$\mu = \alpha\mu_w + (1-\alpha)\mu_a \tag{23}$$

The computational domain is generated by blockMesh utility in OpenFOAM to construct the background mesh with dimension ($39 \times 15 \times 6\ m^3$) and resolution $120 \times 60 \times 40$ cells, then the Snappyhexmesh utility[29] is used to construct the mesh around the KCS model and apply refinement at the free surface region resulting in 9.5×10^6 total number of finite volume cells. In Snappyhexmesh, the *castellatedMesh* and *snap* steps were activated with 20 *resolveFeatureAngle* to extract the KCS geometry and 90 *maxConcave* angle to achieve smooth cell transition, while the *addlayer* step is ignored as this study did not focus on the estimation of the shear stresses and forces applied on the KCS body. The *snap* step is implemented with five smoothing iterations.

The inlet flow velocity is 2.1964 m/s based on the simulated Froude's number $F_r = \frac{U_\infty}{\sqrt{gL}} = 0.26$, where L is the KCS length at water line (LWL) = 7.28 m. The water depth is 3 m, and the problem is simulated for 20 s with initial time step of 0.001 s that is automatically adjustable to satisfy the maximum Courant number = 1.

To test the model in both architectures PINN and FF-PINN, a single snapshot of the free surface is captured at $t = 20$. The free surface is extracted by importing the nodal values at the phase fraction α contour isosurface at 0.5 value that represents the intersection between the water and the air in the domain. Figure 9 shows the free surface and the Kelvin wave pattern generated by the KCS, the pink shaded area represents the study domain for the PINN and FF-PINN. The study subdomain starts about 5 m downstream of the KCS model, far from the boundary layer region. Figure 9(c) shows the contours for the planar velocities u_1 and u_2, vertical velocity u_3 and the free surface elevation η in the shaded area.

The model is tested for three architectures PINN, FF-PINN-a with $\sigma_{f_x} = 1.0, 10.0$ and FF-PINN-b with $\sigma_{f_x} = 0.1, 1.0$, the model total loss function's objective is to satisfy the sampled horizontal velocities u_1 and u_2 and minimize the residuals of equations (17) and (18). Since only a single snapshot is studied for this problem, all temporal derivative terms $\frac{\partial(\cdot)}{\partial t}$ are ignored.

For PINN, FF-PINN-a and FF-PINN-b the NN inputs are the horizontal coordinates x_1 and x_2, while the NN outputs are the velocity components (u_1, u_2, u_3) and the free surface elevation η. The hidden layers consist of six layers with 50 neurons each with *tanh* activation function, in addition, the weights and the biases of the Network neuron are initialized by the Glorot initializer.[17] ADAMS optimizer is used with learning rate 10^{-3} and 4×10^4 total number of iterations. 16×10^3 sampled points for the horizontal velocities u_1 and u_2 training, while 2×10^4 uniformly sampled points are used as collocation points to satisfy the PDEs 17 and 18.

Figure 10 shows the predictions' evolution of the axial velocity u_1 for the PINN, FF-PINN-a and FF-PINN-b architectures during the training cycles. The left panel in Figure 10 shows that PINN architecture only captures the low frequency component through the whole training evolution. The second panel column shows the FF-PINN-a architecture with $\sigma_{f_x} = 1.0, 10.0$, where at 2000 training epochs the predicted velocity u_1 is only the high frequency (noisy) velocity components, as the training advances, the FF-PINN architecture is able to capture u_1 velocity behavior including the high and low frequency components.

The third panel shows the prediction for the FF-PINN-b architecture with $\sigma_{f_x} = 0.1, 1.0$. The FF-PINN-b is able to capture the low and high frequency components in the first training cycle at 2000 epochs.

At later training epochs, both FF-PINN-a and FF-PINN-b architectures captured the velocity u_1 behavior with high and low frequency components, while the PINN architecture has struggles to capture the high frequency component.

The right panel in Figure 10 shows the second order error norm of the three architectures. FF-PINN-a has the highest accuracy during the training

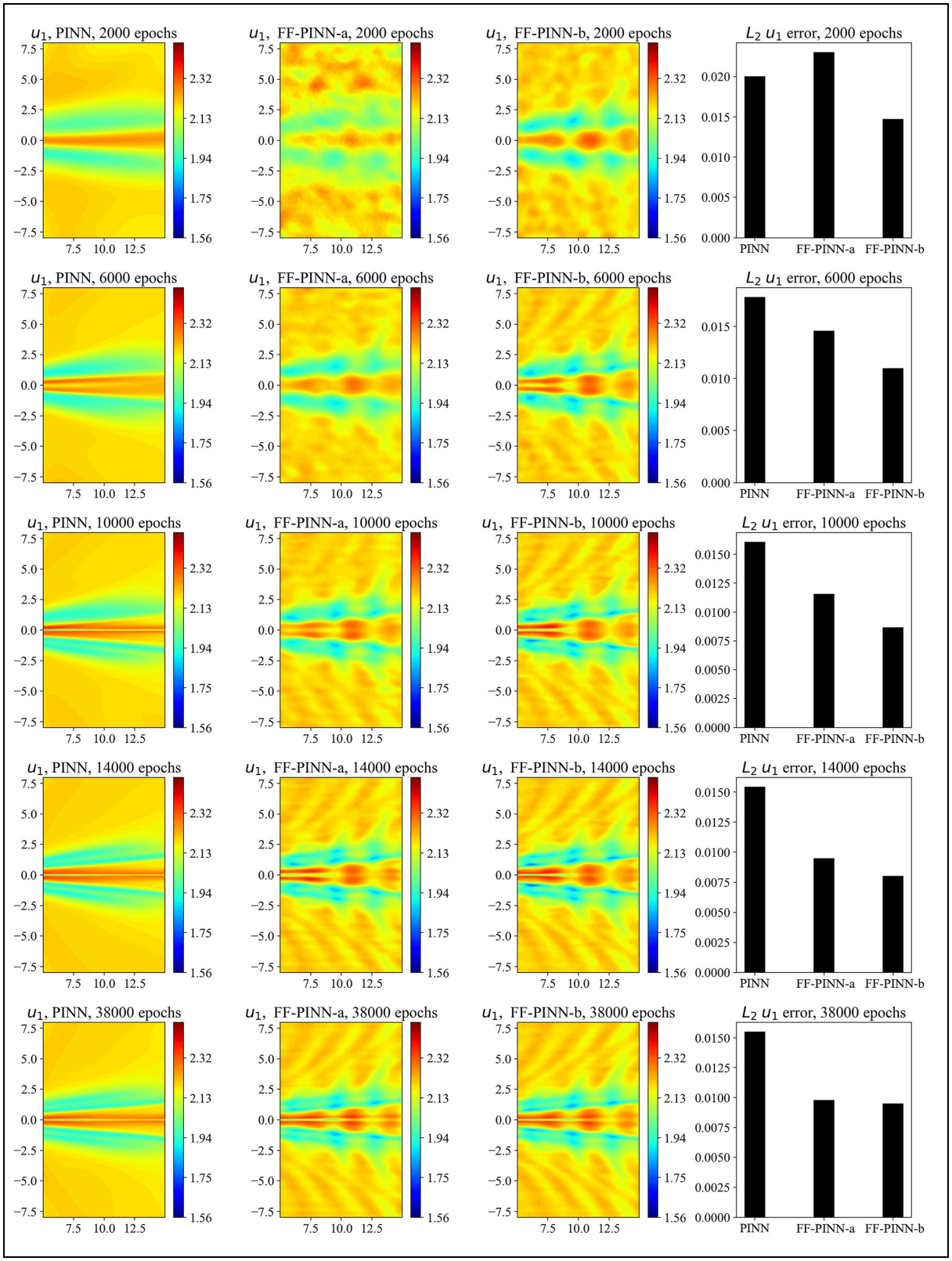

Figure 10. Axial velocity u_1 prediction for PINN, FF-PINN-a and FF-PINN-b. The reference Finite Volume solution is shown in Figure 9 for the KCS model at $F_n = 0.26$.

followed by the FF-PINN-b that behaves better than PINN architecture except in early training epochs as it converged faster to the high frequency components solution.

The same trend is shown for the lateral velocity u_2, the vertical velocity u_3 and free surface elevation η. Figures 11 to 13 show the prediction of the three architectures for u_2, u_3 and η. In these three figures, the

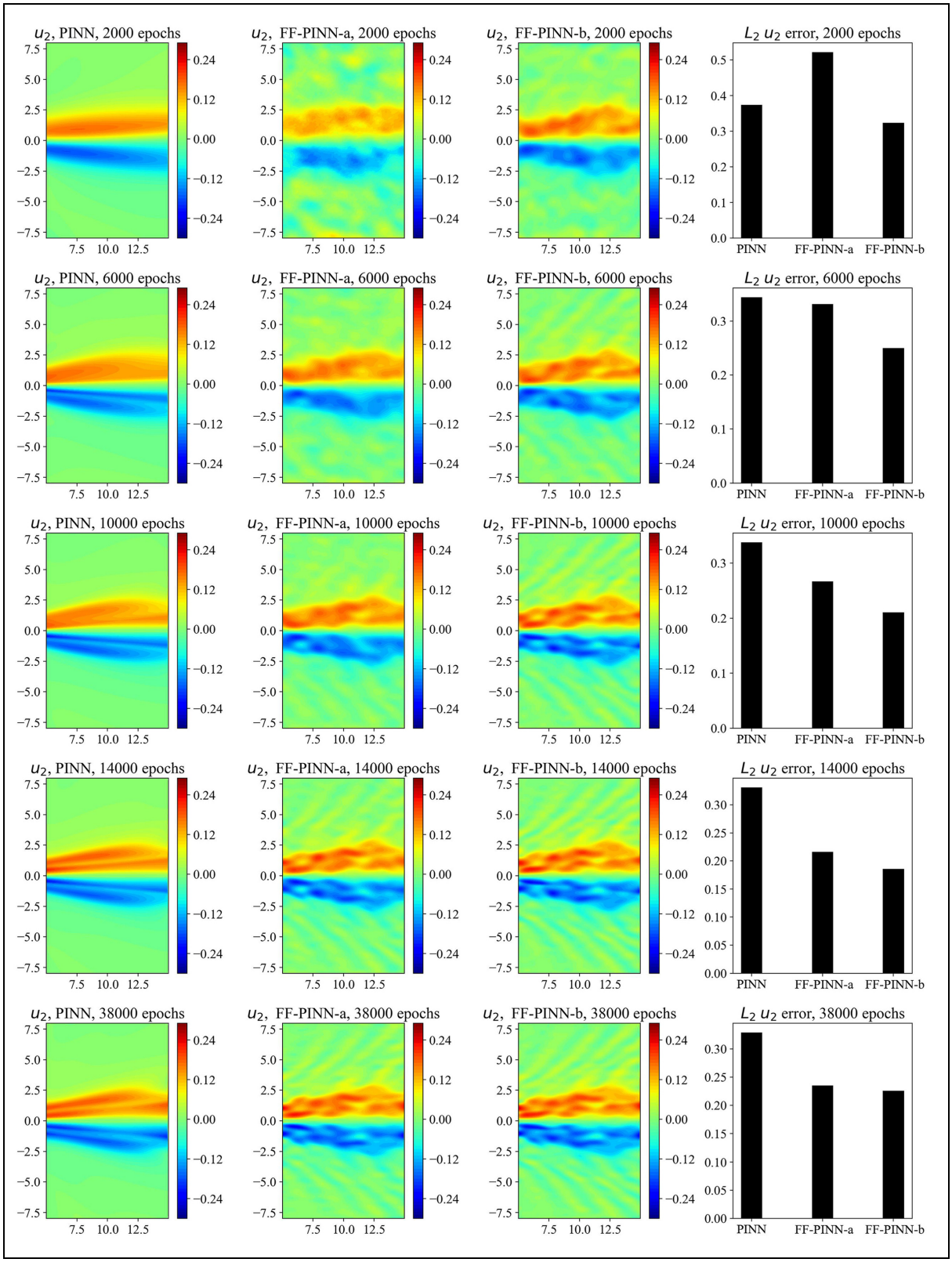

Figure 11. Lateral velocity u_2 prediction for PINN, FF-PINN-a and FF-PINN-b. The reference finite volume solution is shown in Figure 9 for the KCS model at $F_n = 0.26$.

PINN architecture only converges to the low frequency components. The FF-PINN-a architecture with $\sigma_{f_x} = 1.0, 10.0$ learn only the high frequency components in the early training epochs then converges to both high/low frequency components in later training epochs. While FF-PINN-b with $\sigma_{f_x} = 0.1, 1.0$ can synchronously learn high and low frequency component solutions and achieve the

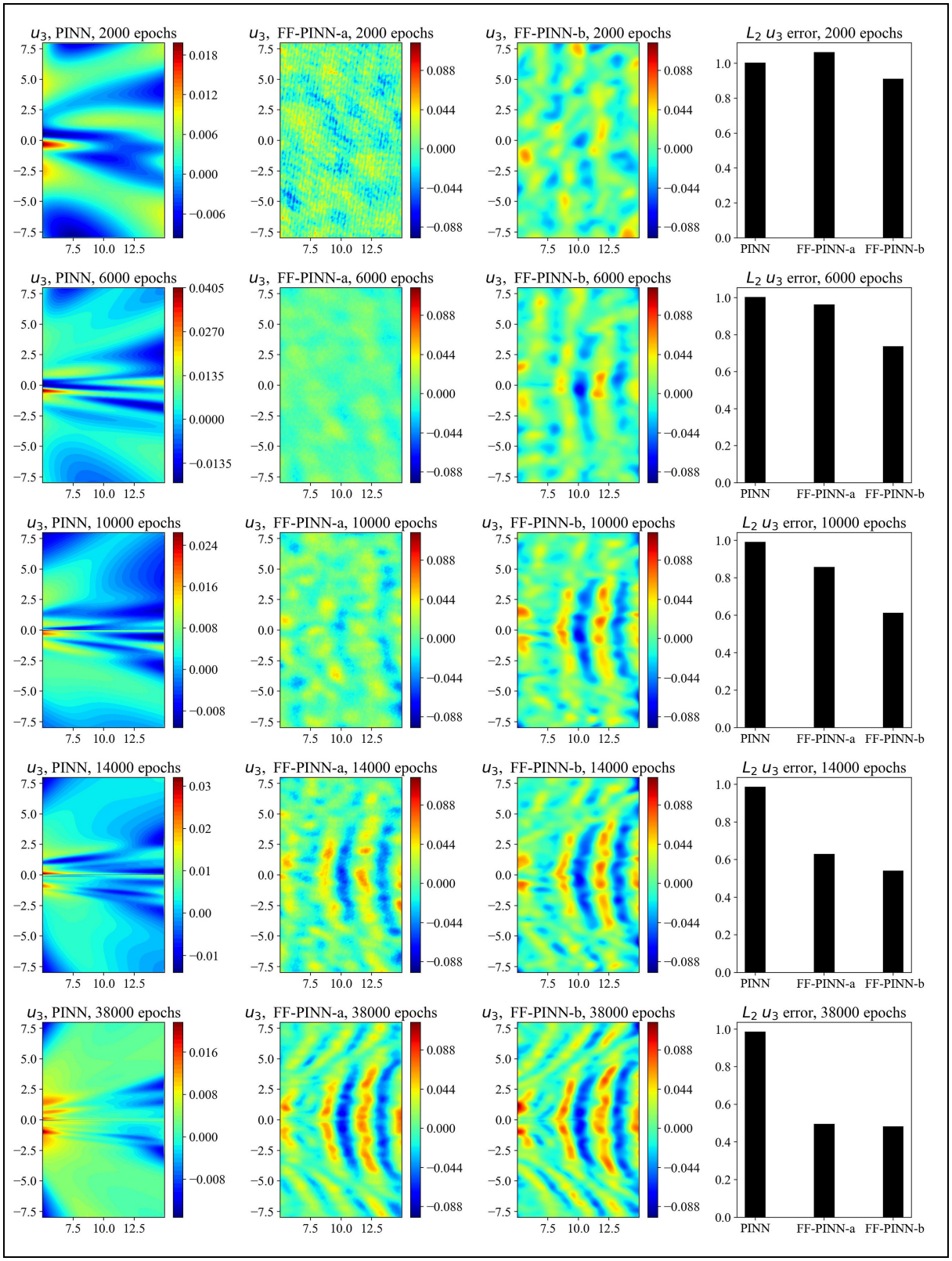

Figure 12. Vertical velocity u_3 prediction for PINN, FF-PINN-a and FF-PINN-b. The reference Finite Volume solution is shown in Figure 9 for the KCS model at $F_n = 0.26$.

highest accuracy for learning the training data (u_1,u_2) and the inferred data (u_3,η).

The top panel of Figure 14 shows the PDEs residual loss for the three architectures and the horizontal velocity training data (u_1, u_2). For the PINN case the PDEs residuals loss is the lowest for the early training epochs, however the training data training loss for PINN is high at early training epochs, this shows that the PINN architecture converged faster to a solution that satisfies the set of PDEs but did not have the ability to

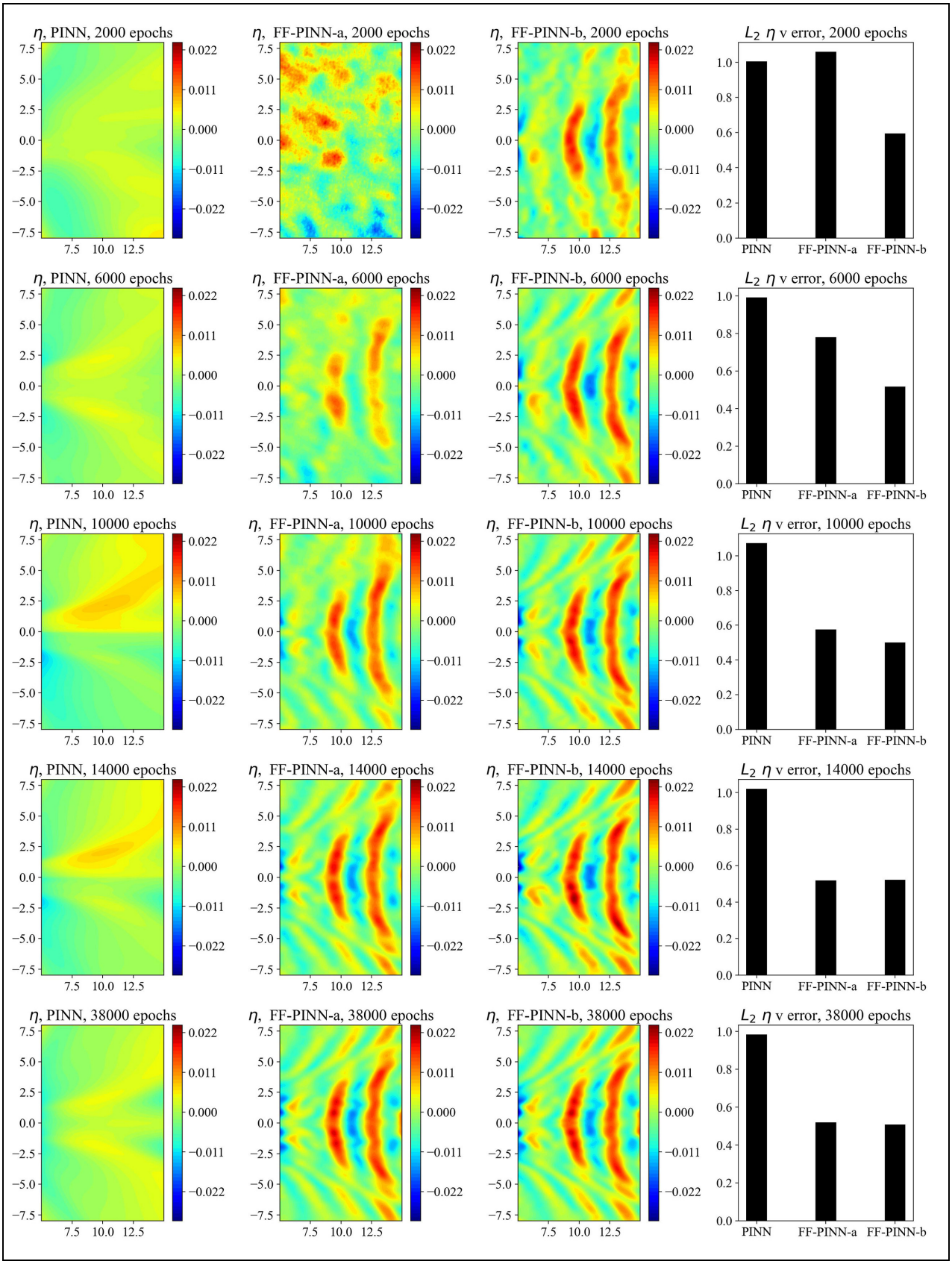

Figure 13. Water free surface elevation η prediction for PINN, FF-PINN-a and FF-PINN-b. The reference Finite Volume solution is shown in Figure 2 for the KCS model at $F_n = 0.26$.

accurately model the problem due to spectral bias. Convergence to a solution satisfies the PDEs does not mean convergence to the correct solution. This one of the drawbacks of PINN where no unique solution

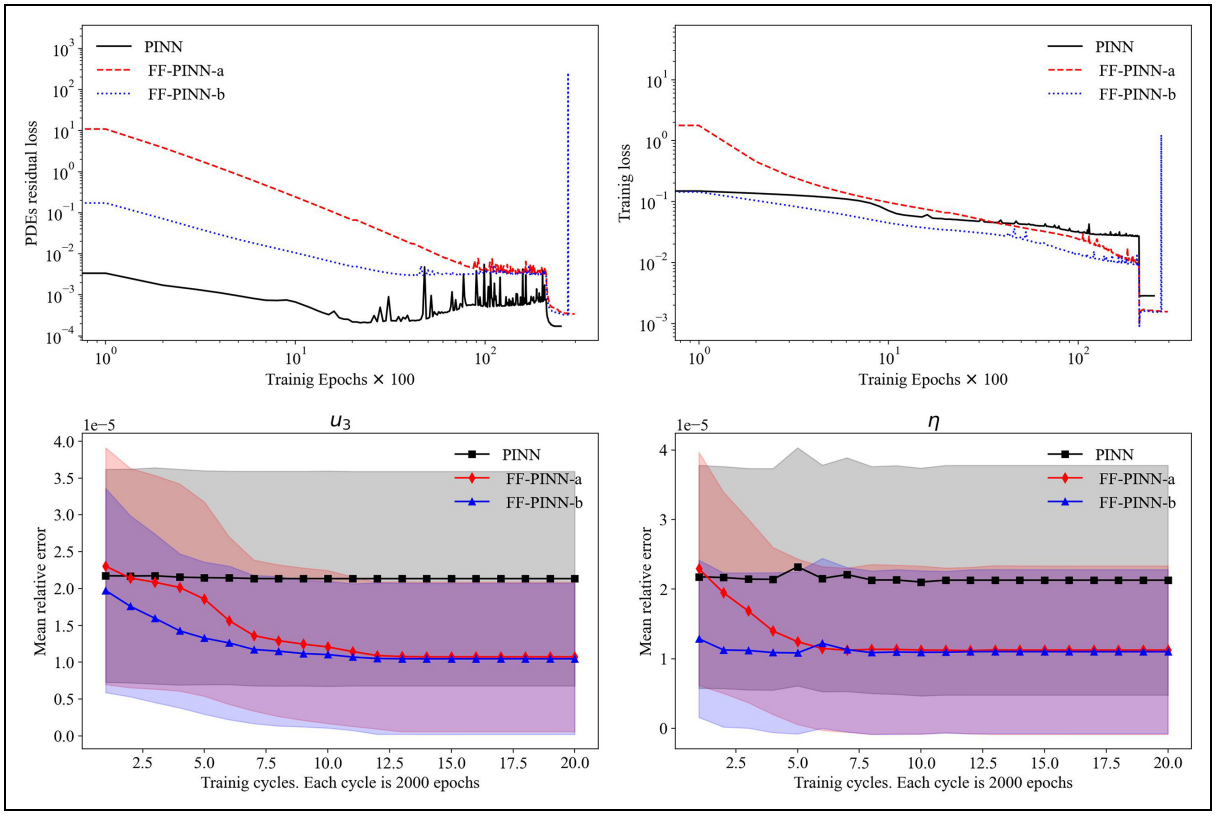

Figure 14. PDEs residuals (top left), u_1 and u_2 velocities training loss (top right), mean absolute relative error for the vertical velocity u_3 and surface elevation η (bottom), the shaded area is the error standard deviation from the mean error.

exists.[30] The Figure also shows that The FF-PINN-b architecture PDEs residual loss is lower than FF-PINN-a during all training epochs. Furthermore, the FF-PINN-b architecture has lowest training loss for (u_1, u_2) during the whole training iterations due to the ability to learn high/low frequency components synchronously during the training.

The bottom two panels in Figure 14 show the Mean relative error for the inferred variables u_3 and η shaded with the error standard deviation in the domain. For both u_3 and η the FF-PINN architectures have lower error compared to the PINN architectures throughout the training iterations, and FF-PINN-b showed the fastest convergence.

Inverse time dependent example, Von Karman Vortex shedding interaction downstream a cylinder array

The third test case is a 2D inverse time dependent incompressible flow for Von Karman vortex shedding interaction downstream of an array of wind turbines at 2.5×10^5 Re_D, where D is the cylinder diameter, $D = 0.5$ m. The axial and lateral distance between the cylinders are 6 m and the kinematic viscosity is 10^{-5}. The reference solution is obtained by Large Eddy Simulation (LES)[31] using OpenFOAM and simulated for 15 s with uniform steady inlet flow velocity $U_\infty = 5\ m/sec$. PISO algorithm is used for velocity pressure coupling.[28] The grid is generated by Snappyhexmesh utility[29] with49,333 grid points. Figure 15 shows the OpenFOAM solution at $t = 15$, the top panel shows the contours of turbulent kinetic energy, while the bottom panels show the solution of the flow fields (u_1,u_2,p) for the shaded area.

This test case objective is to infer high fidelity/resolution solution using PINN/FF-PINN architectures from a low fidelity/resolution learning case. The low fidelity/resolution solution is obtained from numerical solution implemented on a very coarse grid. The PINN and the FF-PINN architectures are trained on the flow fields (u_1,u_2,p) in the shaded pink rectangle domain shown in Figure 15(a). The OpenFOAM computational grid in the shaded has19,000 points, while the PINN/FF-PINN architectures are trained on only 200 points per frame, the sparse training points locations are shown by randomly distributed black points in Figure 15(b).

For both PINN and the FF-PINN architectures, the Neural Networks has 3 inputs (x_1,x_2,t), 3 outputs (u_1,u_2,p), and 5 hidden layers with 50 neurons each with *tanh* activation function, in addition, the weights and the biases of the Network neurons are initialized by the Glorot initializer.[17] ADAMS optimizer is used with learning rate 10^{-3} for 15×10^3 iterations followed by L-BFGS optimizer[32] for another 15×10^3 iterations. Due to the limited GPU memory, the training temporal domain is only 2 s $t \in [13, 15]$ and the OpenFOAM training dataset are sampled every 0.1 s. The FF-PINN

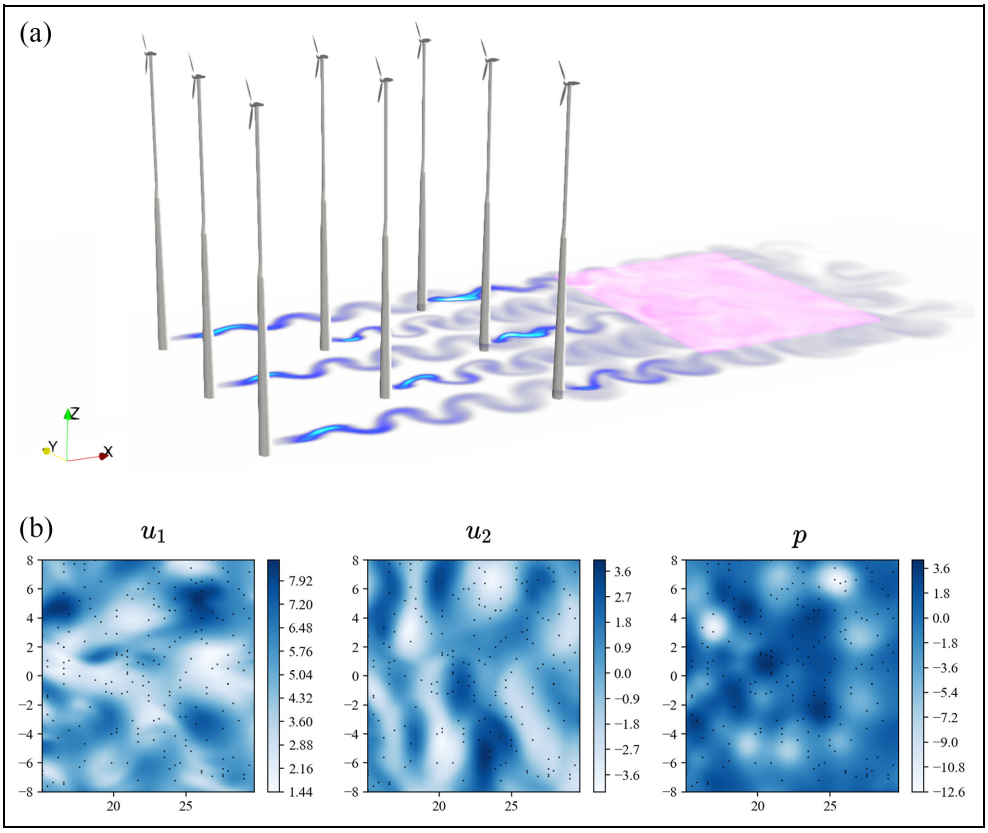

Figure 15. Large Eddy Simulation for 2D flow over cylinder array at Re= 2.5×10^5. The pink shaded area is the training domain for the PINN and the FF-PINN architectures, black dots are the sparse training points. (a) Turbulent kinetic energy, OpenFOAM solution for flow over cylinder array at Re= 2.5×10^5. (b) Velocity and pressure fields at the pink shaded area downstream.

architecture has two spatially and temporally scales with standard deviations $\sigma_{f_x} = \sigma_{f_t} = 1.0, 10.0$.

The prediction of the PINN and the FF-PINN architectures for the planar velocity fields (u_1,u_2) and the pressure fields p are shown in Figures 16 to 18 respectively at $t = 14$ s. Similar to the previous test cases, the FF-PINN architecture learn the low/high frequency solution components synchronously and quickly compared to the PINN architecture that was not able to learn the high frequency component of the solution at early training epochs. After a large number of training iterations, the PINN architecture was able to converge to the low/high frequency solution components, nevertheless the FF-PINN still has better accuracy.

Figure 19 shows the training mean relative error shaded with the error standard deviation for the PINN and the FF-PINN architectures at $t = 14$ s, for all training cycles the mean relative error and the error standard deviation for the PINN architecture is double that of the FF-PINN. Similarly, the FF-PINN has lower PDEs residuals loss at all spatio temporal residual points and lower loss for all spatio temporal training data as shown in Figure 20.

Figure 21 shows the temporal prediction for the OpenFOAM, PINN and the FF-PINN for the flow fields. Both PINN and FF-PINN were not able to predict the high temporal frequency component in the two planar velocity fields (u_1,u_2), but in general the FF-PINN architecture has better fit with the OpenFOAM solution. The pressure field prediction in the right panel in Figure 21 shows that the FF-PINN prediction suffer from over-fitting problem, since the prediction aligns with the OpenFOAM solution only at the time stamps fed to the architecture during training. PINN architecture showed smoother behavior for the pressure field, however, it is not accurate.

Summary and conclusion

In this paper, we implemented the Fourier Features-Physics Informed Neural Network (FF-PINN) on incompressible fluid flow problems to study the effect of the Fourier Features embeddings on the spectral bias phenomenon. Spectral bias is the tendency of Neural Networks to converge to low frequency solutions' faster than to high frequency solutions. In some cases this means convergence to high frequency solutions is not achieved even with high number of iteration.

This study included three test cases for incompressible fluid flow problems where comparative study is implemented between the conventional PINN and the FF-PINN architectures. The first test case is a 2D forward time independent double-lid-driven cavity

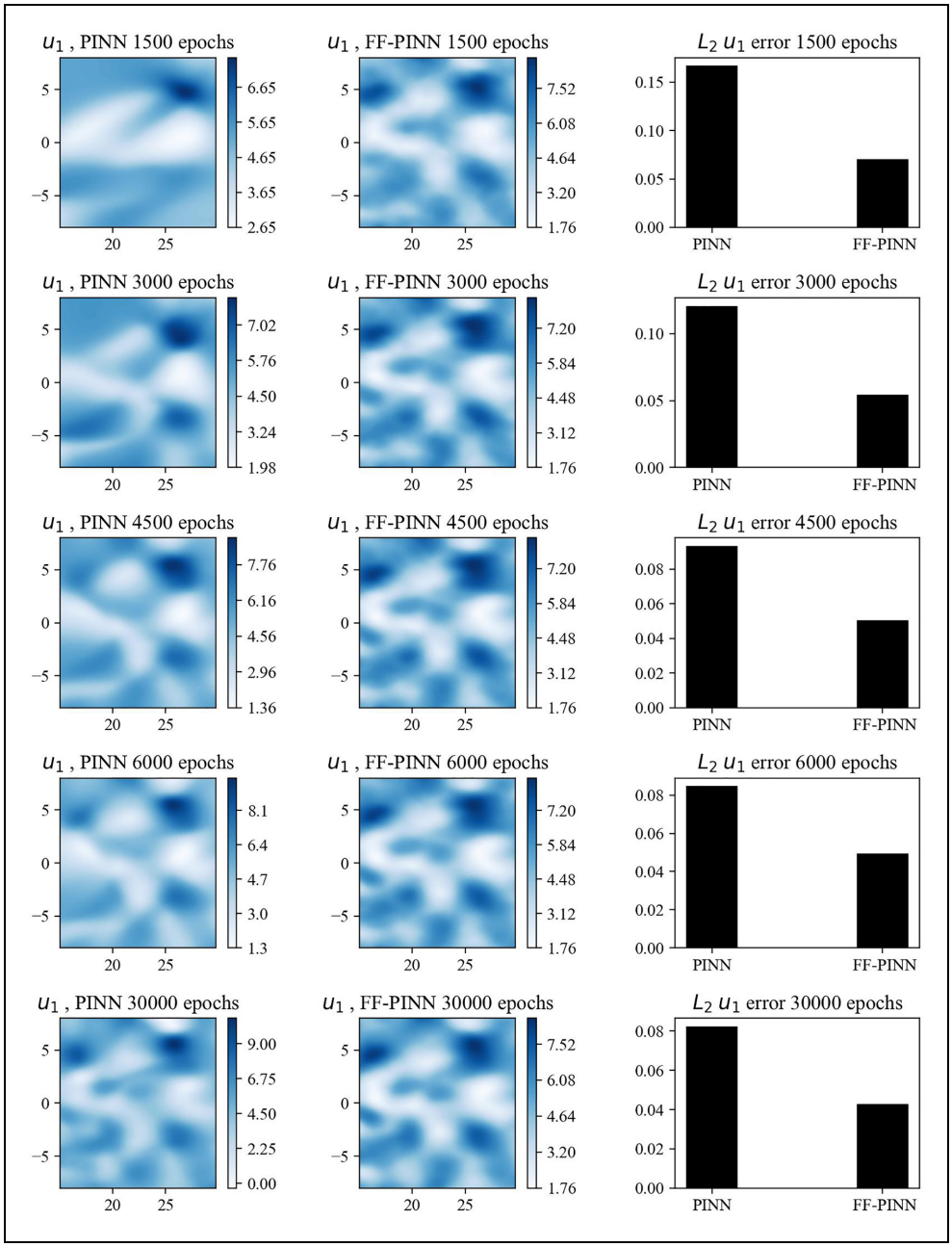

Figure 16. Axial velocity component u_1 prediction by PINN and FF-PINN architectures at $t = 14$ s. The reference Finite Volume solution is shown in Figure 15. The FF-PINN is able to learn the high frequency signal at early training epochs, however, the PINN architecture converges to the low frequency solution due to spectral bias.

problem at 10 Re. The reference solution is obtained by high order FEM solution of NGSolve library. The FF-PINN showed a faster prediction for planar spatially periodic velocity fields (u_1,u_2) at early training iterations compared to the PINN architecture, also FF-PINN showed higher accuracy at the end of the training iterations compared to the PINN architecture.

The second test case is a 2D inverse time independent free surface problem for numerically generated Kelvin pattern waves downstream of an KCS model at $0.26Fn$. The objective of the problem is to infer the vertical velocity component u_3 and the free surface elevation η from the sampled horizontal velocity fields (u_1,u_2). The results showed a dependency of the FF-PINN on the chosen standard deviation σ_x where the spatial frequency f_x is sampled from in the Fourier Features embedding layer. For FF-PINN with higher σ_x the Neural Network learnt the noisy very high frequency component in the velocity fields at early training iterations, then converged toward the correct solution. On the other hand, for PINN with no Fourier Features embeddings the Neural Network was unable to learn the high frequency solution components and only converged to the low frequency solution components. With intermediate values for the spatial standard deviation σ_x, the FF-PINN architecture was able to learn the

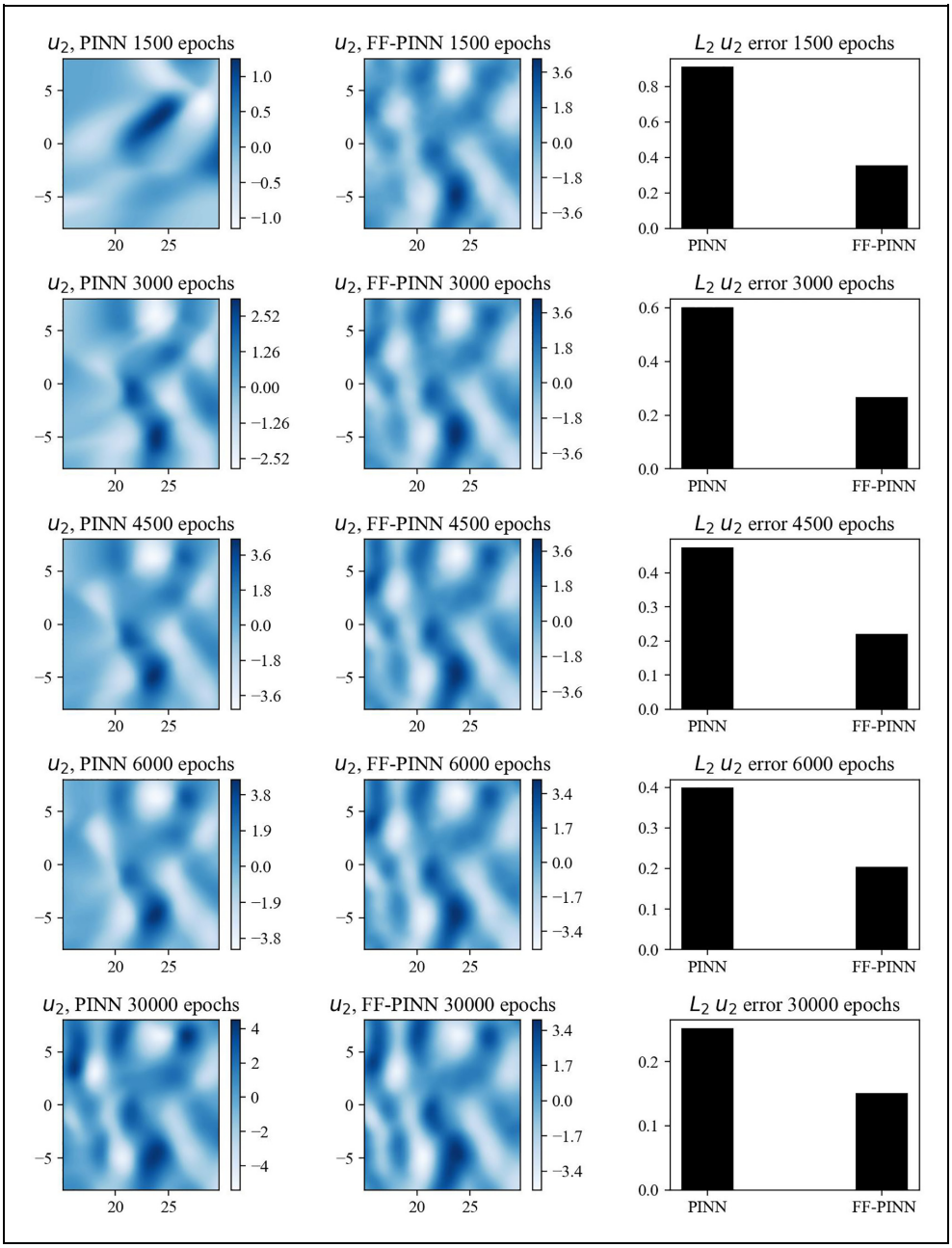

Figure 17. Lateral velocity component u_2 prediction by PINN and FF-PINN architecture at $t = 14$ s. The reference Finite Volume solution is shown in Figure 15. The FF-PINN is able to learn the high frequency signal at early training epochs. The PINN architecture struggles to learn the high frequency solution components due to spectral bias.

low/high frequency components synchronously, which implies that the choice of the Fourier Features distribution is a problem dependent.

The third test case is an inverse 2D time dependent problem for turbulent Von Karman vortex shedding interaction downstream of cylinder array at 2.5×10^5 Re. The PINN and the FF-PINN architectures were trained on (low resolution) sparse data for the velocity and pressure fields to infer high fidelity/resolution solution. FF-PINN architecture showed a faster convergence to the OpenFOAM Large Eddy simulation (LES) solution and better ability to learn the low/high spatial frequency components at early training epochs. Both architectures struggled to learn the high temporal frequency components in the velocity fields. In general the FF-PINN architecture is better than the PINN architecture at predicting the temporal evolution of the velocity fields. On the other hand, the FF-PINN architecture failed to learn the temporal evolution of the pressure field and suffered from over-fitting, since it was only able to predict the pressure at the training time stamps and discontinuous behavior is observed for the predictions at intermediate time stamps.

In conclusion, the Fourier Feature embeddings showed an improvement for Physics Informed

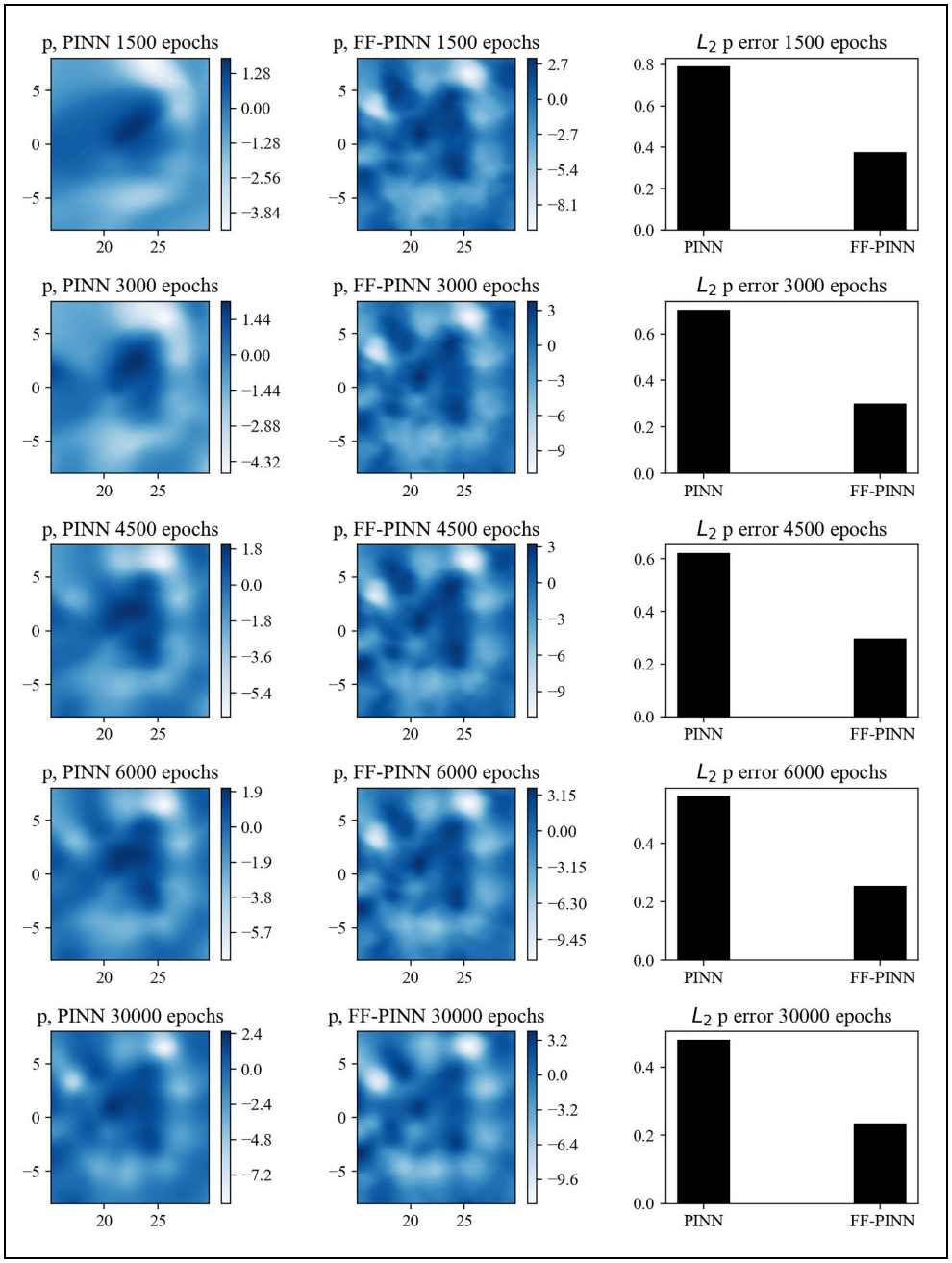

Figure 18. Pressure field prediction by PINN and FF-PINN architectures at $t = 14$ s. The reference finite volume solution is shown in Figure 15. The FF-PINN is able to learn the high frequency signal at early training epochs, whereas the PINN architecture struggles to learn the high frequency solution components due to the spectral bias.

Neural Networks ability to learn the solution for time dependent and independent problems that exhibit multi scale frequency periodic behavior in spatial and temporal domains. This improvement includes faster convergence to the solution, learning low and high frequency components synchronously, and a better fit to multi frequency solutions compared to the conventional PINN architecture that was not able to converge to noisy solution fields due the spectral bias problem. However, for a highly turbulent time dependent flow problem the FF-PINN failed to predict the temporal evolution of the pressure field except at the training time stamps due to over-fitting.

In future work, mini-batch training[33] will be implemented on sparse pressure data; mini-batch training is splitting the training dataset into small batches during training this has showed to reduce over-fitting.[34] Furthermore, the pressure Poisson equation will be substituted for the continuity equation in the PDEs set, this substitution is valid since the pressure Poisson equation satisfies the continuity equation and the flow incompressibility constraint. This could overcome the observed discontinuity in the temporal evolution of the pressure field.

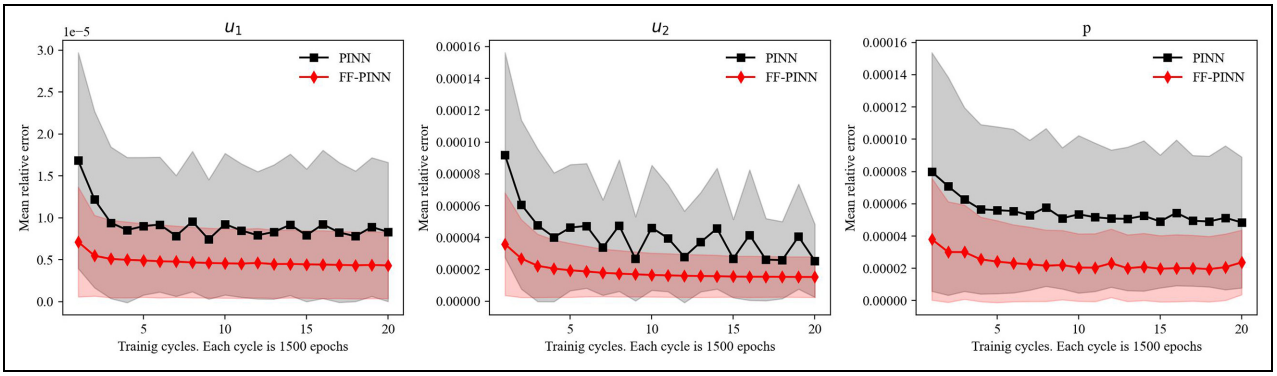

Figure 19. Mean relative error for the flow fields by the PINN and the FF-PINN architectures at $t = 14$ s per training cycle, the shaded area is the error standard deviation from the mean error.

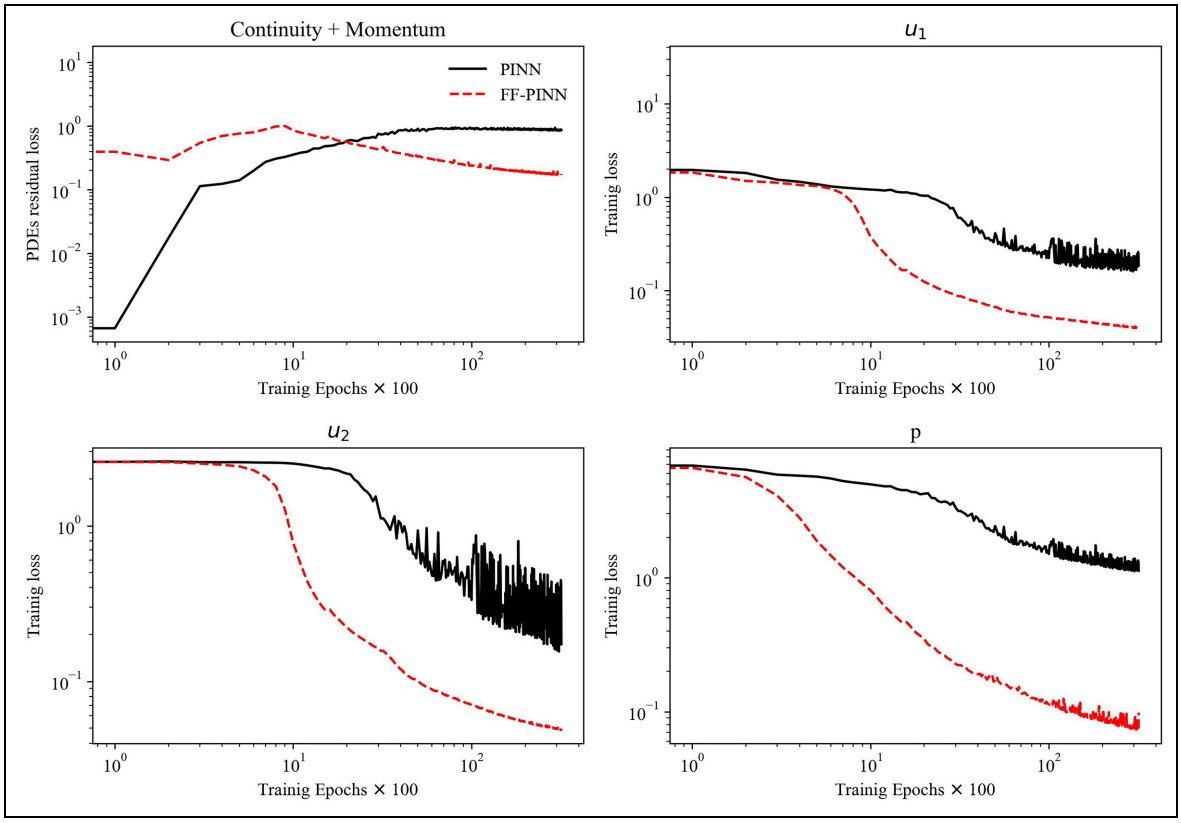

Figure 20. PDEs residual loss and training data loss for the spatio temporal domain per training iteration for PINN and FF-PINN architectures for flow over cylinder array problem at $\mathrm{Re} = 2.5 \times 10^5$. FF-PINN architectures has lower training loss for the velocity components and the pressure, furthermore it is better at satisfy the governing equations (continuity + momentum).

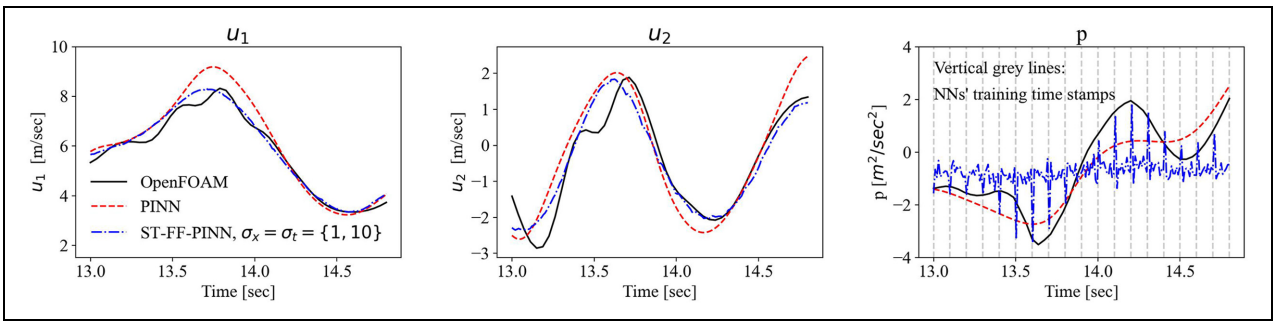

Figure 21. OpenFOAM, PINN and FF-PINN predictions for the temporal evolution of the flow fields at $(x_1 = 25, x_2 = 5)$ for flow over cylinder array problem at $\mathrm{Re} = 2.5 \times 10^5$.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) received no financial support for the research, authorship, and/or publication of this article.

ORCID iDs

Omar Sallam https://orcid.org/0000-0001-5117-8077
Mirjam Fürth https://orcid.org/0000-0003-0052-3437

References

1. Raissi M, Perdikaris P and Karniadakis GE. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint* arXiv:171110561. 2017.
2. Raissi M. Deep hidden physics models: deep learning of nonlinear partial differential equations. *J Mach Learn Res* 2018; 19(1): 932–955.
3. Raissi M, Yazdani A and Karniadakis GE. Hidden fluid mechanics: learning velocity and pressure fields from flow visualizations. *Science* 2020; 367(6481): 1026–1030.
4. Rahaman N, Baratin A, Arpit D, et al. On the spectral bias of neural networks. *PMLR* 2019; 97: 5301–5310.
5. Tancik M, Srinivasan P, Mildenhall B, et al. Fourier features let networks learn high frequency functions in low dimensional domains. *Adv Neural Inf Process Syst* 2020; 33: 7537–7547.
6. Williamson CH, Govardhan R, et al. Vortex-induced vibrations. *Annu Rev Fluid Mech* 2004; 36(1): 413–455.
7. Hamada AA and Furth M. Ground effect on current energy harvesting from a freely-oscillating circular cylinder at low Reynolds number. In: *International conference on offshore mechanics and arctic engineering*, 2021, vol. 85192, p.V009T09A003. New York: American Society of Mechanical Engineers.
8. Li Y and Lin M. Regular and irregular wave impacts on floating body. *Ocean Eng* 2012; 42: 93–101.
9. Wang S, Wang H and Perdikaris P. On the eigenvector bias of Fourier feature networks: from regression to solving multi-scale PDEs with physics-informed neural networks. *Comput Methods Appl Mech Eng* 2021; 384: 113938.
10. Lu L, Meng X, Mao Z, et al. DeepXDE: A deep learning library for solving differential equations. *SIAM Rev* 2021; 63(1): 208–228.
11. Raissi M, Perdikaris P and Karniadakis GE. Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J Comput Phys* 2019; 378: 686–707.
12. Baydin AG, Pearlmutter BA, Radul AA, et al. Automatic differentiation in machine learning: a survey. *J Mach Learn Res* 2018; 18: 1–43.
13. Abadi M, Agarwal A, Barham P, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint* arXiv:160304467, 2016.
14. Paszke A, Gross S, Massa F, et al. Pytorch: an imperative style, high-performance deep learning library. *Adv Neural Inf Process Syst* 2019; 32.
15. Bishop CM. Neural networks and their applications. *Rev Sci Instrum* 1994; 65(6): 1803–1832.
16. Nwankpa C, Ijomah W, Gachagan A, et al. Activation functions: Comparison of trends in practice and research for deep learning. *arXiv preprint* arXiv:181103378, 2018.
17. Glorot X and Bengio Y. Understanding the difficulty of training deep feedforward neural networks. In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics. JMLR Workshop and Conference Proceedings*, 2010, pp.249–256.
18. Kingma DP and Ba J. Adam: A method for stochastic optimization. *arXiv preprint*, arXiv:14126980. 2014.
19. Schöberl J. C + + *11 implementation of finite elements in NGSolve. Institute for analysis and scientific computing*. Vienna: Vienna University of Technology, 2014. pp.30.
20. Brezzi F and Falk RS. Stability of higher-order Hood–Taylor Methods. *SIAM J Numer Anal* 1991; 28(3): 581–590.
21. Sallam O, Feng R, Stason J, et al. *Computer vision techniques for floating structures experimental analysis: a heaving buoy case study*. 2022. https://papers.ssrn.com/sol3/papers.cfm?abstract_id = 4140113
22. Walsh EJ, Hancock D, Hines DE, et al. Wave-measurement capabilities of the surface contour radar and the airborne oceanographic lidar. *Johns Hopkins APL Tech Dig* 1987; 8(1): 74–81.
23. Salmon R. *Introduction to ocean waves*. San Diego, CA: Scripps Institution of Oceanography, University of California, San Diego, 2008.
24. Kim WJ, Van SH and Kim DH. Measurement of flows around modern commercial ship models. *Exp Fluids* 2001; 31(5): 567–578.
25. Shi W, Li M and Yuan Z. Investigation of the ship–seabed interaction with a high-fidelity CFD approach. *J Mar Sci Technol* 2021; 26(3): 931–946.
26. Deshpande SS, Anumolu L and Trujillo MF. Evaluating the performance of the two-phase flow solver interFoam. *Comput Sci Discov* 2012; 5(1): 014016.
27. Jasak H, Jemcov A, Tukovic Z, et al. OpenFOAM: A C + + library for complex physics simulations. In: *International workshop on coupled methods in numerical dynamics*, vol. 1000, 2007, pp.1–20. IUC Dubrovnik Croatia.
28. Holzmann T. *Mathematics, numerics, derivations and Openfoam®*. Loeben: Holzmann CFD, 2016.
29. Gisen D. Generation of a 3D mesh using snappyHexMesh featuring anisotropic refinement and near-wall layers. In: *ICHE 2014. Proceedings of the 11th international conference on hydroscience & engineering*, 2014, pp.983–990.
30. Leiteritz R and Pfluger D. How to avoid trivial solutions in physics-informed neural networks. *arXiv preprint*, arXiv:211205620, 2021.
31. Piomelli U. Large-eddy simulation: achievements and challenges. *Prog Aerosp Sci* 1999; 35(4): 335–362.
32. Liu DC and Nocedal J. On the limited memory BFGS method for large scale optimization. *Math Program* 1989; 45(1–3): 503–528.
33. Li M, Zhang T, Chen Y, et al. Efficient mini-batch training for stochastic optimization. In: *Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining*, 2014, pp.661–670.
34. Kandel I and Castelli M. The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset. *ICT Express* 2020; 6(4): 312–315.