ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) | 978-1-7281-6327-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICASSP49357.2023.10095076

SD-PINN: PHYSICS INFORMED NEURAL NETWORKS FOR SPATIALLY DEPENDENT PDES

Ruixian Liu, Peter Gerstoft

University of California, San Diego

ABSTRACT

The physics-informed neural network (PINN) is able to identify partial differential equation (PDE) coefficients which are constant across the space directly from physical measurements. In this paper, we propose a modification of PINN, named as SD-PINN, which can recover the coefficients in spatially-dependent PDEs using only one neural network without the requirement of domain-specific physical knowledge. The network structure is a simple fully connected neural network, and multiple physical information like the time-invariance and spatial-smoothness of the PDE coefficients is incorporated as loss functions. The method is robust to noise due to introduced physical constraints, which is verified by experiments.

***Index Terms*—** physics informed neural networks (PINNs), spatially-dependent PDEs, deep learning, data-driven.

1. INTRODUCTION

Many natural phenomena are described by parial differential equations (PDEs) which consist of multiple PDE terms. A PDE governing the dynamical behaviors of field U can be described by

$$\mathsf{N}[U] = a_1 U_x + a_2 U_y + a_3 U_t + a_4 U_{tt} + \dots \tag{1}$$

where the partial derivatives $U_x, U_y, U_t, ...$ are the PDE terms and the $a_1, a_2, ...$ are PDE coefficients. The PDE coefficients can be spatially-dependent, which indicates an inhomogeneous medium for the dynamics. In this work, we use the attenuating wave equation as the example:

$$\mathsf{N}[U] = U_{tt} + \alpha U_t - c^2 U_{xx} \,, \tag{2}$$

where U_t, U_{tt} are the 1st and 2nd-order temporal derivatives of U and U_{xx} the 2nd-order spatial derivative of U. The PDE coefficient $\alpha \geq 0$ is the wave attenuation factor and $c^2 > 0$ is the square of the phase speed c.

Recently, the increasing computer resources have encouraged multiple efforts on data-drive PDE recovery from physical measurements [1–14]. Among them, the physics informed neural network (PINN) [1–3] has drawn significant attention from researchers thanks to its robustness against measurement noise.

Given coordinates (x, t) and the measurements $U(x, t)$ on these spatio-temporal points, the PINN can learn the function that maps the coordinate to the measurement by an L-layer fully connected neural network (FCN) as shown in Fig. 1. The PINN assumes we know which PDE is governing the physical system from prior knowledge and only want to find the coefficients within the PDE, e.g., if we know the PDE is the wave equation as (2), the PINN can be employed to recover the PDE coefficients, i.e., the attenuation α and squared phase speed c^2. We denote the weights and bias in all layers as the neural network parameter θ.

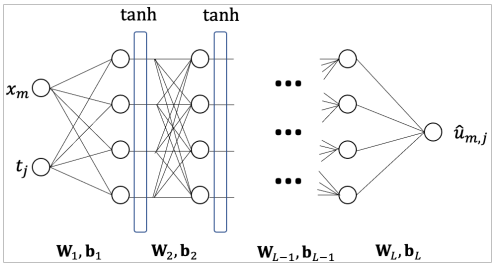

Fig. 1: The structure of the neural network used in PINN and SD-PINN.

In reality, the PDE governing the dynamics within a field can be spatially-dependent, e.g., the phase speed c can vary within an area of the medium (like water) due to the spatial variations of the temperature or the density. The PINN assumes that the PDE coefficients are identical across the whole region of interest (ROI) and thus is unable to identify the PDEs in these cases. There are only a few efforts about using PINN to recover spatially-dependent PDE coefficients [15, 16], in which they construct two neural networks to recover the PDE coefficients in addition to approximating the solution of the PDE. To be specific, both of [15, 16] use one neural network to approximate the dynamical behaviors in the field and another network to estimate the material parameters. The two networks are then joined via domain-specific physical knowledge such as the stress-strain relationship [15] or Maxwell's equations [16].

In this paper, we propose the spatially dependent physics informed neural network (SD-PINN) that is capable of recovering spatially dependent PDEs. In contrast to [15, 16], we can recover the spatially-dependent coefficients at the same time of approximating the PDE solution using only one neural network, and do not require domain-specific physical knowledge. Compared to the previous spatially-dependent PDE recovery work which is based on least squares regression [17], the major advantage of the SD-PINN is its robustness against the noise in the input signals.

2. THEORY

Like for the original PINN, we assume the kind of PDE known and the task is to recover the coefficients for each term in the assumed PDE at all locations within the ROI. In addition, the true PDE coefficients at the spatial boundaries of the ROI are given in this work, and we also know the sign information (non-positive or non-negative) of each coefficients to be recovered. Given that the kind of PDE is known, the sign information is not a strong additional assumption because it is fixed by the physical background of the PDE. For example, in (2) the coefficient $-c^2$ for U_{xx} must be non-positive since c is a real number for the phase speed of the wave, and the α must

be non-negative for a system without input energy from the external.

2.1. Formulation

We fix one term of the PDE as the left hand side (LHS) with its coefficient arbitrarily set to one at every location, and recover the coefficients for all other terms placed in the right hand side (RHS) for all locations. In this part we use the wave equation as the example, and for other PDEs the method works in the same way. We focus on time-invariant homogeneous PDE here, e.g., $\mathsf{N}[U] = 0$ in (2) which indicates that there is no source in the ROI and the α, c^2 do not change with time.

Since $\mathsf{N}[U] = 0$, we can write the wave equation (2) as

$$U_{tt} = -\alpha U_t + c^2 U_{xx} \ . \tag{3}$$

For M spatial locations within the ROI, if the PDE is spatially dependent, then (3) becomes

$$U_{tt} = -\alpha_m U_t + c_m^2 U_{xx}, \ m = 1, ..., M \ . \tag{4}$$

Generally, if there are K PDE terms $d_k, k = 1, ..., K$ in the RHS with coefficients $\lambda_k, k = 1, ..., K$, for location m at time step j, the kth term in the RHS with its coefficient is

$$r_{mk}^j = \lambda_{mk} d_{mk}^j \tag{5}$$

where d_{mk}^j is the kth PDE term evaluated at location m and time j and the coefficient λ_{mk} is identical for all the time j because the coefficients are time-invariant. For all the M locations, the RHS can be denoted by

$$\begin{bmatrix} r_{11}^j & \cdots & r_{1K}^j \\ r_{21}^j & \cdots & r_{2K}^j \\ \vdots & \cdots & \vdots \\ r_{M1}^j & \cdots & r_{MK}^j \end{bmatrix} = \begin{bmatrix} \lambda_{11} & \cdots & \lambda_{1K} \\ \lambda_{21} & \cdots & \lambda_{2K} \\ \vdots & \cdots & \vdots \\ \lambda_{M1} & \cdots & \lambda_{MK} \end{bmatrix} \circ \begin{bmatrix} d_{11}^j & \cdots & d_{1K}^j \\ d_{21}^j & \cdots & d_{2K}^j \\ \vdots & \cdots & \vdots \\ d_{M1}^j & \cdots & d_{MK}^j \end{bmatrix} \tag{6}$$

where the $\circ$ denotes element-wise product. For the wave equation (4), $K = 2$, and d_{m1}^j, d_{m2}^j are for U_t, U_{xx} respectively. The λ_{m1} and λ_{m2} denote $-\alpha_m$ and c_m^2. The (6) indicates one difference between this work and the conventional PINN [1–3], in which there are only K unknown PDE coefficients $\{\lambda_1, \ldots, \lambda_K\}$ to be recovered since the PDE is assumed to be spatially-independent and thus the coefficients are identical for all locations.

Let the LHS at the same location m and time step j denoted by ℓ_m^j, then the PDE is written as

$$\ell_m^j = \sum_{k=1}^{K} r_{mk}^j \ . \tag{7}$$

For the wave equation (4), the ℓ_m^j is the U_{tt} evaluated at time j and location m.

2.2. Loss functions

During training the SD-PINN we are minimizing the overall loss $loss$ in Eq. (8):

$$loss = loss_{\mathrm{u}} + w_{\mathrm{f}} \times loss_{\mathrm{f}} + w_{\mathrm{sm}} \times loss_{\mathrm{sm}} + w_{\mathrm{b}} \times loss_{\mathrm{b}} + w_{\mathrm{si}} \times loss_{\mathrm{si}} \tag{8}$$

which is a linear combination of 5 losses $\{loss_{\mathrm{u}}, loss_{\mathrm{f}}, loss_{\mathrm{sm}}, loss_{\mathrm{b}}, loss_{\mathrm{si}}\}$ with $w_{\mathrm{f}}, w_{\mathrm{sm}}, w_{\mathrm{b}}$ and w_{si} their weights. They can be grouped into 3 categories: (i) the data fitting loss $loss_{\mathrm{u}}$ is a function of only the neural network parameters θ (weights and bias); (ii) the functional loss $loss_{\mathrm{f}}$ is a function of both θ and the PDE coefficients λ; (iii) the smoothness, boundary and sign losses $\{loss_{\mathrm{sm}}, loss_{\mathrm{b}}, loss_{\mathrm{si}}\}$ are the functions of only the PDE coefficients λ. They are detailed as follows.

The SD-PINN first learns the dynamics by maximizing the similarity between the composite function described by the neural network and the true function mapping the time steps t and spatial locations x to corresponding physical measurements u in the training dataset. Assuming there are M locations and T time indices within the training dataset, the method is to minimize a loss function

$$loss_{\mathrm{u}} = \sum_{m=1}^{M} \sum_{j=1}^{T} (Net_\theta(x_m, t_j) - u_{m,j})^2 \tag{9}$$

where the network parametrized by θ (including weights $\mathbf{W}_l$ and bias $\mathbf{b}_l$ in all L layers) shown in Fig. 1 is denoted by Net_θ, which is a non-linear function of (x, t):

$$Net_\theta(x,t) = \mathbf{W}_L^{\mathrm{T}}(\ldots(\mathbf{W}_2^{\mathrm{T}} \tanh(\mathbf{W}_1^{\mathrm{T}} \begin{bmatrix} x \\ t \end{bmatrix} + \mathbf{b}_1) + \mathbf{b}_2)\ldots) + \mathbf{b}_L$$

where the $\tanh$ is applied in an element-wise way.

After the estimated measurement $\widehat{u}_{m,j} = Net_\theta(x_m, t_j)$ is calculated, the derivatives involved by the PDE are computed by automatic differentiation [18]. The estimated $\widehat{u}_x$ at (x_m, t_j) is acquired by computing the derivative $\frac{\partial Net_\theta(x,t)}{\partial x}\Big|_{x=x_m, t=t_j}$, which is a function of (x, t) parametrized by θ (updated in the previous epochs during training) evaluated at (x_m, t_j). For the 2nd order derivative like $\widehat{u}_{xx}$ which is $\widehat{d}_{m2}^j$, the result is acquired by $\frac{\partial}{\partial x}(\frac{\partial Net_\theta(x,t)}{\partial x})|_{x=x_m, t=t_j}$ where the $\frac{\partial}{\partial x}(\frac{\partial Net_\theta(x,t)}{\partial x})$ is also a function parametrized by θ. The automatic differentiation is also used to compute $\widehat{u}_t$ and the LHS $\widehat{\ell}_m^j$, e.g., the estimated $\widehat{u}_{tt}$ in (3). After $\widehat{d}_{mk}^j$ and $\widehat{\ell}_m^j$ are estimated by automatic differentiation, inspired by [1–3], the $\widehat{\lambda}_{mk}$ could be estimated by minimizing

$$\sum_{j=1}^{T} \sum_{m=1}^{M} ((\sum_{k=1}^{K} \widehat{r}_{mk}^j) - \widehat{\ell}_m^j)^2 = \sum_{j=1}^{T} \sum_{m=1}^{M} ((\sum_{k=1}^{K} \widehat{\lambda}_{mk} \widehat{d}_{mk}^j) - \widehat{\ell}_m^j)^2 \tag{10}$$

because of (5) and (7). But we use a modified version of (10) as detailed in the following.

To increase the robustness against noise, we add "virtual measurements" evenly located between neighboring true measurements in both spatial and temporal aspects. Let us evenly insert P "virtual measurements" into two neighboring spacial locations and Q "virtual measurements" into two neighboring time points, then we have $(M-1)(P+1)+1$ spatial coefficients for each PDE term, and every such spatial coefficient is invariant across all the $(T-1)(Q+1)+1$ time steps. These "virtual measurements" are computed from the forward process of the SD-PINN, e.g., $\widehat{u}_{(m+\frac{p}{P+1})(j+\frac{q}{Q+1})} = Net_\theta(x,t)|_{x=x_m+\frac{p}{P+1}\Delta x, t=t_j+\frac{q}{Q+1}\Delta t}$ for $p \in \{1, ..., P\}$ and $q \in \{1, ..., Q\}$ where Δx and Δt are the step size between neighboring true measurements along space and time. In this work, we use $P = 1$ for simplicity, which indicates that there is only one "virtual measurement" inserted at the mid-point of two neighboring spatial locations. We define the functional loss to be

$$loss_{\mathrm{f}} = \sum_{j \in I_{\mathrm{t}}} \sum_{m \in I_{\mathrm{m}}} ((\sum_{k=1}^{K} \widehat{\lambda}_{mk} \widehat{d}_{mk}^j) - \widehat{\ell}_m^j)^2 \tag{11}$$

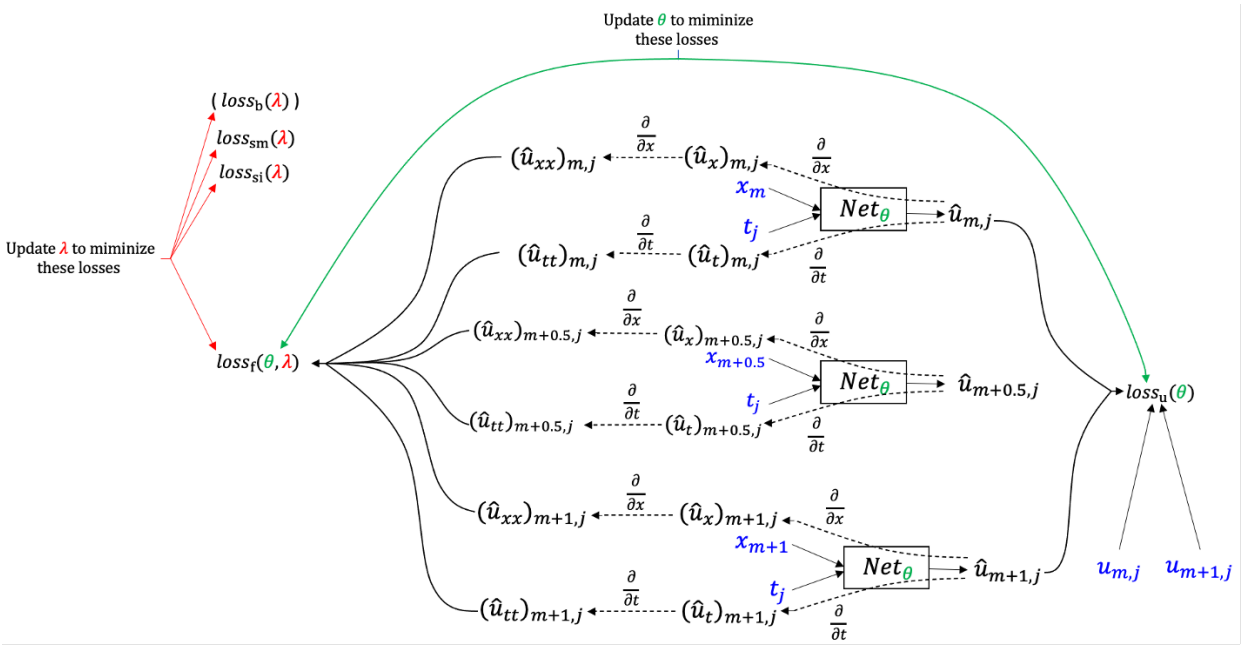

Fig. 2: A demonstration for the losses of SD-PINN, for each loss, only a small part of the summation involving (x_m, t_j), $(x_{m+0.5}, t_j)$ and (x_{m+1}, t_j) is shown. The input data are shown in blue. The spatial index m and time index j correspond to a true measurement instead of an inserted "virtual measurement" and the $\widehat{u}_{m+0.5,j}$ does not affect $loss_{\mathrm{u}}$ since it is virtual. The dashed lines are for automatic differentiation, which is computed by evaluating functions parametrized by θ at given coordinates, e.g., (x_m, t_j). The $loss_{\mathrm{b}}$ is included only if $m = 1$ or $m + 1 = M$ (i.e., at boundaries). The loss functions are the functions of θ and/or λ, and the neural network is parametrized by θ. The three Net_θ blocks denote just one neural network parametrized by θ and structured as Fig.1, with the difference being the inputs and outputs.

where the set $I_t = \{1, 1 + \frac{1}{Q+1}, \ldots, T - 1 + \frac{Q}{Q+1}, T\}$ and set $I_{\mathrm{m}} = \{1, 1.5, 2, \ldots, M - 1, M - 0.5, M\}$. Although in (11) the m uses all values in the set I_{m} which are stepped by 0.5, only the λ_{mk} whose m is an integer is for the PDE coefficient at the true location.

Updating (10) to (11) indicates that the recovered PDE at any location m can also describe the dynamics at Q inserted time steps between every pair of neighboring time steps of the true T observations. Meanwhile, all the recovered $\widehat{\lambda}_{mk}$ with non-integer m are used for smoothing. The smoothness penalty is

$$loss_{\mathrm{sm}} = \sum_{k=1}^{K} \sum_{m=1}^{M-1} (\widehat{\lambda}_{mk} + \widehat{\lambda}_{(m+1)k} - 2\widehat{\lambda}_{(m+0.5)k})^2 . \quad (12)$$

Minimizing (12) encourages a smoother transition of the recovered PDE coefficients between two neighboring sensors.

To accelerate the training, we assume the PDE coefficients on the spatial boundaries of the ROI known. Thus we define the boundary loss $loss_{\mathrm{b}}$ according to the difference between the estimated and true coefficients on the boundaries:

$$loss_{\mathrm{b}} = \sum_{k=1}^{K} ((\widehat{\lambda}_{1k} - \lambda_{1k})^2 + (\widehat{\lambda}_{Mk} - \lambda_{Mk})^2) . \quad (13)$$

In addition, the assumed sign for the unknown PDE coefficient is fixed. For example, in the wave equation (4), the c_m^2 must be non-negative for $\forall m$ to be physically meaningful as it is the squared phase speed. Thus a loss function $loss_{\mathrm{si}}$ penalizing the $\widehat{\lambda}_{mk}$ whose value violates the desired sign can be designed. We define the $loss_{\mathrm{si}}$ as

$$loss_{\mathrm{si}} = \sum_{m} \sum_{k} \mathrm{ReLU}(-\mathrm{sign}(\lambda_{mk})\widehat{\lambda}_{mk}) \quad (14)$$

where $\mathrm{sign}(x) = 1, \forall x > 0$ and $-1, \forall x < 0$. Note that the $\mathrm{sign}(\lambda_{mk})$ is determined by the assumed PDE form and thus is irrelevant to the recovered value $\widehat{\lambda}_{mk}$ and independent of m. For example, in (4), $\mathrm{sign}(\lambda_{mk}) = -1$ for $k = 1$ since λ_{m1} denotes $-\alpha_m \leq 0$ (if the λ_{mk} can be zero like here, we only care about the possible sign when it is non-zero). ReLU is the Rectified Linear Unit defined as $\mathrm{ReLU}(x) = x$ for $x > 0$ and 0 otherwise. By minimizing (14), the $\widehat{\lambda}_{mk}$ is encouraged to have its assumed sign.

A demonstration of how the inputs, the nueral network parameter θ and the PDE coefficients λ are related by the losses discussed above is shown in Fig. 2.

2.3. Robustness to noise

The advantage of the SD-PINN over the PDE recovery methods based on finite difference (FD) [19] is its robustness against noise. For SD-PINN the LHS l and the PDE terms in the RHS d in (6) are both computed by automatic differentiation from the neural network estimation $\widehat{u} = Net_\theta(x, t)$, which is supposed to be less noisy (than the noisy measurements u) because it is also regularized by $loss_{\mathrm{f}}$, $loss_{\mathrm{sm}}$, $loss_{\mathrm{si}}$ in addition to the data fitting loss $loss_{\mathrm{u}}$. Both $loss_{\mathrm{f}}$ and $loss_{\mathrm{si}}$ help suppressing the noise by introducing physical constraints from the assumed PDE knowledge, and $loss_{\mathrm{sm}}$ help suppressing the noise by introducing smoothness constraint between the recovered coefficients at neighboring locations.

We use two least squares regression (LSQ) based methods as the baselines for comparison, where the PDE terms are directly computed by FD from noisy measurements and then the LSQ is implemented to find the PDE coefficients. For example, if the aim is to recover the $-\alpha_m$ and c_m^2 in (4), the $T \times 1$ vectors $\mathbf{u}_t$, $\mathbf{u}_{tt}$ and $\mathbf{u}_{xx}$ are first computed by FD using the data around the m-th location

in the noisy measurements, and then $[-\alpha_m \; c_m^2]^{\mathrm{T}} = [\mathbf{u}_t \;\; \mathbf{u}_{xx}]^{\dagger}\mathbf{u}_{tt}$ where the † denotes pseudo-inverse (which is the least squares regression to solve $\mathbf{u}_{tt} = [\mathbf{u}_t \;\; \mathbf{u}_{xx}][-\alpha_m \; c_m^2]^{\mathrm{T}}$).

If a naive FD [19] is used to compute the PDE terms, we name the method as FD-LSQ. The naive FD can be robustified against noise by adding a total variation (TV) regularization [20], and we name the method using this TV-regularized FD as TVR-FD-LSQ.

3. EXPERIMENTS

We conduct two experiments, one for recovering one PDE coefficient (phase speed) from measurements with large noise, and the other for recovering two PDE coefficients (phase speed and the more implicit attenuation factor) from observations with noise. The datasets are generated by finite difference modeling [21]. In both experiments, the SD-PINN is trained by Adam [22].

3.1. Recovering phase speeds from noisy measurements

We assume no attenuation in the wave field for this experiment and thus $\alpha_m = 0$ is set everywhere in (4). So there is only $K = 1$ PDE coefficient to recover, which is c_m^2 represented by λ_{m1}.

The network has 9 layers, and $w_{\mathrm{f}} = w_{\mathrm{sm}} = w_{\mathrm{b}} = w_{\mathrm{si}} = 10$ in the loss function (8). We set $Q = 3$. The wavefield is shown in Fig. 3a in which the ROI is between $[3, 23]\Delta x$ in space (thus $M = 21$) and $[5, 195]\Delta t$ in time as indicated by the red lines. The governing PDE for the field is wave equation (4) without the U_t term and the spatially-dependent c^2 is indicated by the "True" line in Fig. 4a. The measurements of the field are polluted by the zero-mean Gaussian noise with the signal-to-noise ratio $\mathrm{SNR} = 20$ dB.

The recovered $\widehat{\lambda}_{m1}$, i.e., the c^2 with respect to locations is shown in Fig. 4a. The phase speeds recovered by least squares regression (LSQ) are also included for comparison, in which the U_{tt} and U_{xx} are calculated numerically by naive finite difference or TVR finite difference at first, and then the coefficient for U_{xx} is computed by least squares regression. As shown in the row for Sec. 3.1 in Table 1, the spatially-dependent c^2 recovered by SD-PINN has a much smaller MSE (mean squared error) with respect to the true values.

		SD-PINN	FD-LSQ	TVR-FD-LSQ
Sec. 3.1	c^2	1.25×10^{-2}	7.87×10^{-1}	1.49
Sec. 3.2	$-\alpha$	1.80×10^{-5}	5.31×10^{-4}	1.84×10^{-4}
	c^2	5.41×10^{-3}	2.30×10^{-1}	2.52×10^{-1}

Table 1: For Sec. 3.1 and 3.2, the MSE between the true PDE coefficients and the recovered ones from various methods.

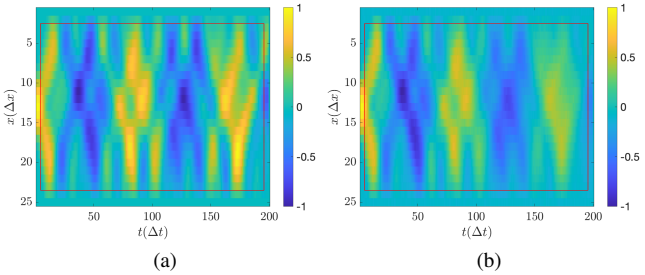

Fig. 3: The noisy measurements of (a) the waves without attenuation; (b) the waves with attenuation. The ROI covering $3 \leq x \leq 23$ and $5 \leq t \leq 195$ is indicated by red lines.

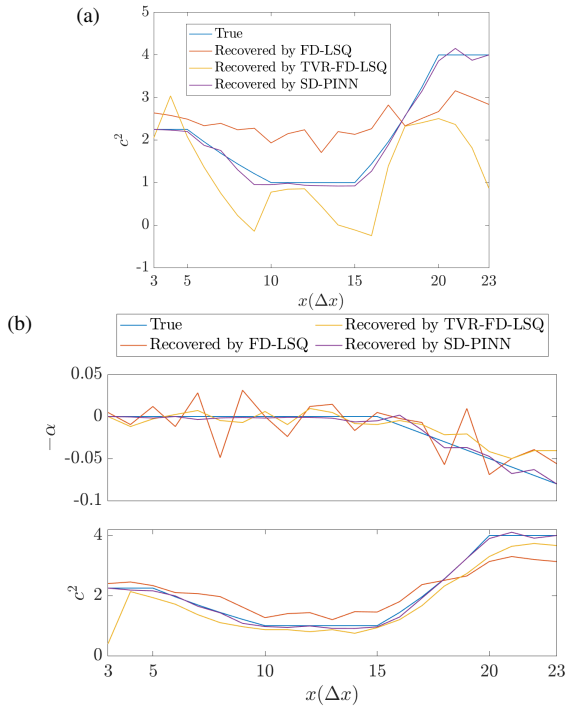

Fig. 4: (a) For the noisy data shown in Fig. 3a, the true c^2 and the recovered values from SD-PINN and the least squares regression (LSQ). (b) For the noisy data shown in Fig. 3b, the true $-\alpha$, c^2 and the recovered values from SD-PINN and the LSQ.

3.2. Recovering attenuation and phase speeds from noisy data

With attenuation, wave equation (4) is used. The attenuation α is harder to recover since its absolute value can be much smaller and the wave propagation is more obvious to observe than attenuation.

To recover (4), we still keep the U_{tt} in the LHS and set the number of PDE terms in RHS $K = 2$. The $k = 1$ is for U_t, and $k = 2$ is for U_{xx}. Thus, the λ_{m1} is for $-\alpha$ and λ_{m2} for c^2 at location m.

We use a similar dataset as in Sec. 3.1 for this experiment, see Fig. 3b. The initial state and the phase speeds distribution are the same as before, and the only difference is that the attenuation is nonzero for a part of ROI, as shown by the "True" line in Fig. 4b. The measurements are polluted by the zero-mean Gaussian noise with $\mathrm{SNR} = 30$ dB.

The network has 5 layers here, and $w_{\mathrm{f}} = w_{\mathrm{sm}} = w_{\mathrm{b}} = w_{\mathrm{si}} = 10$ in the loss function (8). We set $Q = 1$. The SD-PINN works well as the recovered α and c^2 are closer to the ground truth compared to the baseline methods, as shown in Fig. 4b. The MSEs between the true PDE coefficients and the recovered ones from various methods are recorded in the rows for Sec. 3.2 in Table 1, in which the coefficients recoverd by SD-PINN have much smaller MSEs.

4. CONCLUSION

In this work, we proposed a neural network termed as SD-PINN that can recover spatially dependent PDE coefficients using only one network without the domain knowledge pertinent to a specific situation. The network structure is a simple FCN and the physical information for the PDE is encoded into the loss functions. The SD-PINN is robust to noise, which is demonstrated by various experiments.

5. REFERENCES

[1] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *J. Comput. Phys.*, vol. 378, pp. 686–707, 2019.

[2] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations," *arXiv preprint arXiv:1711.10561*, 2017.

[3] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics informed deep learning (part ii): data-driven discovery of nonlinear partial differential equations," *arXiv preprint arXiv:1711.10566*, 2017.

[4] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems," *Proc. Natl. Acad. Sci.*, vol. 113, no. 15, pp. 3932–3937, 2016.

[5] S. H. Rudy, S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Data-driven discovery of partial differential equations," *Sci. Adv.*, vol. 3, no. 4, pp. e1602614, 2017.

[6] H. Schaeffer, G. Tran, and R. Ward, "Extracting sparse high-dimensional dynamics from limited data," *SIAM J. Appl. Math.*, vol. 78, no. 6, pp. 3279–3295, 2018.

[7] Z. Long, Y. Lu, and B. Dong, "Pde-net 2.0: Learning pdes from data with a numeric-symbolic hybrid deep network," *J. Comput. Phys*, vol. 399, pp. 108925, 2019.

[8] S. L. Brunton and J. N. Kutz, *Data-driven science and engineering: Machine learning, dynamical systems, and control*, Cambridge University Press, 2019.

[9] Camps-Valls G., Martino L., Svendsen D. H., Campos-Taberner M., Muñoz-Marí J., Laparra V., Luengo D., and García-Haro F. J., "Physics-aware gaussian processes in remote sensing," *Appl. Soft Comput.*, vol. 68, pp. 69–82, 2018.

[10] R. Liu, M. J. Bianco, and P. Gerstoft, "Wave equation extraction from a video using sparse modeling," in *Proc.53th Asilomar Conf. on Circuits, Systems and Computers.* IEEE, 2019, pp. 2160–2165.

[11] S. Zhang and G. Lin, "Robust data-driven discovery of governing physical laws with error bars," *Proc. Math. Phys. Eng. Sci.*, vol. 474, no. 2217, pp. 20180305, 2018.

[12] R. Liu, M. Bianco, and P. Gerstoft, "Automated partial differential equation identification," *J. Acoust. Soc. Am.*, vol. 150, no. 4, pp. 2364–2374, 2021.

[13] H. Xu, H. Chang, and D. Zhang, "DL-PDE: Deep-learning based data-driven discovery of partial differential equations from discrete and noisy data," *Commun. Comput. Phys.*, vol. 29, pp. 698–728, 2021.

[14] P. Pilar and N. Wahlström, "Physics-informed neural networks with unknown measurement noise," *arXiv preprint arXiv:2211.15498*, 2022.

[15] E. Zhang, M. Yin, and G. E. Karniadakis, "Physics-informed neural networks for nonhomogeneous material identification in elasticity imaging," *arXiv preprint arXiv:2009.04525*, 2020.

[16] Y. Zhang, H. Fu, Y. Qin, K. Wang, and J. Ma, "Physics-informed deep neural network for inhomogeneous magnetized plasma parameter inversion," *IEEE Antennas Wirel. Propag. Lett.*, vol. 21, no. 4, pp. 828–832, 2022.

[17] R. Liu, M. Bianco, P. Gerstoft, and B. D. Rao, "Data-driven spatially dependent pde identification," in *Proc. 2022 IEEE Int. Conf. Acoust. Speech Signal Process., ICASSP*. IEEE, 2022, pp. 3383–3387.

[18] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, "Automatic differentiation in machine learning: a survey," *J. Mach. Learn. Res.*, vol. 18, pp. 1–43, 2018.

[19] W. F. Ames, *Numerical methods for partial differential equations*, Academic press, 2014.

[20] R. Chartrand, "Numerical differentiation of noisy, nonsmooth data," *Int. Sch. Res. Notices*, vol. 2011, 2011.

[21] G. D. Smith, *Numerical solution of partial differential equations: finite difference methods*, Oxford university press, 1985.

[22] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.