A Universal PINNs Method for Solving Partial Differential Equations with a Point Source

Xiang Huang[1] , **Hongsheng Liu**[2] , **Beiji Shi**[2] , **Zidong Wang**[2] , **Kang Yang**[2] , **Yang Li**[2] , **Min Wang**[2] , **Haotian Chu**[2] , **Jing Zhou**[2] , **Fan Yu**[2] , **Bei Hua**[1] , **Bin Dong**[3] , **Lei Chen**[4*]

[1]University of Science and Technology of China
[2]Huawei Technologies Co. Ltd
[3]Peking University
[4]Hong Kong University of Science and Technology
sahx@mail.ustc.edu.cn, {liuhongsheng4, shibeiji, wang1, yangkang22, liyang477, wangmin106, chuhaotian2, zhoujing3, fan.yu}@huawei.com, bhua@ustc.edu.cn, dongbin@math.pku.edu.cn, leichen@cse.ust.hk

Abstract

In recent years, deep learning technology has been used to solve partial differential equations (PDEs), among which the physics-informed neural networks (PINNs) method emerges to be a promising method for solving both forward and inverse PDE problems. PDEs with a point source that is expressed as a Dirac delta function in the governing equations are mathematical models of many physical processes. However, they cannot be solved directly by conventional PINNs method due to the singularity brought by the Dirac delta function. In this paper, we propose a universal solution to tackle this problem by proposing three novel techniques. Firstly the Dirac delta function is modeled as a continuous probability density function to eliminate the singularity at the point source; secondly a lower bound constrained uncertainty weighting algorithm is proposed to balance the physics-informed loss terms of point source area and the remaining areas; and thirdly a multi-scale deep neural network with periodic activation function is used to improve the accuracy and convergence speed. We evaluate the proposed method with three representative PDEs, and the experimental results show that our method outperforms existing deep learning based methods with respect to the accuracy, the efficiency and the versatility.

1 Introduction

Partial differential equations (PDEs) play a significant role in science and engineering disciplines, which are used to formulate physical problems such as sound propagation, fluid flow, electromagnetic field, etc. In the past decades, numerical methods of PDEs have been developed and successfully applied to many real-world applications like aerodynamics and weather forecasting. However, as the scale and complexity of the problem increase, the computational complexity of numerical methods becomes prohibitively high.

On the other hand, with the increase use of numerical methods in both industry and academia, a growth in the collection of simulation data has been observed in computational fluid dynamics, electromagnetic simulation and many other areas, and data-driven deep learning approaches [Lu *et al.*, 2019; Li *et al.*, 2020] are proposed to alleviate the computational burden. Data-driven deep learning methods have the advantage that once the neural network is trained, the prediction process is fast. However, the cost of data acquisition is prohibitive for complex physical, biological, and engineering systems, and the generalization ability of the model (from one experimental condition to another) is poor when there are not sufficient labeled data [Cai *et al.*, 2021]. Therefore, physics-informed methods have been proposed in recent years that rely on the universal approximation ability of deep neural network [Pinkus, 1999] and automatic differentiation of the deep learning frameworks. Compared with traditional PDE solvers, deep learning solvers [Sirignano and Spiliopoulos, 2018; Weinan and Yu, 2018; Raissi *et al.*, 2019] can not only deal with parametrized simulations, but also address the inverse or data assimilation problems that traditional solvers cannot deal with. Among these methods, the physics-informed neural networks (PINNs) method [Raissi *et al.*, 2019] attracts increasing attentions from machine learning and applied mathematics communities due to its simple and effective way of approximating the solutions of PDEs using neural networks. The PINNs method preserves physical information in the governing equations and adapts to the boundary conditions. The PINNs method has been applied to solve many well-known PDEs such as Navier-Stokes equations [Jin *et al.*, 2021] and Maxwell's equations [Zhang *et al.*, 2021].

PDEs with a point source expressed as a Dirac delta function $\delta(\mathbf{x})$ have many applications in physical simulation. For instance, a point source can be a pulse exciting the electric field in electromagnetic simulation [Sullivan, 2013], or a sound source in an acoustic wave equation [Moseley *et al.*, 2020]. However, conventional PINNs method reviewed

*Corresponding author

above cannot be directly used to solve this problem due to the singularity brought by the Dirac delta function. Several methods have been proposed in the literature to solve this problem under certain constraints. For example, the Deep Ritz method [Weinan and Yu, 2018] and the variational method in NVIDIA SimNet [Hennigh *et al.*, 2021] need to transform the point source term into a computer computable form, which however is not applicable to all PDEs with a point source; PINNs-based methods proposed by [Zhang *et al.*, 2021; Moseley *et al.*, 2020; Bekele, 2020] require some labeled data as an additional loss term that may not be available in some cases.

In this paper, we propose a universal approach to solve PDEs with a point source based on the PINNs method. Our approach does not rely on any labeled data or variational forms, and meanwhile outperforms existing deep learning based methods with respect to the accuracy, the efficiency and the versatility. Our method employs three novel techniques:

- A continuous unimodal probability density function is used to model the Dirac delta function to eliminate the singularity at the point source;
- A lower bound constrained uncertainty weighting algorithm is proposed to balance the loss terms of the point source area and the remaining areas;
- A neural network architecture is built with multi-scale DNN [Xu *et al.*, 2019] and periodic activation functions to improve the accuracy and convergence speed of the PINNs method.

2 Preliminaries

The general form of a time-dependent PDE is given by:

$$\begin{aligned} \mathcal{N}(u(\mathbf{x},t)) &= f(\mathbf{x},t), \quad (\mathbf{x},t) \in \Omega \times [0,T] \\ u(\mathbf{x},t) &= g(\mathbf{x},t), \quad (\mathbf{x},t) \in \partial\Omega \times [0,T] \\ u(\mathbf{x},0) &= h(x), \quad \mathbf{x} \in \Omega \end{aligned} \tag{1}$$

where $\mathcal{N}(\cdot)$ is the differential operator, u is the solution, and $f(\mathbf{x},t)$ is the time-dependent forcing term. For PDEs with a point source that we consider in this paper, the forcing term $f(\mathbf{x},t)$ contains the Dirac delta functions.

2.1 PDEs with a Point Source

Here we introduce two well-known time-(in)dependent PDEs with a point source forcing term. The Poisson's equation with a point source has the following form:

$$\begin{aligned} -\Delta u &= \delta(\mathbf{x}-\mathbf{x}_0), \quad \mathbf{x} \in \Omega, \\ u &= 0, \quad \mathbf{x} \in \partial\Omega \end{aligned} \tag{2}$$

where δ denotes the Dirac delta function. In the field of electromagnetic simulations, the time domain Maxwell's equations with excitation can be expressed as

$$\begin{aligned} \nabla \times E &= -\mu \frac{\partial H}{\partial t} - J(\mathbf{x},t), \\ \nabla \times H &= \epsilon \frac{\partial E}{\partial t}. \end{aligned} \tag{3}$$

where ϵ is the permittivity, μ is the permeability. The forcing term $J(\mathbf{x},t) = \delta(\mathbf{x}-\mathbf{x}_0)g(t)$ excites a wave propagation at $\mathbf{x}_0$ with a particular type of pulse $g(t)$.

2.2 The PINNs Method

In order to solve PDE problems with deep learning technology, the PINNs method is proposed to approximate the solution $u(\mathbf{x},t)$ with a deep neural network $u(\mathbf{x},t;\theta)$, where θ represents the trainable parameters of the deep neural network. The loss function given by such approximation is defined as

$$L_{total}(\theta) = L_r(\theta) + \lambda_{ic}L_{ic}(\theta) + \lambda_{bc}L_{bc}(\theta), \tag{4a}$$

$$L_r(\theta) = \frac{1}{N_r}\sum_{i=1}^{N_r} ||\mathcal{N}(u(\mathbf{x}_i,t_i;\theta)) - f(\mathbf{x}_i,t_i)||_2^2, \tag{4b}$$

$$L_{ic}(\theta) = \frac{1}{N_{ic}}\sum_{i=1}^{N_{ic}} ||u(\mathbf{x}_i,0;\theta) - h(\mathbf{x}_i)||_2^2, \tag{4c}$$

$$L_{bc}(\theta) = \frac{1}{N_{bc}}\sum_{i=1}^{N_{bc}} ||u(\mathbf{x}_i,t_i;\theta) - g(\mathbf{x}_i,t_i)||_2^2. \tag{4d}$$

where L_r, L_{ic}, L_{bc} correspond to the loss term of PDE residual, initial conditions, and boundary conditions, respectively. The hyperparameters λ_{ic} and λ_{bc} aim to balance the interplay of different terms in the loss function. N_r, N_{ic}, and N_{bc} represent the number of samples in the entire region, initial conditions, and boundary conditions, respectively.

When $f(\cdot), g(\cdot), h(\cdot)$ are continuous, the network weights θ can be optimized by minimizing the total training loss $L_{total}(\theta)$ via standard gradient descent procedures used in deep learning. Unfortunately, due to the singularity brought by $\delta(\mathbf{x})$, the PINNs method cannot be directly used to solve the PDEs with a point source.

2.3 Related Works

One can use the Deep Ritz method [Weinan and Yu, 2018] to solve Eq.(2). Through mathematical derivation, Eq.(2) can be converted to the following minimization problem:

$$\min \int_\Omega \frac{1}{2}\|\Delta u(\mathbf{x})\|^2 d\mathbf{x} - u(\mathbf{x}_0) + \int_{\partial\Omega} \|u(\mathbf{x})\|^2 d\mathbf{x} \tag{5}$$

where the Dirac function term $\delta(\mathbf{x}-\mathbf{x}_0)$ in Eq.(2) will generate the constant term $u(\mathbf{x}_0)$ in Eq.(5) after integration. However, the Deep Ritz method is unable to solve PDEs like Eq.(3) that cannot be derived from any variational problem. The NVIDIA SimNet [Hennigh *et al.*, 2021] uses variational formulation to solve PDEs with a point source. However, its performance heavily depends on the selection of the test functions and the computational time increases linearly with the number of test functions.

[Zhang *et al.*, 2021] and [Moseley *et al.*, 2020] use classical numerical method to simulate the early period ($0 \le t \le T_0 < T$) of wave equations to avoid point source singularity, and then applied the PINNs method to solve the PDEs with given initial conditions at $t = T_0$. [Bekele, 2020] attempted to leverage the PINNs loss by adding a new loss term with a significant amount of known labels when solving the Barry and Mercer's source problem with time-dependent fluid injection. However, these methods rely on classical computational methods to generate the labeled data which is expensive or even infeasible in many situations.

In this work, we propose a universal solution based on the PINNs method that does not rely on any labeled data or variational forms.

3 Methodology

In this section, we explain our strategy to tackle the singularity problem in detail.

3.1 Approximation to $\delta(\mathbf{x})$

The Dirac delta function $\delta(x)$ is mathematically defined by:

$$\begin{aligned} \delta(x) &= \begin{cases} +\infty, & x = 0 \\ 0, & x \neq 0 \end{cases} \\ \int_{-\infty}^{+\infty} \delta(x)dx &= 1. \end{aligned} \tag{6}$$

Taking $\eta(x)$ to be any symmetric unimodal continuous probability density function centered at 0, $\eta(x) = \alpha^{-1}\eta(\frac{x}{\alpha})$ can be a natural approximation to $\delta(x)$ as the kernel width $\alpha \to 0$. In a d-dimensions space, $\eta_\alpha(\mathbf{x}) = \alpha^{-d}\eta(\frac{\mathbf{x}}{\alpha})$ can be used instead. In our experiments, we use Gaussian distribution, Cauchy distribution and Laplacian distribution with sufficiently small kernel width α to approximate $\delta(\mathbf{x})$.

3.2 Lower Bound Constrained Uncertainty Weighting

Our approximation to $\delta(x)$ eliminates function discontinuity at the origin, but it also brings a difficulty for the PINNs method to optimize its training loss. As the probability density function is highly concentrated at the origin as $\alpha \to 0$, the residual samples close to the origin tend to dominate the whole residual loss, making it difficult for the network to learn the governing equations and therefore the standard optimizer may not converge to the solution. To address this problem, we split the feasible region into two subdomains such that:

$$\Omega = \Omega_0 \cup \Omega_1 \quad \Omega_0 \cap \Omega_1 = \emptyset$$

where Ω_0 represents the subdomain that contains the origin and Ω_1 covers the complement region. By taking this decomposition, the residual loss L_r in Eq.(4) is divided into two parts:

$$L_r(\theta) = \lambda_{r,0}L_{r,0}(\theta) + \lambda_{r,1}L_{r,1}(\theta) \tag{7}$$

where $L_{r,0}(\theta)$ and $L_{r,1}(\theta)$ correspond to the residual losses in Ω_0 and Ω_1, respectively. In our experiment, for the point source excited at $\mathbf{x}_0 \in \Omega$, we split Ω as

$$\Omega_0 = \{\mathbf{x}_0 + \mathbf{x} \in \Omega \,|\, \|\mathbf{x}\| \leq 3\alpha\}, \quad \Omega_1 = \Omega \backslash \Omega_0.$$

Together with the IC&BC conditions, the total PINNs loss in Eq.(4a) changes to be:

$$\begin{aligned} L_{total}(\theta) =& \lambda_{r,0}L_{r,0}(\theta) + \lambda_{r,1}L_{r,1}(\theta) + \lambda_{ic}L_{ic}(\theta) + \\ & \lambda_{bc}L_{bc}(\theta). \end{aligned} \tag{8}$$

Properly setting the weight vector $\lambda = (\lambda_{r,0}, \lambda_{r,1}, \lambda_{ic}, \lambda_{bc})$ is critical to enhance the trainability of constrained neural networks [McClenny and Braga-Neto, 2020; Wang *et al.*, 2020], but searching the optimal weight vector through manual tuning is infeasible.

In multi-task learning, different tasks share the same backbone network but correspond to different loss terms. These loss terms need to be weighted to balance their contributions to the total loss during training. Several algorithms [Kendall *et al.*, 2018; Heydari *et al.*, 2019] have been proposed to solve the loss weighting problem in multi-task learning, and some of them are utilized in PINNs method. Specifically, [Wang *et al.*, 2021] propose a learning rate annealing algorithm that uses back-propagation gradient statistics in the training process to adaptively balance the contribution of each loss term, and [Bischof and Kraus, 2021] propose ReLoBRaLo method to perform loss weighting based on the SoftAdapt method proposed by [Heydari *et al.*, 2019].

In this work, we adapt the uncertainty weighing method in [Kendall *et al.*, 2018] to solve the problem. Specifically, for a total loss function including m loss terms $L(\theta) = \sum_{i=1}^{m} \lambda_i L_i(\theta)$, we add a nonnegative ϵ^2 to w^2, and get the total PINNs loss function as follows:

$$L_{total}(\theta; \mathbf{w}) = \sum_{i=1}^{m} \frac{1}{2(\epsilon^2 + w_i^2)} L_i(\theta) + \log(\epsilon^2 + w_i^2) \tag{9}$$

The uncertainty of the i-th loss term is approximated by $\epsilon^2 + w_i^2$, whose lower bound is constrainted by ϵ^2 when decreasing w_i to 0. We argue that such reformulation is nontrivial since it provides a lower bound for the uncertainty estimation of each loss term and thus increases the stability of the self-adaptive weighting algorithm.

Fig.1 shows the convergence process of the mean L_2 $error$s with different loss weighting methods in solving Barry and Mercer's source problem. It shows that our lower bound constrained uncertainty weighting method outperforms other methods. SoftAdapt [Heydari *et al.*, 2019] and ReLoBRaLo [Bischof and Kraus, 2021] methods both fail to converge to good accuracies in this scenario, even worse than the equally weighted PINNs method ('Original PINNs'). We believe this is due to the extreme imbalance between $L_{r,0}(\theta)$ and other loss terms, whereas SoftAdapt and ReLoBRaLo can only deal with cases where differences between loss terms are not significant. The learning rate annealing algorithm ('LR Annealing') is better than the original uncertainty weighting method ('Uncertainty') [Kendall *et al.*, 2018], but worse than our proposed method. It is worth noting that the learning rate annealing algorithm requires the back-propagation gradient statistics, which brings additional training time overhead.

3.3 Multi-Scale DNNs with Periodic Activation Function

For a 2-D Possion's equation with a point source located at (x_0, y_0) within a domain $\Omega = (0, \pi)^2$, the analytical solution is as follows:

$$u(x, y) = \frac{4}{\pi^2} \sum_{m,n=1}^{\infty} \frac{\sin mx \sin mx_0 \sin ny \sin ny_0}{m^2 + n^2} \tag{10}$$

The analytical solution is formed by superposition of multiple frequency components. This multi-frequency phenomenon is

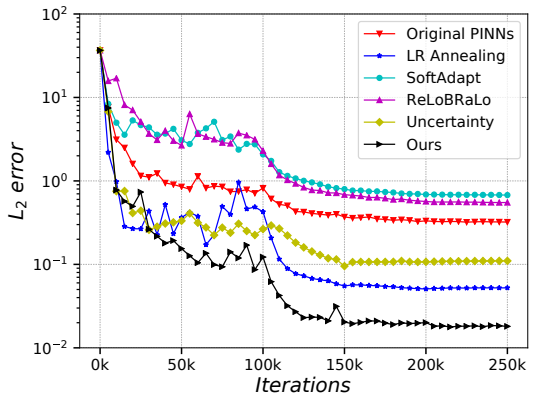


Figure 1: Barry and Mercer's source problem: Convergence speed of the mean L_2 *error*s with different loss weighting methods. *Original PINNs* means the equally weighted PINNs method.

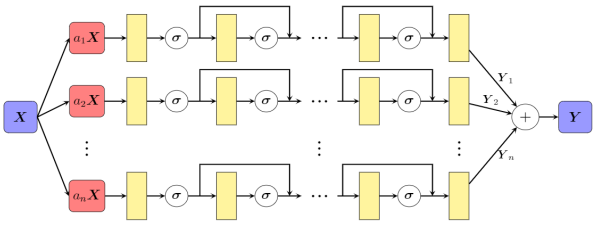


Figure 2: Architecture of MS-SIREN that consists of n subnets with different scaling parameter $\{a_1, \ldots, a_n\}$ and the activation function $\sigma(\mathbf{x}) = \sin(\mathbf{x})$.

common in solutions of many PDEs. Recent development in deep learning theories [Xu *et al.*, 2019] reveals that the neural network fits the hidden physical law from low to high frequency components. Based on this theory, [Liu *et al.*, 2020] proposes multi-scale deep neural networks (MscaleDNNs) which exhibit faster convergence in higher frequency components.

Periodic nonlinearity has been extensively investigated over the past decades. Recently SIREN architecture [Sitzmann *et al.*, 2020] is proposed to use *Sine* function as a periodic activation function and outperforms alternative activation functions when dealing with implicit neural representations and their derivatives. Compared with other activation functions, e.g., *Tanh* and *ReLU*, derivative of a *Sine* function is *Cosine*, a phase-shifted *Sine*. Therefore, derivatives of a SIREN inherit the SIREN's properties, thus enabling us to supervise any derivative of SIREN with complex natural signals from PDEs.

Borrowing ideas from MscaleDNNs and SIREN, we propose Multi-Scale SIREN (MS-SIREN) that combines multi-scale DNNs with *Sine* activation function, as shown in Fig.2. Skip connections are added between consecutive hidden layers to accelerate training and improve accuracy as they help to avoid the vanishing gradient problem [He *et al.*, 2016]. Fig.3 shows the effectiveness of MS-SIREN compared with other alternative architectures.

Network Architectures			L_2 *error*			
# subnets	# layers	# neurons	E_x	E_y	H_z	mean
1	7	256	0.192	0.183	0.433	0.269
1	9	256	0.147	0.146	0.070	0.121
2	7	128	0.079	0.074	0.058	0.072
2	9	128	0.054	0.053	0.019	0.027
4	7	64	0.021	0.022	0.001	**0.018**
4	9	64	0.025	0.022	0.017	0.021

Table 1: Maxwell's equations: L_2 *error*s achieved by different network architectures. '# subnets' indicates the number of subnets in the multi-scale network; '# layers' indicates the number of full connected layers of each subnets; '# neurons' indicates the number of neurons of each layer.

4 Numerical Experiments

To evaluate the effectiveness of our proposed method, we apply it to solve three classical point source problems which come from different fields of physics: (1) electromagnetic simulation with transverse electric (TE) mode [Gedney, 2011]; (2) Poisson's equation with a point source; (3) Barry and Mercer's source problem with time-dependent fluid injection [Bekele, 2020]. Unless otherwise specified, we take the following default training settings:

- The Dirac delta function $\delta(\mathbf{x})$ is approximated by a Gaussian distribution with default kernel width $\alpha = 0.01$.
- $\epsilon = 0.01$ is chosen for the lower bound constrained uncertainty weighting algorithm.
- For MS-SIREN architecture, four subnets are used with scale coefficients set to $\{1, 2, 4, 8\}$.
- The model is trained using Adam optimizer with an initial learning rate of 0.001, and the learning rate attenuates 10 times when the training process reaches 40%, 60%, and 80%, respectively.

Average relative L_2 *error* is used to evaluate the performance of the trained network $u(\mathbf{x}; \theta)$, which is computed as:

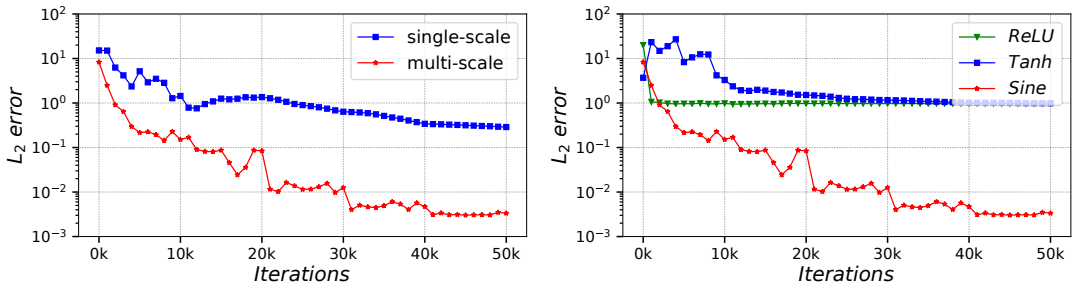
$$L_2 \; error = \frac{\sum_{j=1}^{N} \|u_{ref}(\mathbf{x}_j) - u(\mathbf{x}_j, \theta)\|_2}{\sum_{j=1}^{N} \|u_{ref}(\mathbf{x}_j)\|_2} \tag{11}$$

where N is the number of test points in Ω and $u_{ref}(\cdot)$ represents the ground truth. Unless otherwise specified, all the experiments are conducted under the MindSpore[1] and trained on the Ascend 910 AI processors[2]. Readers can read our source code[3] for implementation details.

4.1 Electromagnetic Simulation with TE Mode

Maxwell's equations are a set of coupled partial differential equations related to the fundamental physical modeling of electromagnetism. We use 2-D Maxwell's equations to validate the robustness of our method. The governing equations

[1]https://www.mindspore.cn/
[2]https://e.huawei.com/en/products/servers/ascend
[3]https://gitee.com/mindspore/mindscience/tree/master/MindElec/



(a) Single-scale VS. Multi-scale.

(b) Activation functions.

Figure 3: Poisson's equation: Comparative results under different architectures. (a) shows a single-scale network failing to converge to the solution. (b) shows the *Sine* activation functions outperforming other alternatives.

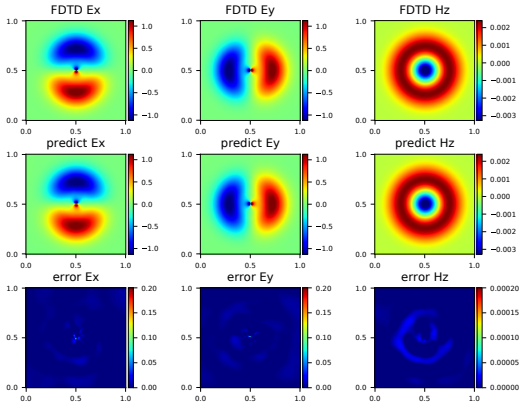


Figure 4: Maxwell's equations: Solutions at the time of $2.4ns$. Top: The numerical solution (E_x, E_y, H_z) obtained by FDTD method; Middle: The predicted solution (E_x, E_y, H_z) output by our model; Bottom: The absolute error between model prediction and the reference solution.

of TE wave are given as:

$$\frac{\partial E_x}{\partial t} = \frac{1}{\epsilon}\frac{\partial H_z}{\partial y}, \tag{12a}$$

$$\frac{\partial E_y}{\partial t} = -\frac{1}{\epsilon}\frac{\partial H_z}{\partial x}, \tag{12b}$$

$$\frac{\partial H_z}{\partial t} = -\frac{1}{\mu}(\frac{\partial E_y}{\partial x} - \frac{\partial E_x}{\partial y} + J). \tag{12c}$$

where E_x, E_y and H_z are the electromagnetic fields. The constants μ and ϵ are known as the permeability and permittivity of the free space. A Gaussian pulse containing multiple frequency components is used as the source function that is expressed as:

$$J(x, y, t) = e^{-(\frac{t-d}{\tau})^2}\delta(x - x_0)\delta(y - y_0). \tag{13}$$

where d is the temporal delay and τ is a pulse-width parameter. The initial electromagnetic fields are zero and the standard Mur's second-order absorbing boundary condition [Gedney, 2011] is utilized. We solve the problem in a rectangular domain $\Omega = [0, 1]^2$, and the point source is located at the point $(x, y) = (0.5, 0.5)$. The total simulation time is set to $4ns$ so that the pulse signals can propagate throughout the whole space of truncated domain. The reference solution is obtained through the finite-difference time-domain (FDTD) method [Gedney, 2011] at a spatial resolution of $\Delta = 0.005$. The spatial point source is approximated by the Gaussian distribution with kernel width $\alpha = 2\Delta = 0.01$.

Network Architectures			L_2 *error*
# subnets	# layers	# neurons	
1	5	256	0.289
1	7	256	1.081
2	5	128	0.003
2	7	128	0.006
4	5	64	0.003
4	7	64	**0.002**

Table 2: Poisson's Equation: L_2 *error*s obtained by different network architectures.

The instantaneous electromagnetic fields at the time of $2.4ns$ are showed in Fig.4. We can see that the predicted electromagnetic fields by our method are almost indistinguishable from the reference solution. The SIREN architecture (# subnets = 1) and our MS-SIREN architecture are used to solve the Maxwell's equations with a point source under different settings, and the final L_2 *error*s of all output components and their mean values are shown in Table 1. It is obvious that MS-SIREN outperforms SIREN under all settings, and the best performance is achieved by the architecture comprising four subnets.

In order to approximate the Dirac delta function, the setting of kernel width α is important. In our experiments we find that the training process may not converge if α is too large or too small. We adopt an online fine-tuning method to decrease the value of α gradually during the training, which is illustrated in Fig.5 where the Gaussian distribution is used. The training process starts with $\alpha = 0.01$, the L_2 *error* drops rapidly and after 140k iterations the final mean L_2 *error* is stable below 0.05. Then the online fine-tuning is performed to reduce the value of α by half, which makes the L_2 *error* rise first but then decrease quickly as the training progresses, and after 70k iterations the final mean L_2 *error* is stable below 0.08. The fine-tuning operation repeats twice that reduces the

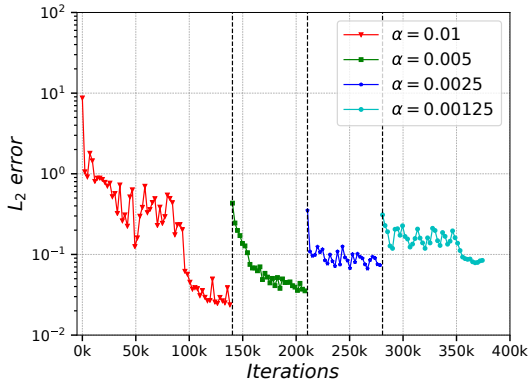


Figure 5: Maxwell's equation: The mean L_2 *error* when different kernel widths are used. The value of α is set to 0.01 for the first 140k iterations and then halved every 70k iterations.

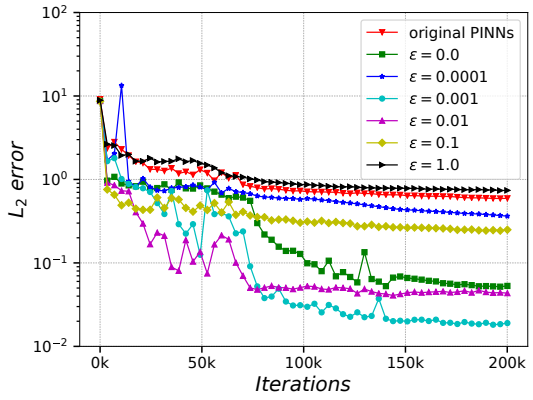


Figure 6: Maxwell's equations: L_2 *error*s for different uncertainty lower bound. 'Original PINNs' means the equally weighted PINNs method.

value of α to 0.0025 and 0.00125, respectively. According to the experimental results in Fig.5, we choose $\alpha = 0.01$ as the kernel width and fix it in the other experiments. We further use Cauchy distribution and Laplacian distribution with $\alpha = 0.01$ to approximate the point source, and get the final mean L_2 *error*s as 0.04 and 0.05, respectively. This set of experiments demonstrate that approximating the Dirac delta function with a symmetric unimodal continuous probability density function is feasible.

Fig.6 shows the L_2 *error*s when different ϵ in Eq.(9) is used in solving Maxwell's equations with a point source. When $\epsilon = 0.01$ and $\epsilon = 0.001$, the lower bound constrained uncertainty weighting method outperforms the original uncertainty weighting method ($\epsilon = 0$) with not only lower final L_2 *error*s, but also faster convergence speed.

The MS-SIREN architecture uses skip connections between consecutive hidden layers. To validate its effectiveness, we compare MS-SIREN architectures with and without skip connections, and show the results in Fig.7. It is clear that skip connections make L_2 *error* decrease faster and reach a lower value.

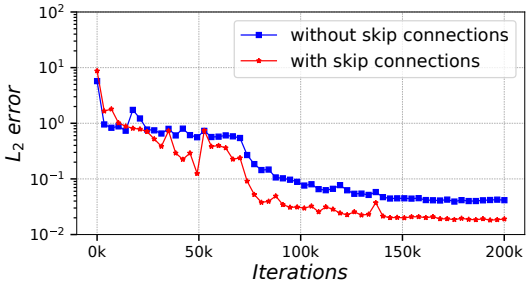


Figure 7: Maxwell's equations: The mean L_2 *error*s got by MS-SIREN with/without skip connections.

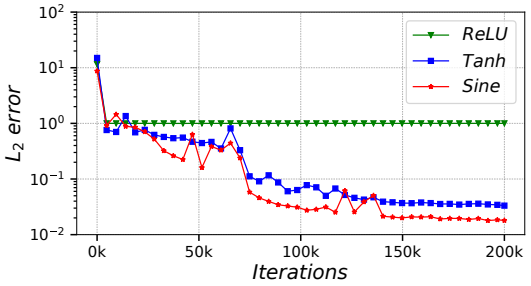


Figure 8: Maxwell's equations: Influence of activation functions on the prediction accuracy.

Activation function is another key factor affecting the prediction accuracy. To validate the effectiveness of *Sine* periodic activation function, we compare it with *ReLU* and *Tanh*, and the results are shown in Fig.8. *Sine* produces the best results in terms of convergence speed and the final mean L_2 *error*. In comparison, *Tanh* performs slightly worse than *Sine*, and *ReLU* fails to let the network learn any useful information.

4.2 Poisson's Equation with a Point Source

We apply our approach to solve Eq.(2) in the region of $\Omega = [0, \pi] \times [0, \pi]$ with the point source located at $\mathbf{x}_0 = (\frac{\pi}{2}, \frac{\pi}{2})$. The point source is approximated using Gaussian distribution with fixed $\alpha = 0.01$ and the lower bound hyperparameter ϵ is set to 0.01.

Table 2 shows the prediction accuracy obtained by different network architectures. The SIREN architecture (# subnets =

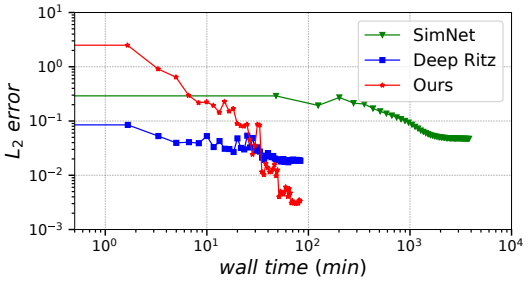


Figure 9: Poisson's Equation: Comparison of different methods.

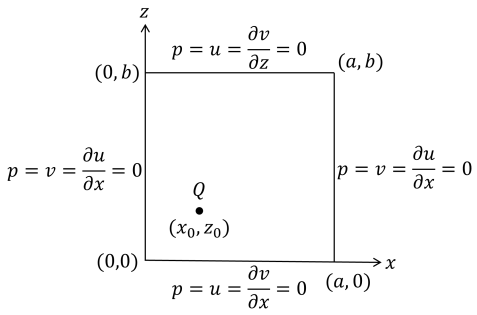


Figure 10: Barry and Mercer's source problem: Illustration of the computational domain and boundary conditions.

1) cannot converge in all settings due to the singularity problem. However, when the number of subnets increases to 2 or 4, the L_2 *error* drops remarkably. This experiment indicates that a multi-scale structure is necessary to solve this problem.

For comparison, we also apply Deep Ritz method and SimNet variational method to solve the problem. Fig.9 shows that the convergence speed of our method is close to that of Deep Ritz method, but faster than that of SimNet. The final L_2 *error* obtained by our method is 0.003, which is significantly better than Deep Ritz (0.018) and SimNet (0.047). Moreover, our method is a more general approach that is not limited by the type of PDEs, whereas the Deep Ritz method and SimNet variational methods are the opposite.

4.3 Barry and Mercer's Source Problem

The third problem we solve is the nondimensionalized poroelastic Barry and Mercer's source problem with time-dependent fluid injection. The governing equations are given by

$$\frac{\partial^2 u}{\partial t \partial x} + \frac{\partial^2 v}{\partial t \partial z} - \frac{\partial^2 p}{\partial x^2} - \frac{\partial^2 p}{\partial z^2} - \beta Q = 0, \quad (14a)$$

$$(\eta + 1)\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial z^2} + \eta \frac{\partial^2 v}{\partial x \partial z} - (\eta + 1)\frac{\partial p}{\partial x} = 0, \quad (14b)$$

$$\frac{\partial^2 v}{\partial x^2} + (\eta + 1)\frac{\partial^2 v}{\partial z^2} + \eta \frac{\partial^2 u}{\partial x \partial z} - (\eta + 1)\frac{\partial p}{\partial z} = 0. \quad (14c)$$

where u and v are the deformations of porous medium along the x and z directions, p is the pore fluid pressure, Q is a fluid source or sink term, η and β are nondimensional parameters which are functions of the real material parameters. An illustration of the problem is shown in Fig.10. An oscillating point source Q located at (x_0, z_0) in the rectangular domain $[0, a] \times [0, b]$ is given by:

$$Q(x, z, t) = \delta(x - x_0)\delta(z - z_0)sin(\omega t) \quad (15)$$

where ω is the oscillation frequency. The parameters are set as follows: $a = b = 1$, $t \in [0, 2\pi]$, $\beta = 2$, $\eta = 1.5$, $(x_0, z_0) = (0.25, 0.25)$ and $\omega = 1$.

Table 3 shows the L_2 *error*s achieved by different network architectures. Compared with the SIREN architecture (# subnets = 1), MS-SIREN does not bring any benefit in this case as the point source in this problem contains only one frequency component, and thus there are not multiple frequency components in the analytical solution. Fig.11 compares the convergence process with different activation functions. The periodic activation function *Sine* exhibits superior performance over *ReLU* and *Tanh*. In addition to Gaussian distribution, we also use Cauchy distribution and Laplacian distribution to approximate the Dirac delta function, and obtain L_2 *error* being 0.051 and 0.025, respectively.

Network Architectures			L_2 *error*			
# subnets	# layers	# neurons	u	v	p	mean
1	5	256	0.022	0.021	0.026	0.023
1	7	256	0.011	0.016	0.026	0.018
2	5	128	0.010	0.011	0.026	**0.016**
2	7	128	0.012	0.013	0.026	0.017
4	5	64	0.012	0.013	0.026	0.017
4	7	64	0.018	0.014	0.026	0.019

Table 3: Barry and Mercer's source problem: L_2 *error*s got by different network architectures.

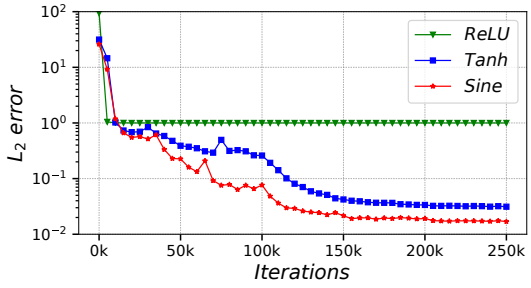


Figure 11: Barry and Mercer's source problem: Convergence speed of the mean L_2 *error*s when different activation functions are used.

5 Conclusions

PDEs with a point source pose a great challenge to conventional PINNs method due to the singularity problem, and all the methods trying to tackle this problem so far can only work under certain constraints. We propose a universal approach based on the PINNs method that can solve the PDEs with a point source without any specific assumptions. The novelty of our approach lies in three aspects, approximating the Dirac delta function with a symmetric unimodal continuous probability density function, balancing the training loss terms of different areas with a lower bound constrained uncertainty weighting algorithm, and using multi-scale DNN with *Sine* activation function to build the network architecture. Three representative physical problems are used to verify the effectiveness of our method, and these experiments show that our method outperforms existing deep learning-based methods in accuracy, efficiency and versatility.

Acknowledgments

This work was supported by National Key R&D Program of China under Grant No. 2021ZD0110400.

References

[Bekele, 2020] Yared W Bekele. Physics-informed deep learning for flow and deformation in poroelastic media. *arXiv preprint arXiv:2010.15426*, 2020.

[Bischof and Kraus, 2021] Rafael Bischof and Michael Kraus. Multi-objective loss balancing for physics-informed deep learning. *arXiv preprint arXiv:2110.09813*, 2021.

[Cai *et al.*, 2021] Shengze Cai, Zhicheng Wang, Sifan Wang, Paris Perdikaris, and George Em Karniadakis. Physics-informed neural networks for heat transfer problems. *Journal of Heat Transfer*, 143(6):060801, 2021.

[Gedney, 2011] Stephen D Gedney. Introduction to the finite-difference time-domain (fdtd) method for electromagnetics. *Synthesis Lectures on Computational Electromagnetics*, 6(1):1–250, 2011.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[Hennigh *et al.*, 2021] Oliver Hennigh, Susheela Narasimhan, Mohammad Amin Nabian, Akshay Subramaniam, Kaustubh Tangsali, Zhiwei Fang, Max Rietmann, Wonmin Byeon, and Sanjay Choudhry. Nvidia simnet™: An ai-accelerated multi-physics simulation framework. In *International Conference on Computational Science*, pages 447–461. Springer, 2021.

[Heydari *et al.*, 2019] A Ali Heydari, Craig A Thompson, and Asif Mehmood. Softadapt: Techniques for adaptive loss weighting of neural networks with multi-part loss functions. *arXiv preprint arXiv:1912.12355*, 2019.

[Jin *et al.*, 2021] Xiaowei Jin, Shengze Cai, Hui Li, and George Em Karniadakis. Nsfnets (navier-stokes flow nets): Physics-informed neural networks for the incompressible navier-stokes equations. *Journal of Computational Physics*, 426:109951, 2021.

[Kendall *et al.*, 2018] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7482–7491, 2018.

[Li *et al.*, 2020] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

[Liu *et al.*, 2020] Ziqi Liu, Wei Cai, and Zhi-Qin John Xu. Multi-scale deep neural network (mscalednn) for solving poisson-boltzmann equation in complex domains. *arXiv preprint arXiv:2007.11207*, 2020.

[Lu *et al.*, 2019] Lu Lu, Pengzhan Jin, and George Em Karniadakis. Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv preprint arXiv:1910.03193*, 2019.

[McClenny and Braga-Neto, 2020] Levi McClenny and Ulisses Braga-Neto. Self-adaptive physics-informed neural networks using a soft attention mechanism. *arXiv preprint arXiv:2009.04544*, 2020.

[Moseley *et al.*, 2020] Ben Moseley, Andrew Markham, and Tarje Nissen-Meyer. Solving the wave equation with physics-informed deep learning. *arXiv preprint arXiv:2006.11894*, 2020.

[Pinkus, 1999] Allan Pinkus. Approximation theory of the mlp model. *Acta Numerica 1999: Volume 8*, 8:143–195, 1999.

[Raissi *et al.*, 2019] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[Sirignano and Spiliopoulos, 2018] Justin Sirignano and Konstantinos Spiliopoulos. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of computational physics*, 375:1339–1364, 2018.

[Sitzmann *et al.*, 2020] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33, 2020.

[Sullivan, 2013] Dennis M Sullivan. *Electromagnetic simulation using the FDTD method*. John Wiley & Sons, 2013.

[Wang *et al.*, 2020] Sifan Wang, Xinling Yu, and Paris Perdikaris. When and why pinns fail to train: A neural tangent kernel perspective. *arXiv preprint arXiv:2007.14527*, 2020.

[Wang *et al.*, 2021] Sifan Wang, Yujun Teng, and Paris Perdikaris. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43(5):A3055–A3081, 2021.

[Weinan and Yu, 2018] E Weinan and Bing Yu. The deep ritz method: a deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6(1):1–12, 2018.

[Xu *et al.*, 2019] Zhi-Qin John Xu, Yaoyu Zhang, Tao Luo, Yanyang Xiao, and Zheng Ma. Frequency principle: Fourier analysis sheds light on deep neural networks. *arXiv preprint arXiv:1901.06523*, 2019.

[Zhang *et al.*, 2021] Pan Zhang, Yanyan Hu, Yuchen Jin, Shaogui Deng, Xuqing Wu, and Jiefu Chen. A maxwell's equations based deep learning method for time domain electromagnetic simulations. *IEEE Journal on Multiscale and Multiphysics Computational Techniques*, 6:35–40, 2021.