Finite Basis PINNs

DLSC Project B

Guglielmo Pacifico, Fabian Jaeger

July 2023

Contents

1. Introduction

Although PINNs are a powerful framework to solve many simpler differential equations, they lack the ability to capture the complexity of high-frequency and multi-scale problems, which is sometimes referred to as the spectral bias of Neural Networks. This in turn implies that for increasingly complex problems, the PINNs network size grows rapidly in addition to having a more complex optimisation problem to solve.

To try and mitigate this problem, FBPINNs can be employed. Through a decomposition of the domain into smaller, partially overlapping subdomains, FBPINNs aim to turn the complex global problem into a easier to solve and smaller local optimisation problem. The goal of this report is to contrast the two frameworks and reproduce certain aspects of the paper "Finite Basis Physics-Informed Neural Networks (FBPINNs): a scalable domain decomposition approach for solving differential equations".

2. Methods and Problem-Statement

All Neural Networks in this report were optimized using the ADAM optimizer with a learning rate of 0.001.

2.1. FBPINN Framework

The approximate solution $\hat{u}(x)$ for a FBPINN Network to a differential equation over a domain Ω of the form

$$\begin{aligned} \mathcal{D}[\hat{u}(x);\lambda] &= f(x), \qquad x \in \Omega \\ \mathcal{B}_k[\hat{u}(x)] &= g_k(x), \qquad x \in \Gamma_k \subset \partial\Omega \end{aligned} \tag{1}$$

where $\mathcal{D}$, $\mathcal{B}$ represents the differential, boundary operator respectively, is given by

$$\hat{u}(x;\theta) = \mathcal{C}[\overline{NN}(x;\theta)]. \tag{2}$$

The constraining operator $\mathcal{C}$ is applied to the output of the Neural Network. This idea comes from the theory of functional connections (TFC) and transforms the problem from a constrained to a non-constrained problem.

The output of the Neural Network

$$\overline{NN}(x;\theta) = \sum_{i}^{n} w_i(x) \cdot \text{unnorm} \circ NN_i \circ \text{norm}_i(x) \tag{3}$$

is essentially a concatenation of the outputs of a Neural Network over the n individual subdomains, which is composed of the following operations:

- $\text{norm}_i(x)$: serves to normalize the domain of the input vector x in each subdomain to the range $[-1, 1]$.
- NN_i: The output of the Neural Network for each subdomain which has as input the normalized x over the subdomain i. The Neural Network architecture used here is a simple fully-connected feed-forward Network but it can in principle be any other architecture.
- unnorm: unormalises the output of the Neural Network to lie again in the range $[-1, 1]$.
- $\omega_i(x)$: defines the window function which serves to confine the output to each neural network to its respective subdomain. A condition that is imposed on these functions is that they are supposed to be differentiable so that the gradients of the Neural Network are again differentiable. We will make use of the following window function construction:

$$w_i(x) = \prod_{j}^{d} \phi\left(\left(x^j - a_i^j\right)/\sigma_i^j\right) \phi\left(\left(b_i^j - x^j\right)/\sigma_i^j\right) \tag{4}$$

  where j is the dimension of the input vector, a_i^j and b_i^j the midpoint of the left and right overlapping subdomain regions and

$$\phi(x) = \frac{1}{1 + e^{-x}} \tag{5}$$

  is the sigmoid function.

FBPINNs are then trained using the following unconstrained loss function (unconstrained refers to the fact that we only use the physics loss) for a given output (2):

$$\mathcal{L}(\theta) = \mathcal{L}_p(\theta) = \frac{1}{N_p} \sum_{i}^{N_p} \left\| \mathcal{D}\left[\hat{u}\left(x_i;\theta\right);\lambda\right] - f\left(x_i\right)\right\|^2 \tag{6}$$

using N_p sample points $\{x_i\}$ over the full domain Ω. The omission of the boundary loss is done in part because the boundary conditions are implemented using a suitable ansatz, as will be evident from the examples below.

2.2. Overview of experiments

2.2.1. One-dimensional, single frequency example

To verify our FBPINN code worked as expected, we first implemented the following one-dimensional ordinary differential equation:

$$\frac{du}{dx} = \cos(\omega x), \quad u(0) = 0 \tag{7}$$

once for $\omega = 1$ and once for $\omega = 15$ and compared it to the analytical solution

$$u(x) = \frac{1}{\omega}\sin(\omega x) \tag{8}$$

and the approximate PINN solutions. The following ansatz was used for the FBPINN and PINN solutions

$$\hat{u}(x;\theta) = \tanh(\omega x) NN(x;\theta), \tag{9}$$

which satisfies the boundary conditions $\mathcal{B}$ and possesses further useful properties (see original paper for more information). The loss function for the problem is then given by:

$$\mathcal{L}(\theta) = \mathcal{L}_p(\theta) = \frac{1}{N_p} \sum_{i}^{N_p} \left\| \frac{d}{dx}\hat{u}\left(x_i;\theta\right) - \cos\left(\omega x_i\right)\right\|^2 \tag{10}$$

2.2.2. Multi-scale problems

We were tasked with training a FBPINN for a multi-scale problem involving two frequencies.

$$\frac{\mathrm{d}u}{\mathrm{d}x} = \omega_1 \cos(\omega_1 x) + \omega_2 \cos(\omega_2 x), \quad u(0) = 0 \tag{11}$$

where $\omega_1 = 1$ and $\omega_2 = 15$. This problem has the analytical solution

$$u(x) = \sin(\omega_1 x) + \sin(\omega_2 x). \tag{12}$$

For both the PINN and FBPINN, the ansatz in this problem will take the form

$$\hat{u}(x;\theta) = \tanh(\omega_2 x) NN(x;\theta) \tag{13}$$

where now the unnormalisation is done by multiplying by 2.

The same principle can be extended to treat the general multi-scale ODE boundary problem, which takes the form

$$\frac{du}{dx} = \sum_{i=1}^{n} \omega_i \cos(\omega_i x), \quad u(0) = 0 \tag{14}$$

If n represents the number of frequencies in the multi-scale problem, we set the unnormalization to be n and the ansatz to take the shape

$$\hat{u}(x;\theta) = \tanh(\omega_n x) NN(x;\theta) \tag{15}$$

where ω_n represents the largest frequency in the problem.

2.2.3. Scalability of PINNs and FBPINNs with increasing number of multi-scale components

As a last question we wanted to investigate the scalability of both FBPINNs and PINNs with respect to the number of frequency components in the multi-scale problem. Additionally we studied the behaviour for a fixed multi-scale problem and how the number of subdomains for FBPINNs and number of hidden layers for PINNs affects the loss and training time.

3. Results

As in the paper, for low-frequencies (in our case $\omega = 1$) we expect both the PINN and FBPINN to be able to solve this problem with near-perfect accuracy. However for increasing ω, the spectral bias problem of Neural Networks in the PINN framework should become more apparent.

3.1. Single-frequency case

In the low-frequency case $\omega = 1$ we divide the domain $x \in [-2\pi, 2\pi]$ into $n = 5$ equidistant overlapping subdomains with a width of 1.3 of overlap. Each subdomain network is trained with 200 points and 10000 epochs and possesses the same architecture as in the paper with 2 hidden layers with 16 neurons each.

3.1.1. PINN Implementation

The PINN solutions to this problem for $\omega = 1$ and $\omega = 15$ can be found in the Figure (3) in the appendix.

3.1.2. FBPINN Implementation

For FBPINN the solution to the single-frequency ODE (7) with $\omega = 1$ can be found in Figure (4). For this simplified problem we again do not see any significant precision gains in using FBPINNs over PINNs solution, which are shown in Figure (3). However in the $\omega = 15$ case, we obtain the plots seen in Figure (5) for FBPINN, where FBPINNs clearly outperform PINNs (the results for the PINN can be seen in Figure (3)).

Furthermore the plots seem consistent with the results of the paper. However a different window function setup was used, so the individual network solutions seem to contribute in a different way than in the original paper.

3.2. Multi-Scale problem

The FBPINN individual network solutions, the full solution and three PINN solutions for varying neural network complexities can be seen in Figure (1). For training the PINNs, we used 50000 epochs with 3000 domain points. We can see that also the PINN manages to closely approximate the exact solution for the multi-scale problem with 5 layers and 128 hidden units per layer. However for an increasing amount of frequencies in the multi-scale problem we suspect that the gap in time and accuracy between the two frameworks will only increase.

3.3. Scalability of PINNs and FBPINNs with increasing number of multi-scale components

To further investigate this claim we wanted to see how the FBPINNs and PINNs scale with respect to the number of subdomains and number of layers added to PINNs to observe the computational cost in addition to scaling the complexity of the problem. As parameters we chose the following number of subdomains $[10, 20, 30, 40]$ for the multi-scale case $\omega = [2, 4, 8, 16, 32]$ and $[10, 30, 50, 100]$ for $\omega = [2, 4, 8]$. For the PINNs we kept the number of neurons fixed at 32 and varied the number of hidden layers from $[2, 3, 5, 8]$.

The results for the minimal L_1-loss and the time for training is shown in Figure (2). In Figure (6) one can see the evolution of the L_1-loss. It should be noted that we divide the training time for the FBPINNs by the number of subdomains since we didn't parallelized the training to keep a more accurate estimate of the actual training time. As expected the training time for PINNs grows linearly with respect to the depth of the Neural Network while if properly parallelized, the FBPINNs shouldn't have a significant increase of training time with respect to the increase of subdomains (other than the overlap regions that need to be trained sequentially). We also note a more stable convergence of the loss for the FBPINNs over the PINNs.

Since the $i = 3$ multi-scale problem is not too complex there seems to be no reason to use FBPINNs over PINNs except if time is relevant factor since the performance between the two is comparable.

Although also no significant performance gains can be observed by increasing the number of subdomains for the $i = 5$ multi-case problem, we suspect that with an increase of training points and a more accurate fine-tuning of the overlap and window functions, the same performance of the PINNs can be achieved at a fraction of the computational time if properly parallelized, especially when increasing the complexity of the problem.

A. Individual Contributions and personal motivation

Guglielmo implemented the baseline task. Fabian produced the report and did the extension task (a).

Personal Motivation

We chose this particular project in part because we can use the general insights for our thesis.

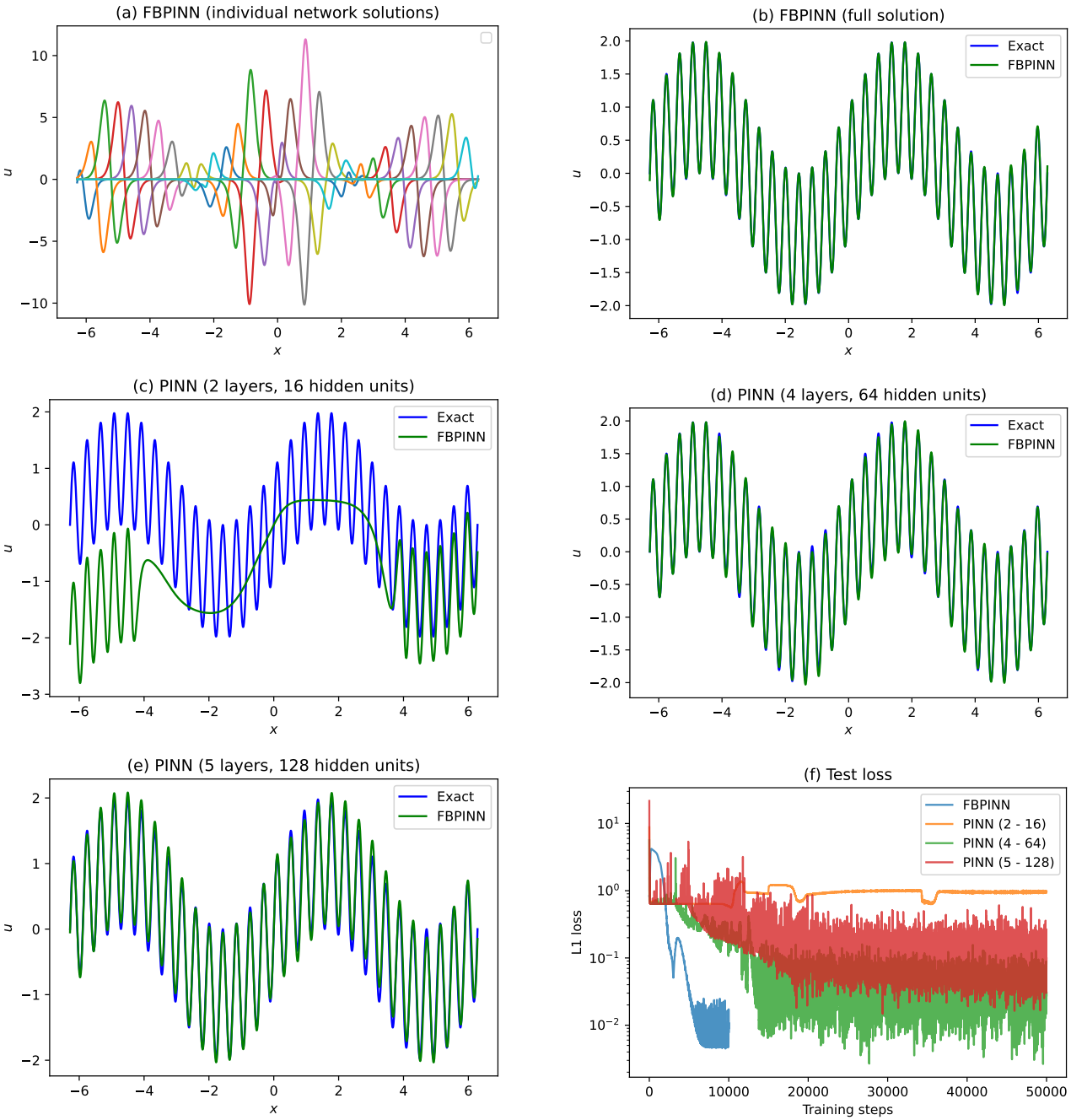

Figure 1: Performance of the FBPINN (a) - (b) and PINN (c), (d), (e) for different architecture complexities for the multi-scale problem (11) with $\omega_1 = 1$ and $\omega = 15$ (equivalent to Figure 7 in the paper)

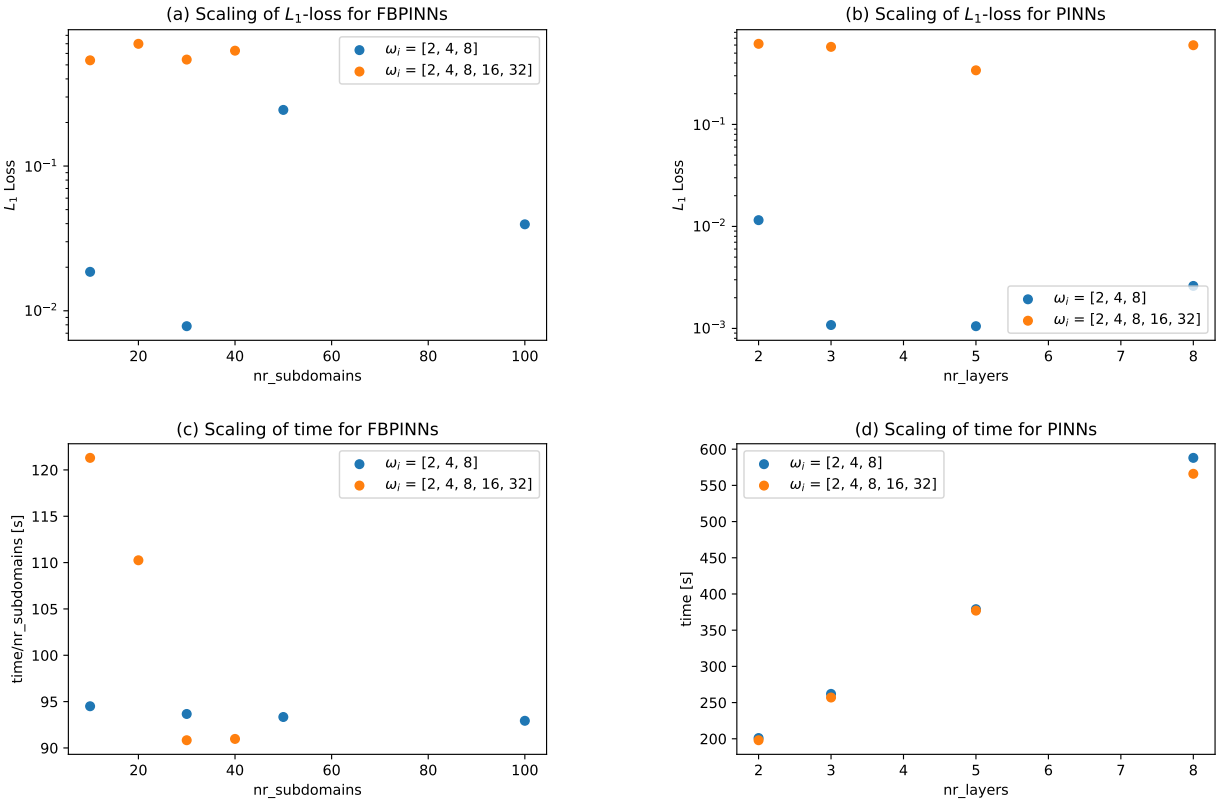

Figure 2: Evolution of the L_1-Loss for the multi-scale problem between the analytical and approximate (a) FBPINN and (b) PINN solution as well as the time-cost (c) and (d) as a function of number of subdomains and number of layers respectively. In all runs we used 50000 epochs with 3000 points.

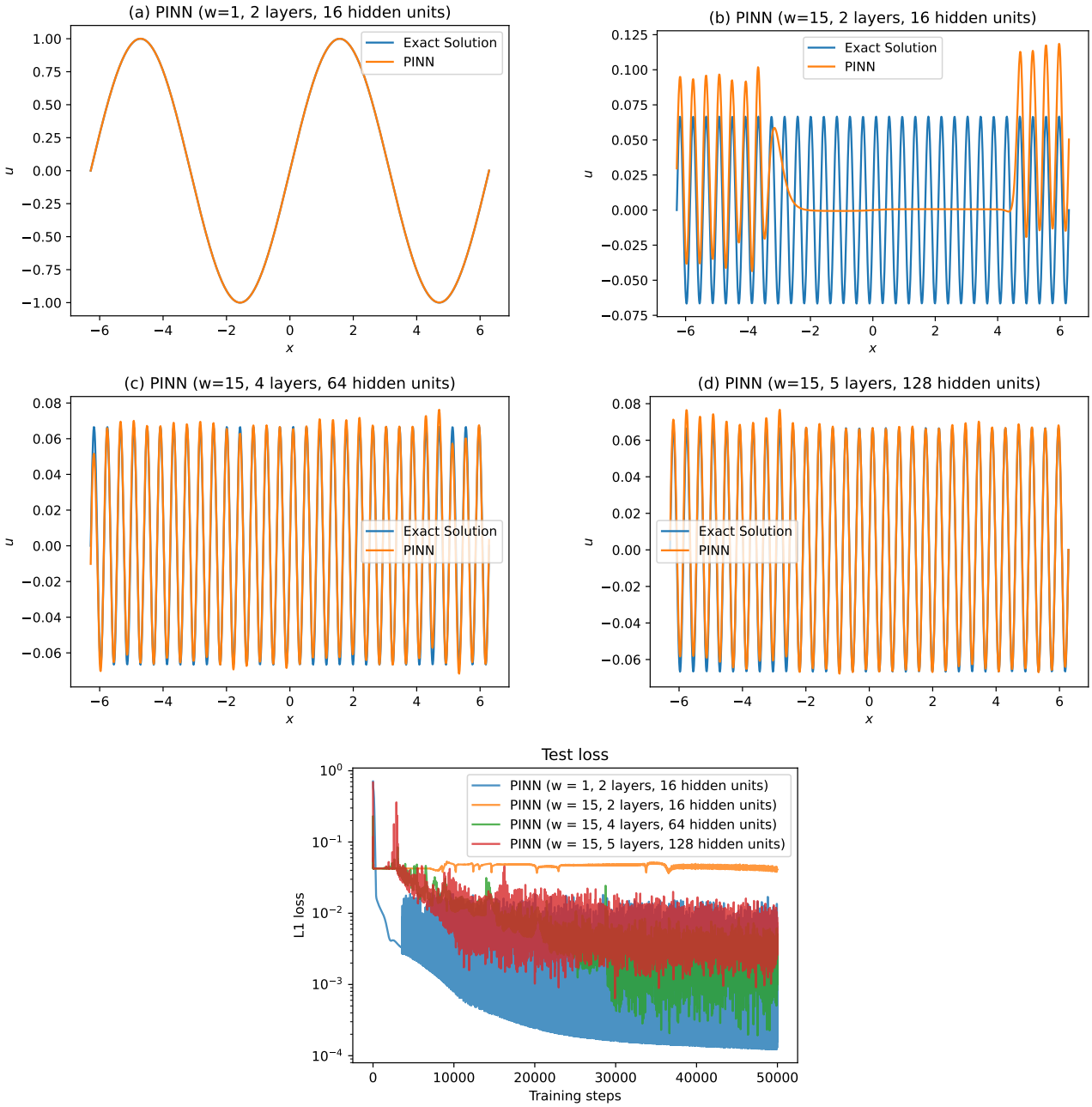

Figure 3: Performance of PINNs at solving the $\omega = 1$ (a) and $\omega = 15$ (b) - (d) single frequency ODE with varying network complexity.

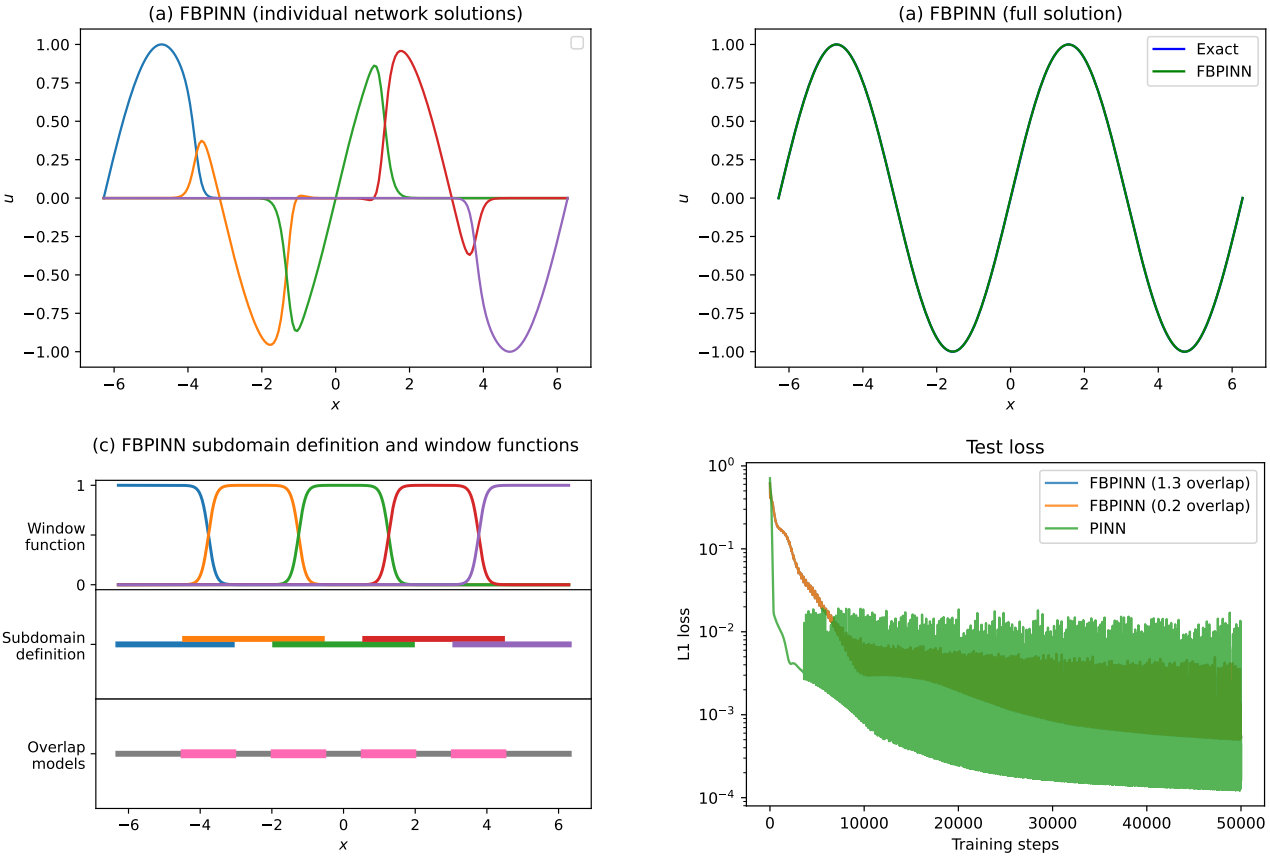

Figure 4: Performance of the FBPINN for the differential equation (7) (equivalent to Figure 5 in the original paper). As is evident from Figure (c), the domain of the problem going from $[-2\pi, 2\pi]$ is symmetrically decomposed with 5 equally sized window functions. The overlap was chosen to be 1.3 and the overlap function (4) was constructed using a uniform $\sigma_i^j = 0.1$.

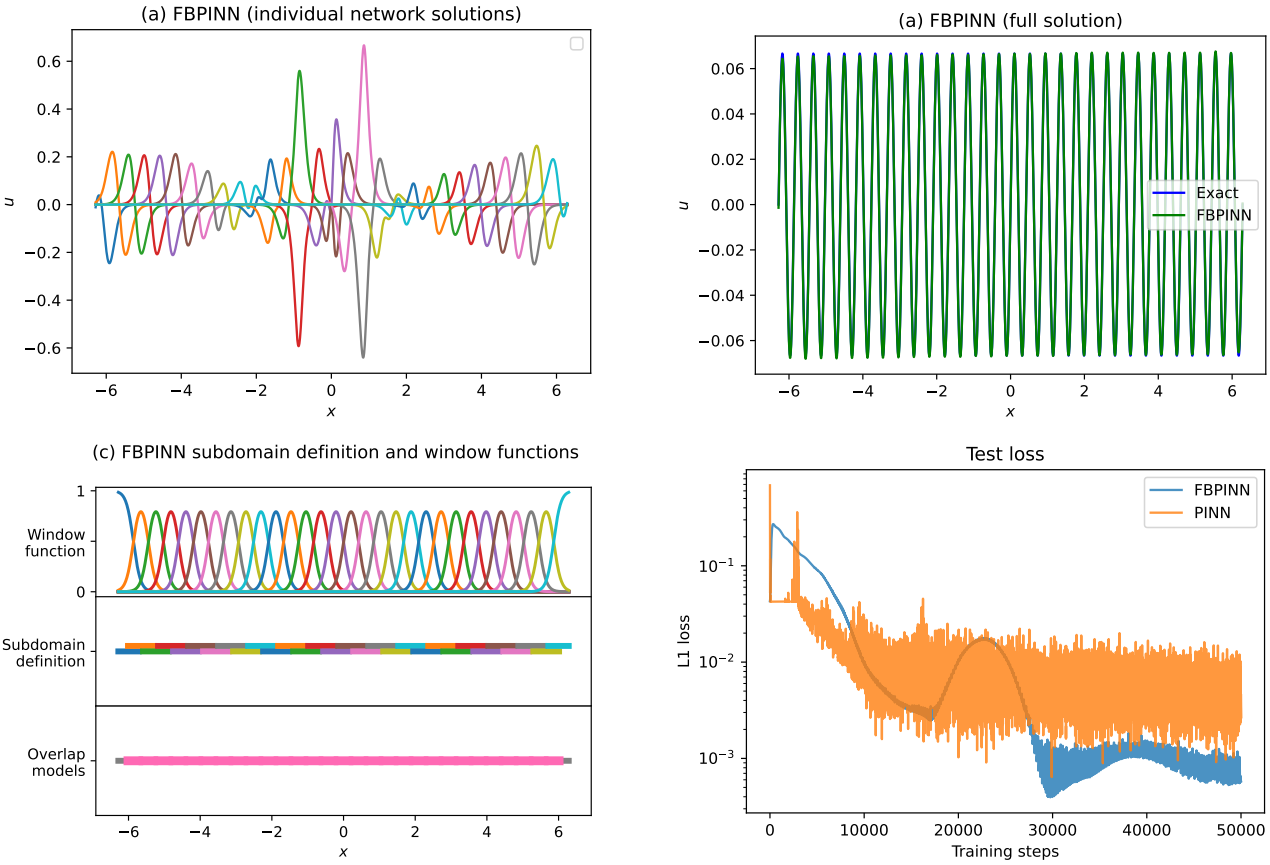

Figure 5: FBPINN for the simple one-dimensional ODE (7) with $\omega = 15$ (equivalent to Figure 6 in the original paper). In this case we chose 30 equidistant subdomains with an overlap of 1.3 and a Neural Network of 2 hidden layers with 16 neurons each. The window functions were constructed using $\sigma = 0.1$.

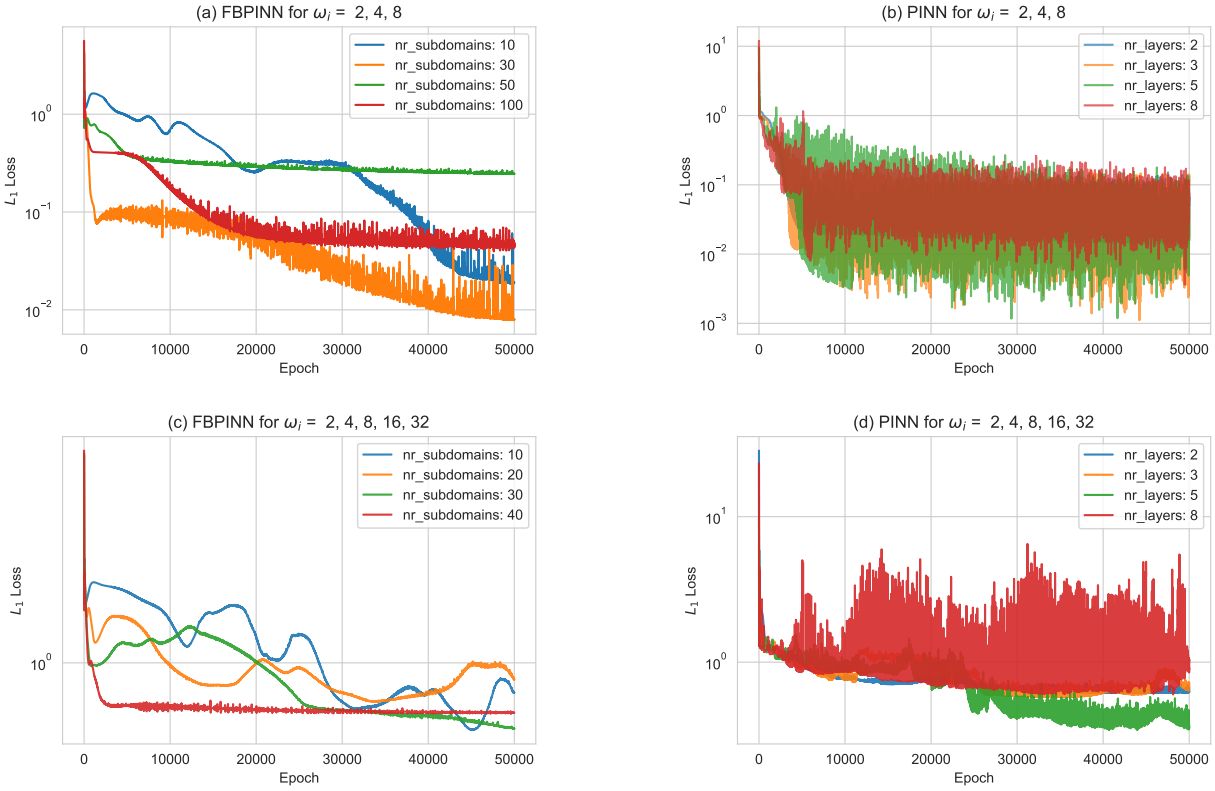

Figure 6: Evolution of the L_1-loss as a function of training epoch for the multi-scale problem of $\omega = 2^i$ with $i = 3$ (plots (a) and (b)) and $i = 5$ (plots (c) and (d)) for PINNs and FBPINNs.