Engineering with Computers
https://doi.org/10.1007/s00366-024-01957-5

ORIGINAL ARTICLE

Physics informed neural networks for an inverse problem in peridynamic models

Fabio V. Difonzo[1] · Luciano Lopez[2] · Sabrina F. Pellegrino[3]

Received: 18 December 2023 / Accepted: 13 February 2024

Abstract
Deep learning is a powerful tool for solving data driven differential problems and has come out to have successful applications in solving direct and inverse problems described by PDEs, even in presence of integral terms. In this paper, we propose to apply radial basis functions (RBFs) as activation functions in suitably designed Physics Informed Neural Networks (PINNs) to solve the inverse problem of computing the perydinamic kernel in the nonlocal formulation of classical wave equation, resulting in what we call RBF-iPINN. We show that the selection of an RBF is necessary to achieve meaningful solutions, that agree with the physical expectations carried by the data. We support our results with numerical examples and experiments, comparing the solution obtained with the proposed RBF-iPINN to the exact solutions.

Keywords Physics Informed Neural Network · Peridynamic Theory · Radial Basis Functions · Inverse Problems

Mathematics Subject Classification 34A36 · 15B99

1 Introduction to the peridynamic inverse problem

We consider the following PDE in peridynamic formulation:

$$\partial_{tt}\theta(x,t) = \int_{\mathbb{R}} C(|x-y|)[\theta(x,t)-\theta(y,t)]\ \mathrm{d}y, \tag{1.1}$$

where $C : \mathbb{R} \to \mathbb{R}$, representing the so-called kernel function, is a nonnegative even function.

✉ Fabio V. Difonzo
fabiovito.difonzo@cnr.it

Luciano Lopez
luciano.lopez@uniba.it

Sabrina F. Pellegrino
sabrinafrancesca.pellegrino@poliba.it

1 Istituto per le Applicazioni del Calcolo "Mauro Picone", Consiglio Nazionale delle Ricerche, Via G. Amendola 122/I, 70126 Bari, Italy

2 Dipartimento di Matematica, Università degli Studi di Bari Aldo Moro, Via E. Orabona 4, 70125 Bari, Italy

3 Dipartimento di Ingegneria Elettrica e dell'Informazione, Politecnico di Bari, Via E. Orabona 4, 70125 Bari, Italy

In the one-dimensional case, the model describes the dynamic response of an infinite bar composed of a linear microelastic material.

The main important aspect of such constitutive model is that it takes into account long-range interactions and their effects. The equation of motion (1.1) was proposed by Silling in 2000 in [29] in the framework of continuum mechanics theory with the name of linear peridynamics. This is an integral-type nonlocal model involving only the displacement field and not its gradient. This leads to a theory able to incorporate cracks, fractures and other kind of singularities.

The general initial-value problem associated with (1.1) is well-posed (see [13]) and due to the presence of long-range forces, the solution shows a dispersive behavior. The length-scale of the long-range interactions is parameterized by a positive scalar value $\delta > 0$ called horizon, which represents the maximum interaction distance between two material particles. In the more general setting, this parameter is intrinsically incorporated into the kernel function C, that is meant to weigh the nonlocal interactions.

If the kernel function C, also known as micromodulus function, is a suitable generalized function, in the limit of short-range forces, or equivalently taking the limit as $\delta \to 0$, the linear peridynamic model (1.1) reduces to the wave equation $\partial_{tt}\theta(x,t) - \partial_{xx}\theta(x,t) = 0$, (see [25] and references therein). As a consequence, the length-scale parameter δ can be viewed as a measure of the degree of nonlocality of the model.

In order to maintain the consistency with Newton's third law, the micromodulus function must be even:

$$C(\xi) = C(-\xi), \quad \xi \in \mathbb{R}. \tag{1.2}$$

Moreover, due to the dispersive effects C must be such that

$$\int_{\mathbb{R}} (1-\cos(k\xi))C(\xi)\,\mathrm{d}\xi > 0, \tag{1.3}$$

for every wave number $k \neq 0$.

Additionally, since the interaction between two material particles should become negligible as the distance between particles become very large, we can assume that

$$\lim_{\xi\to\pm\infty} C(\xi) = 0. \tag{1.4}$$

If a material is characterized by a finite horizon, so that no interactions happen within particles that have relative distance greater than δ, then we can assume that the support of the kernel function is given by $[-\delta,\delta]$ and the model (1.1) writes as

$$\partial_{tt}\theta(x,t) = \int_{B_\delta(x)} C(|x-y|)[\theta(x,t)-\theta(y,t)]\ \mathrm{d}y. \tag{1.5}$$

Of course, such condition is less restrictive than (1.4).

It is clear that a different microelastic material corresponds to a different kernel function and, as a consequence, the kernel function involved in the model provides different constitutive models.

In literature there are several kernel functions satisfying conditions (1.2), (1.3), and (1.4). In particular, according to [36], we will focus on a Gauss-type kernel in the form

$$C(\xi) = \lambda e^{-\mu\xi^2}, \qquad \lambda,\ \mu > 0.$$

Moreover, we aim to validate the choice of a distributed kernel function with shape

$$C(\xi) = \begin{cases} \frac{|\xi|-\lambda+\delta}{\delta}, & |\xi| \geq \lambda-\delta, \\ 0, & |\xi| < \lambda-\delta, \end{cases} \qquad \lambda > 0,$$

proposed in [7] in nonlocal unsaturated soil model contexts.

In this paper, we aim to solve the inverse problem described in (1.1) for learning the shape of the kernel function C, by implementing a Physics Informed Neural Network (PINN). More specifically, we show that this inverse problem requires a careful selection of activation functions in all the layers and a correct interaction with kernel initializers. It can be seen, in fact, that a naive choice on these functions would result in unreliable predictions and possibly unfeasible solutions. More precisely, we see that, if the neural network structure is not chosen accordingly to appropriate geometric knowledge relative to the data, then PINN output returns different, still acceptable, results, showing a lack of uniqueness. Therefore we will show that, as long as the peridynamic operator is bounded on a compact support $[-\delta,\delta]$ and the PINN architecture is build accordingly, as a consequence of the well posedness conditions of the peridynamic formulation, the learned kernel fulfills all the requirements expected, provided that PINN structure is accurate enough.

2 Introduction to PINNs

Physics-Informed Neural Networks (PINNs) have emerged as a transformative approach to tackle both direct and inverse problems associated with PDEs. These innovative neural network architectures seamlessly integrate the principles of physics into the machine learning framework. By doing so, PINNs offer a promising solution to efficiently and accurately model, simulate, and optimize complex systems governed by PDEs. More specifically, they can be employed to solve both direct and inverse problems; in the latter case, such PINNs are commonly referred to as inverse PINNs.

Direct problems involve finding solutions to PDEs that describe the evolution of physical systems under specified initial and boundary conditions. Traditional numerical methods, such as finite element analysis (see [18, 39]), finite difference methods with composite quadrature formulas (see [21]) and applied to spectral fractional models (see [12]), model order reduction methods (see [27]), meshfree methods (see [28, 30]), adaptive refinement techniques (see [2, 9]) and collocation and Galerkin methods (see [1]) have been widely used for solving direct problems. Moreover, more recently spectral methods with volume penalization techniques (see for instance [17, 20]) and Chebyshev spectral methods (see [22, 35]) have been developed in order to increase the order of convergence, to improve the accuracy of the results and to maintain the consistency of the method even in presence of singularities.

However, these approaches often require substantial computational resources and may struggle with high-dimensional or non-linear problems. Additionally, such methods need to know the constitutive parameters of the model such as the analytic expression of the kernel function, the size of the horizon and the Young's modulus to predict fractures in the material under consideration and, in suitable configurations, they fail to impose boundary conditions. In order to provide some hint in this direction, a data-driven approach can be developed. In [31] the authors propose a geometry-aware method in physics informed neural network to exactly imposing boundary conditions over complex domains. In [40] the authors investigate both a forward and an inverse problems of high-dimensional nonlinear wave equations via a deep neural networks with the activation function, while in [34] a combination of an orthogonal decomposition with a neural network is applied to build a reduced order model. In [23], the authors present an unsupervised convolutional neural network architecture with nonlocal interactions for solving PDEs using Peridynamic Differential Operator as a convolutional filter. Indeed, this approach results to be very efficient when the model is governed by an integral operator (see for instance [37]).

Inverse problems, on the other hand, are concerned with determining unknown parameters, boundary conditions, or the PDE itself, given limited or noisy observations of the system behavior. These problems frequently arise in real-world applications, including medical imaging [38], geophysics [4, 6], material characterization [10], and industrial process optimization [24]. Inverse problems are inherently ill-posed, as multiple solutions or no solutions may exist, making their resolution challenging. In fact, several issues could arise in solving inverse problems, especially related to irregular geometries [15], or also small data regimes, incomplete data or incomplete models [26].

In the context of nonlocal elasticity theory, in [33] the authors propose a methodology based on a constrained least squares optimization to solve inverse problems in heterogeneous media using state-based peridynamics in order to derive parameter values characterizing several material properties and to establish conditions for fracture patterns in geological setting.

Thus, Physics-Informed Neural Networks represent a paradigm shift in the way to approach direct and inverse problems associated with PDEs. Their ability to combine data-driven learning with physical principles opens up new frontiers in scientific research, engineering design, and problem-solving across a wide spectrum of domains.

2.1 PINN paradigm

In this paper, we will consider a Feed-Forward fully connected Neural Network (FF-DNN), also called Multi-Layer Perceptron (MLP) (see [5] and references therein).

In a PINN the solution space is approximated through a combination of activation functions, acting on all the hidden layers, with the independent variable used as the network inputs. Letting $x \in \mathbb{R}^n$, in a Feed-Forward network each layer feeds the next one through nested transformation, so that a it can be seen, letting L be the number of layers, as

$$\begin{aligned} &z_0 = x,\\ &z_l = \sigma_l\big(\Lambda_l(z_{l-1})\big),\ \ \Lambda_l(z_{l-1}) := W_l z_{l-1} + b_l, \quad l = 1,\dots,L, \end{aligned} \tag{2.1}$$

where, for each layer $l = 1,\dots,L$, $\sigma_l : \mathbb{R}^n \to \mathbb{R}^m$ is the activation function, which operates componentwise, W_l is the weight matrix and b_l is the bias vector. Thus, the output $z_L \in \mathbb{R}^m$ of a FF-NN can be expressed as a single function of the input vector x, defined as the composition of all the layers above in the following way:

$$z_L := (\sigma_L \circ \Lambda_L \circ \dots \circ \sigma_1 \circ \Lambda_1)(x).$$

The aim of a PINN is to minimize, through a Stochastic Gradient Descent method, a suitable objective function called *loss function*, that would take into account the physics of the problem, with respect to all the components, called trainable parameters, of W_l, b_l, for $l = 1,\dots,L$.

More specifically, given a general PDE of the form $\mathcal{P}(f) = 0$, where $\mathcal{P}$ represents the differential operator acting on f, the loss function used by a PINN is usually given by

$$\mathcal{L}(f) := \|f - f^*\| + \|\mathcal{P}(f) - 0^*\|, \tag{2.2}$$

where f^* is the training dataset (of points inside the domain or on the boundary), and 0^* is the expected (true) value for the differential operation $\mathcal{P}(f)$ at any given training or sampling point; the chosen norm $\|\cdot\|$ (it may be different for each term in the loss function) depends on the functional space and the specific problem. Selecting a correct norm (so to avoid overfitting) for the loss function evaluation is an important problem in PINN, and recently in [32] authors have proposed spectral techniques based on Fourier residual method to overcome computational and accuracy issues. The first term in the right-hand side of (2.2) is referred to as data fitting loss, while the second term is referred to as residual loss, which is responsible to make a NN be informed by physics. We address the construction of the loss function in Sect. 3.3.

The operator $\mathcal{P}$ is usually performed using `autodiff` (Automatic Differentiation algorithm). In the context of peridynamic theory, in [16] authors propose, for the first time, a nonlocal alternative to `autodiff` by replacing the evaluation of f and its partial derivatives through the action of a Peridynamic Differential Operator (PDDO) on f.

A recent review on PINNs and related theory can be found in [11].

3 RBF-iPINN for the kernel function

In case one wants to solve an inverse problem, there will be more trainable parameters than only those coming from weight matrices and bias vectors. Hence, such further parameters have to be considered in the minimization iterations and their respective gradients must be computed as well.

However, in our case, the inverse problem does not involve the mere computation of scalar quantities, but rather a whole function, specifically the kernel function C in (1.1), which has also analytical and geometrical properties to be accounted for, such as nonnegativity and symmetry. In the PINN architecture proposed, these features reflect in the implementation of a NN model with two separated sets of layers, one for C and the other for θ, and whose output would be both the solution to (1.1) and the unknown function C; moreover, the loss function (2.2) has been accordingly endowed with further terms necessary to enforce the requirements on C.

From the point of view of the architecture, while nonnegativity of C has been simply enforced by requiring all trainable parameters in (2.1) to be nonnegative, symmetry has required a more specific treatment, both in terms of activation functions and in the way we have defined the loss function. Our idea has been to wisely select different activation functions for the two different sets of layers, inspired by the properties coming along with C and the data on θ. In the following section we will introduce and discuss the technical approaches to deal with activation and loss functions.

3.1 Radial Basis Function Layer

As activation function for the first layer, whose input is x, a Radial Basis Function (RBF) is selected. By definition, a Radial Basis Function (RBF) is a real-valued function whose output depends only on the distance from a fixed center or prototype point. An RBF can be defined as

$$\sigma_{\text{rbf}}(x) = \phi(\|x - c\|), \tag{3.1}$$

where ϕ is the RBF function, x is the input to the RBF, c is the center or prototype point, $\|x - c\|$ represents the distance between x and c.

RBFs are commonly used in various fields and, when used as activation functions in neural networks, they give rise to Radial Basis Function Neural Networks (RBFNNs) (see [14]).

We have considered two families of RBFs (3.1), sketched in Fig. 1, given by

$$\sigma_{\text{rbf}}(x) := \frac{\rho}{1 + \gamma(x - \mu)^2}, \quad \rho, \gamma, \mu > 0, \tag{3.2a}$$

$$\sigma_{\text{rbf}}(x) := \rho\sqrt{1 + \gamma(x - \mu)^2}, \quad \rho, \gamma, \mu > 0, \tag{3.2b}$$

called inverse quadratic and multiquadric RBFs respectively, where all the parameters above could be taken to be trainable. This approach has recently been proposed in [3] in the context of direct problem for nonlinear PDEs, but used in the middle layer. However, we have extensively noticed that such a choice is not efficient to learn the kernel function C in (1.1), resulting in nonphysical

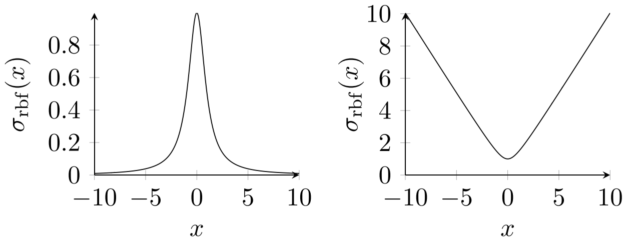

Fig. 1 Qualitative shapes of Radial Basis Functions defined in (3.2a) and (3.2b), respectively

results and excessively large computational time and cost. In fact, this has led us to introduce an RBF inverse PINN, that we called RBF-iPINN, where we select a Radial Basis Function as the activation function for the first layer, that has significantly sped up performance while providing the expected result if compared to the exact solution.

3.2 RBF-iPINN Structure

Since the kernel function C depends on the sole spatial variable x, while the solution to (1.1) θ depends on both space and time, then x, t need to be handled separately. To this purpose, the proposed RBF-iPINN is implemented in a serialized fashion, suitably connecting, as we are going to explain in details below, two different Neural Networks, which we may call *spatial NN* and *temporal NN*. In the spatial NN, the spatial variable x is the sole input of a hidden layer of 20 neurons, activated by an RBF as in (3.2), followed by 8 layers with 20 neurons each, activated by `ReLu` function. The output of this sequence of layers is then concatenated with t, providing the input for the temporal Neural Network. More specifically, this second part is made up by 8 layers, each containing 20 neurons and activated by a `sigmoid` function. Finally, the overall output of the RBF-iPINN is returned as an array that lists two tensors, the first carrying the kernel C, and the other carrying the dependent variable θ; the structure of the RBF-iPINN is sketched in Fig. 2.

Let us notice that selecting the `ReLU` activation function for all the layers of the architecture could result in a loss of compatibility potential of the PINN, as reported in [19]. This consideration, also supported by several experiments, justifies the choice of the `sigmoid` activation function in the temporal NN. We witness that, however, other selections than `sigmoid` function do not perform satisfactorily enough as in the cases reported in Sect. 4.

Moreover, the spatial NN is endowed with a kernel initializer of type `glorot_normal`, to better keep the variance of the weights consistent across layers, thus helping with

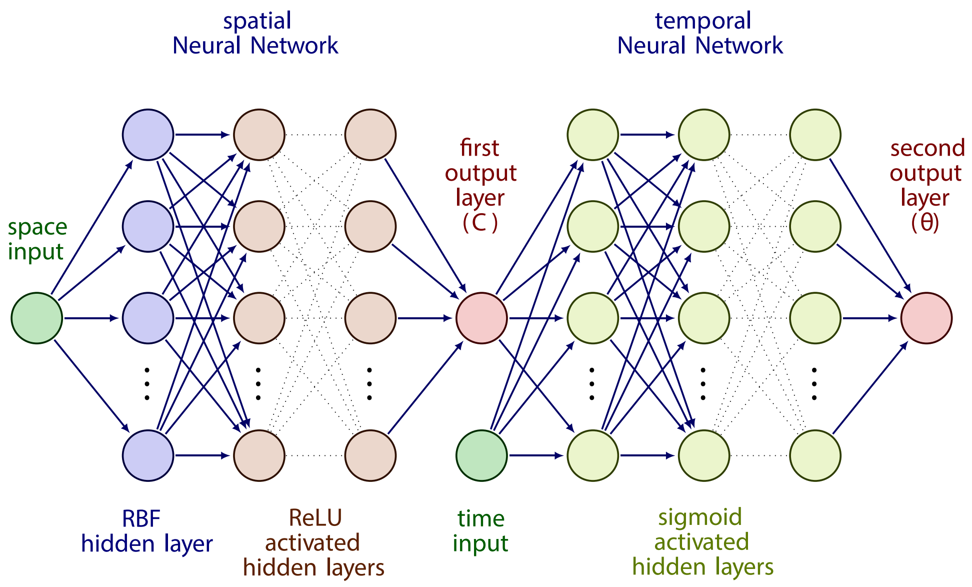

Fig. 2 RBF-iPINN structure. In our simulations, we set the number of `ReLU` and `sigmoid` activated hidden layers to 8, and the number of neurons per each layer, including the RBF layer, to 20

training stability, while the temporal NN is endowed with a `random_uniform` kernel initializer; also, a nonnegative kernel constraint and a kernel regularizer of type `l1_l2`, with weights $l_1 = l_2 = 0.01$ is applied to the spatial NN, which yields the computed kernel function. The nonnegativity constraint is expected to take care of that the kernel function is nonnegative, and is also coherent with the nonnegativity of the RBF activating the first layer of the spatial NN, which is in turn fed into the successive layers; the `l1` parameter is meant to avoid overfitting, to encourage sparsity and to effectively perform feature selection; through the `l2` parameter, on the other hand, we enforce the spatial NN to be more robust to outliers.

3.3 Loss Function

Given the constraints on C, we have to accordingly construct a loss function as in (2.2) with as many components as there are constraints to be enforced in the RBF-PINN. More specifically, we consider the following components to be part of $\mathcal{L}$:

$$\text{pde loss: } \mathcal{L}_{\text{pde}} := \|\mathcal{P}(f) - 0^*\|_2, \tag{3.3a}$$

$$\text{data fitting loss: } \mathcal{L}_{\text{data}} := \|f - f^*\|_2, \tag{3.3b}$$

$$\text{symmetry loss: } \mathcal{L}_{\text{sym}} := \|f(x,t) - f(-x,t)\|_1. \tag{3.3c}$$

In the selection of norms above, we have been guided by the features we want our PINN to take into account. More specifically, since we are interested in fitting data and in satisfying our model as much as possible, we selected the 2-norm for these contributions; however, since symmetry is to be expected from the PINN architecture and, in particular, from the first RBF layer, we measure its loss via the 1-norm, which is more sensible to small errors. Finally, we consider a weighted sum of the contributions given above as

$$\mathcal{L} = w_{\text{pde}}\mathcal{L}_{\text{pde}} + w_{\text{data}}\mathcal{L}_{\text{data}} + w_{\text{sym}}\mathcal{L}_{\text{sym}}, \tag{3.4}$$

where, for our simulations in Sect. 4, we have set

$$w_{\text{pde}} = 2,\ w_{\text{data}} = 1,\ w_{\text{sym}} = 2.$$

These values have been suitably tuned, and turned out to perform well in all the experiments we carried out below.

3.4 Learning rate

The learning rate α has been selected to be decreasing with the epoch in a quadratic way. More precisely, we implemented the following scheduler:

$$\begin{aligned} &\alpha_0 = 10^{-4},\ \alpha_1 = 0.7\alpha_0 \\ &\alpha_i = \left(1 - \left(\frac{i}{N}\right)^2\right)\alpha_0 + \left(\frac{i}{N}\right)^2 \alpha_1, \quad i = 0, \dots, N, \end{aligned} \tag{3.5}$$

where N is the number of epochs chosen for the training. Thus, starting with a learning rate of α_0 at epoch 0, it progressively gets reduced according to (3.5) over the epochs, until it reaches the value α_1 at epoch N.

4 Numerical Simulations

In this section, we show results with our RBF-iPINN. All the experiments have been run using 1000 epochs with a learning rate defined in (3.5) and employed the ADAM optimizer.

The machine used for the experiments is an Intel Core i7-8850 H CPU at 2.60GHz and 64 GB of RAM. Moreover, the PINNs, providing results in the examples below, have been developed in Python 3.10, using the library TensorFlow 2.15.0 within the interface Keras 3.0.1.

Moreover, real data, which are used in our simulations to compute the loss function (3.3b), are synthetically built using appropriate spectral methods from [20] to solve (1.1).

A main feature of the numerical computation is the evaluation of the integral on the right-hand side of (1.1). In fact, in order to exploit the power of Keras on computing convolution products, we notice that

$$\begin{aligned}\int_{\mathbb{R}} & C(|x-y|)[\theta(x,t)-\theta(y,t)]\,\mathrm{d}y \\ &= \theta(x,t)\int_{\mathbb{R}} C(|x-y|)\,\mathrm{d}y - \int_{\mathbb{R}} C(|x-y|)\theta(y,t)\,\mathrm{d}y \\ &= \theta(x,t)\int_{\mathbb{R}} C(|x-y|)\,\mathrm{d}y - C(|x|) * \theta(x,t),\end{aligned}$$

where the second term in the right-hand side above is the convolution product between the kernel C and the unknown function θ. It has to be noticed here that the kernel function C is compactly supported, with support $[-\delta,\delta]$. Now, in order to numerically compute such convolution product, let $[0,X]$ be the space interval and let $0<x_1<x_2<\ldots<x_{N-1}<x_N=X$ be the uniform spatial discretization of the interval $[0,X]$ with stepsize $h>0$. the convolution product above can be numerically treated by determining the exact number of components in the vector $[C(x_i)]_{i=1}^{n}$ so that only points x_i such that

$$|x_i-x_j|<\delta, \quad i,j=1,\ldots,N,$$

come into play when computing $C(\|x\|) * \theta(x,t)$. Since $x_i = i\cdot h$, then we deduce that the only indices involved in the convolution product are $i,j=1,\ldots,N$ such that

$$|i-j|<\frac{\delta}{h}.$$

Since the peridynamic integral-operator in (1.5) is linear, we can derive in terms of the Green's function a solution to the initial-value problem using continuous Fourier Transform. Such solution can be used to provide a dataset for the next simulations.

In the next two experiments, we exemplify on datasets derived from V-shaped kernel functions. This choice of kernel is justified by the fact that it is implemented in some nonlocal formulations of Richards' equation as it is able to easily incorporate Dirichlet boundary conditions in the model (see [7]).

Thus, we select as activation function for the first layer of the spatial NN an RBF of type (3.2b). Further, we tried both to keep all the three parameters γ,ρ,μ trainable, and to fix γ while letting ρ,μ be trainable. According to our experience and for the following two cases, fixing γ improves convergence performance and result quality.

Example 4.1 Here we consider a dataset with $t\in[0,20]$, $x\in[-10,10]$ with spatial stepsize $h=2\cdot10^{-1}$ and $\delta=10$, and for which the analytical expression of the kernel is

$$C(x)=\frac{3}{5}|x|. \tag{4.1}$$

We set $\gamma=0.09$, obtaining the results showed in Fig. 3a, where we compare the true kernel function in (4.1) to the output of the proposed inverse RBF-iPINN. Setting $\gamma=0.05$ in (3.2b) provides qualitatively comparable results, as can be observed in Fig. 3b.

Example 4.2 Here we consider a dataset with $t\in[0,20], x\in[-10,10]$ with spatial stepsize $h=2\cdot10^{-1}$ and $\delta=1$, and for which the analytical expression of the kernel is

$$C(x)=\begin{cases}\frac{\delta-x-10}{\delta}, & x\le -10+\delta,\\ 0, & -10+\delta<x\le 10-\delta,\\ \frac{\delta+x-10}{\delta}, & x>10-\delta.\end{cases} \tag{4.2}$$

We set $\gamma=0.09$, obtaining the results showed in Fig. 4a, where we compare the true kernel function in (4.2) to the output of the proposed inverse RBF-iPINN. Setting $\gamma=0.05$ provides results in Fig. 4b.

In the next example, we consider a bell-shaped kernel function to test the proposed RBF-iPINN. Accordingly, an RBF of type (3.2a) is chosen to activate the first layer of the spatial NN.

Example 4.3 For this example, first we tuned hyperparameters, setting the kernel regularizers `l1_l2` with weights 0.01 and 0.1 respectively, in order to try and catch, as better as possible, the correct qualitative behavior of the kernel in the interior of its compact support; moreover, on the account of the knowledge of the kernel shape, we decided to activate the first layer of the RBF-iPINN through (3.2a), where we set the hyperparameter $\gamma=1$; finally, for a better data fitting, we also selected the sup-norm in (3.3b).

In this case, the neural network shows a discrete ability to catch shape and support of the bell-shaped kernel, but fails in a good approximation of the characteristic parameters, as shown in Fig. 5. In fact, in this case the true kernel is given by

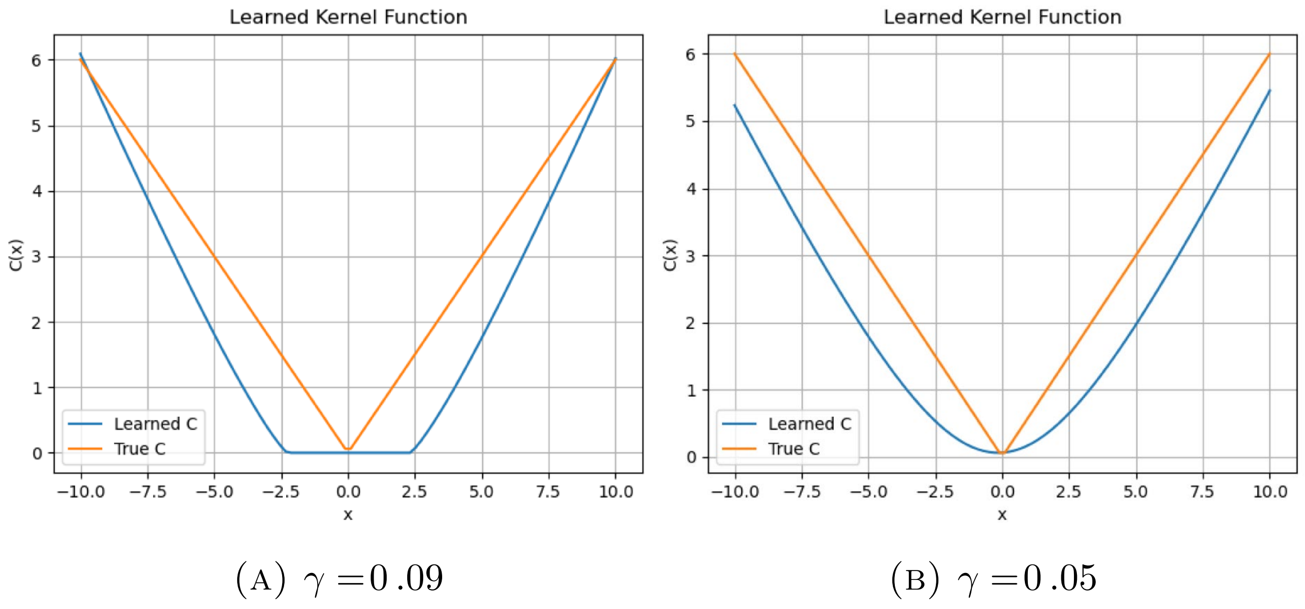

Fig. 3 Learned kernel functions relative to Example 4.1 for different values of γ in (3.2b)

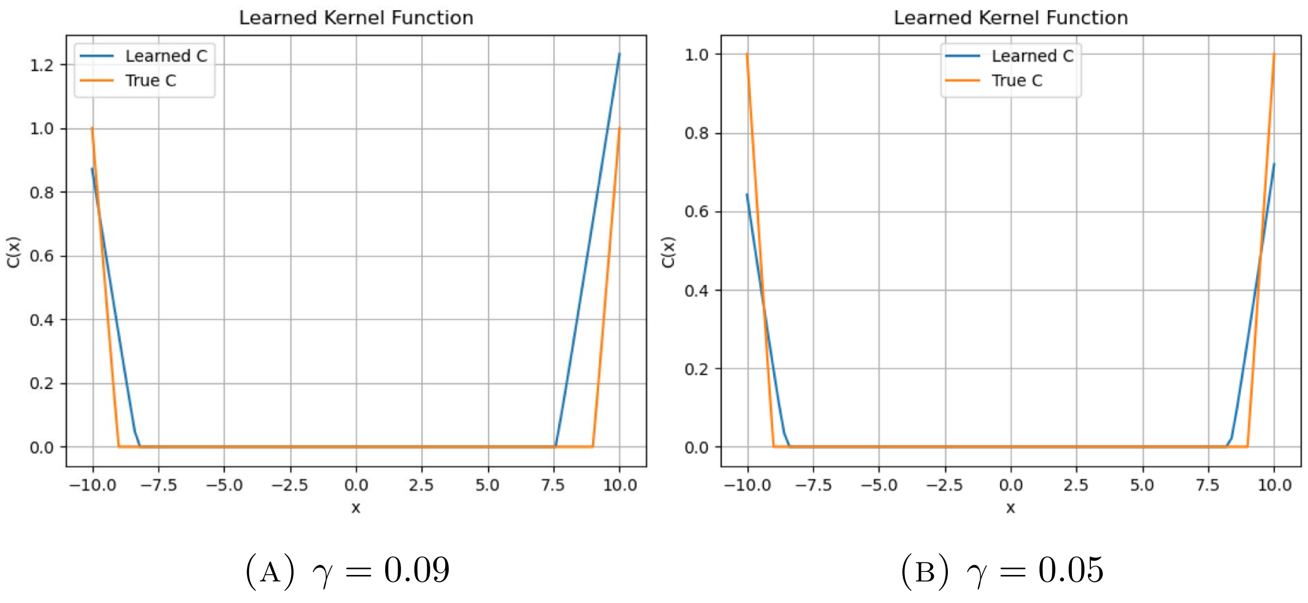

Fig. 4 Learned kernel functions for different values of γ in (3.2b) relative to Example 4.2

$$C(x) = \frac{4}{\sqrt{\pi}} e^{-x^2}. \tag{4.3}$$

Therefore, we have performed a further analysis by implementing a standard inverse PINN to learn parameters γ^* and σ^* in

$$C^*(x) := \gamma^* e^{-\sigma^* x^2}. \tag{4.4}$$

Starting with initial guesses for $\gamma^* = 3$ and for $\sigma^* = 0.5$, we run a PINN whose structure is the same as the second portion relative to θ of the RBF-iPINN described above (see the architecture in Fig. 2). The training phase has been performed over 1000 epochs and with the same learning rate

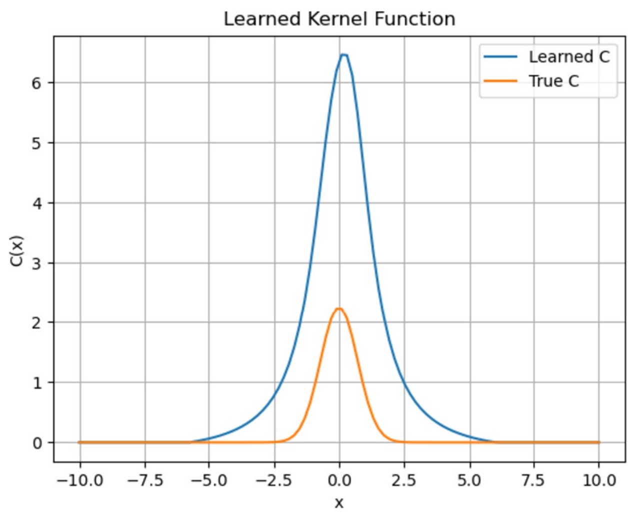

Fig. 5 Learned kernel function compared to the true one from Example 4.3

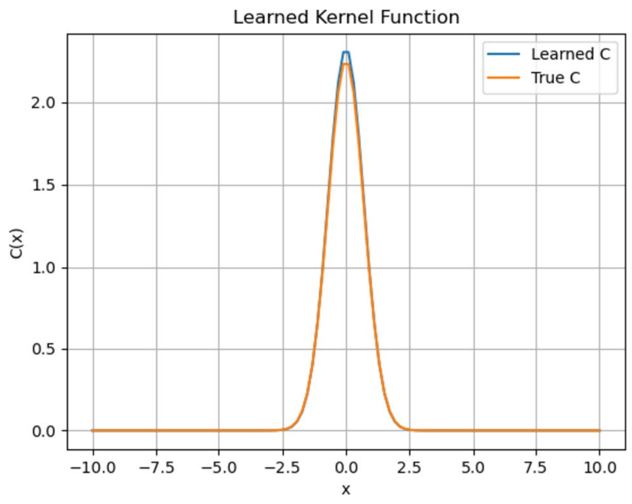

Fig. 6 Gaussian kernel function learned from (4.4), compared to the true one from Example 4.3

scheduler described in Sect. 3.4, but with a faster descent obtained by setting $\alpha_0 = 10^{-3}$. Results are depicted in Fig. 6. It can be deduced that the inverse PINN has been able to correctly detect the learned values which, at convergence, are given by $\gamma^* = 2.3302033$ and $\sigma^* = 1.0218402$, being $\frac{4}{\sqrt{\pi}} \approx 2.2567583$.

Remark 4.4 We stress that a prior geometrical knowledge about the kernel function to learn is necessary to correctly set up the spatial NN of the RBF-iPINN. In fact, we report that, activating the spatial NN with an RBF of type (3.2a) in Example 4.1 and Example 4.2 results in poor and nonphysical predictions; similar negative results show up if RBFs of type (3.2b) are used in Example 4.3.

5 Conclusions

In this work we have analyzed a peridynamic formulation of a classical wave equation, trying to compute the kernel function responsible for the nonlocal behavior of the model considered. We have proposed to utilize a Radial Basis Function (RBF) as activation function for the first layer of a suitably designed Physics Informed Neural Network (PINN) to solve the inverse problem. Specifically, our inverse PINN architecture has two neural networks working in series: the first set of layers, called spatial NN and whose first layer activated by some Radial Basis Function, takes the spatial data as input and yields the first output for recovering the kernel function; then, this first output is concatenated to the temporal data, thus providing a new tensor serving as input to the second set of layers, that is called temporal NN and that produces the output describing θ, solution to the peridynamic wave equation. We called the proposed model RBF-iPINN. We have shown that, with a wise scheduler for the learning rate and necessary initializations of the spatial NN responsible for the kernel function computation, for standard selections of the activation RBF, the RBF-iPINN can provide reliable prediction of the kernel function, in case it has a V-shape behavior. We also tackle the case of Gaussian kernel function: here, RBF-iPINN is able to adequately detect the shape, but a further analysis is necessary to learn the expected parameters of a bell-shaped function. We did so by implementing a standard inverse PINN, practically tuning the very same second set of layer of the RBF-iPINN.

Such models turn out to be promising tools for investigating optimal controls problems in dimension 2 or higher (see, e.g., [8]), where an inverse PINN approach seems to provide robust and scalable results. Such considerations pave the way to further investigations about how to deal with more complicated peridynamic models via PINNs and Radial Basis Functions, that seem to be a powerful approach to this kind of problems, due to their inherently symmetric nature.

Acknowledgements SFP has been supported by *REFIN* Project, grant number D1AB726C funded by Regione Puglia, and by *PNRR MUR - M4C2* project, grant number N00000013 - CUP D93C22000430001. The three authors gratefully acknowledge the support of INdAM-GNCS 2023 Project, grant number CUP_E53C22001930001, and INdAM-GNCS 2024 project, grant number CUP_E53C23001670001. They are also part of the INdAM research group GNCS.

Author Contributions All authors contributed equally to the paper.

Funding Open access funding provided by Consiglio Nazionale Delle Ricerche (CNR) within the CRUI-CARE Agreement.

Availability of data and materials No datasets were generated or analysed during the current study.

Declarations

Conflict of interest The authors declare no conflict of interest.

References

1. Alebrahim R (2023) Modified wave dispersion properties in 1D and 2D state-based peridynamic media. Comput Math Appl 151:21–35
2. Alebrahim R, Marfia S (2023) Adaptive PD-FEM coupling method for modeling pseudo-static crack growth in orthotropic media. Eng Fract Mech 294:109710
3. Bai Jinshuai, Liu Gui-Rong, Gupta Ashish, Alzubaidi Laith, Feng Xi-Qiao, YuanTong Gu (2023) Physics-informed radial basis network (PIRBN): A local approximating neural network for solving nonlinear partial differential equations. Computer Methods in Applied Mechanics and Engineering 415:116290
4. Bandai T, Ghezzehei TA (2022) Forward and inverse modeling of water flow in unsaturated soils with discontinuous hydraulic conductivities using physics-informed neural networks with domain decomposition. Hydrol Earth Syst Sci 26(16):4469–4495
5. Bengio Y, Ducharme R, Vincent P, Janvin C (2003) A neural probabilistic language model. J Mach Learn Res 3:1137–1155
6. Berardi M, Girardi G (2024) Modeling plant water deficit by a nonlocal root water uptake term in the unsaturated flow equation. Commun Nonlinear Sci Numer Simul 128:107583
7. Berardi M, Difonzo FV, Pellegrino SF (2023) A numerical method for a nonlocal form of Richards' Equation based on Peridynamic theory. Comput Math Appl 143:23–32
8. Berardi M, Difonzo FV, Guglielmi R (2023) A preliminary model for optimal control of moisture content in unsaturated soils. Comput Geosci 27(6):1133–1144
9. Bobaru F, Yang M, Alves S, Silling F, Askari E, Xu J (2009) Convergence, adaptive refinement, and slaning in 1D peridynamics. Int J Numer Mech Eng 77:852–877
10. Chen X, Cao BT, Yuan Y, Meschke G (2023) Transfer learning based physics-informed neural networks for solving inverse problems in engineering structures under different loading scenarios. Comput Methods Appl Mech Eng 405:115852
11. Cuomo S, Cola VSD, Giampaolo F, Rozza G, Raissi M, Piccialli F (2022) Scientific machine learning through physics-informed neural networks: where we are and what's next. J Sci Comput 92(3):88
12. Difonzo FV, Garrappa R (2023) A numerical procedure for fractional-time-space differential equations with the spectral fractional laplacian. In: Angelamaria C, Marco D, Fabio D, Roberto G, Mariarosa M, Marina P (eds) Fractional Differential Equations, pages 29–51, Springer Nature Singapore, Singapore
13. Emmrich E, Puhst D (2015) Survey of existence results in nonlinear peridynamics in comparison with local elastodynamics. Comput. Methods Appl. Math. 15(4):483–496
14. Fasshauer GE (2007) Meshfree approximation methods with Matlab (With Cd-rom). World Scientific Publishing Company, Interdisciplinary Mathematical Sciences
15. Gao H, Zahr MJ, Wang J-X (2022) Physics-informed graph neural Galerkin networks: a unified framework for solving PDE-governed forward and inverse problems. Comput Methods Appl Mech Eng 390:114502
16. Haghighat E, Bekar AC, Madenci E, Juanes R (2021) A nonlocal physics-informed deep learning framework using the peridynamic differential operator. Comput Methods Appl Mech Eng 385:114012
17. Jafarzadeh S, Larios A, Bobaru F (2020) Efficient solutions for nonlocal diffusion problems via boundary-adapted spectral methods. J Peridyn Nonlocal Model 2:85–110
18. Kilic B, Madenci E (2010) Coupling of peridynamic theory and the finite element method. J Mech Mater Struct 5(5):703–733
19. Kuangdai L, Jeyan T (2023) On the compatibility between neural networks and partial differential equations for physics-informed learning
20. Lopez L, Pellegrino SF (2021) A spectral method with volume penalization for a nonlinear peridynamic model. Int J Numer Methods Eng 122(3):707–725
21. Lopez L, Pellegrino SF (2022) A space-time discretization of a nonlinear peridynamic model on a 2D lamina. Comput Math Appl 116:161–175
22. Luciano Lopez and Sabrina Francesca Pellegrino (2023) Computation of eigenvalues for nonlocal models by spectral methods. J Peridyn Nonlocal Model 5(2):133–154
23. Mavi A, Bekar AC, Haghighat E, Madenci E (2023) An unsupervised latent/output physics-informed convolutional-LSTM network for solving partial differential equations using peridynamic differential operator. Comput Methods Appl Mech Eng 407
24. Meng Z, Qian Q, Mengqiang X, Bo Y, Yildiz AR, Mirjalili Seyedali (2023) PINN-FORM: a new physics-informed neural network for reliability analysis with partial differential equation. Comput Methods Appl Mech Eng 414:116172
25. Oterkus S, Madenci E, Agwai A (2014) Peridynamic thermal diffusion. J Comput Phys 265:71–96
26. Raissi M, Perdikaris P, Karniadakis GE (2019) Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J. Comput. Phys. 378:686–707
27. Regazzoni F, Dedè L, Quarteroni A (2019) Machine learning for fast and reliable solution of time-dependent differential equations. J. Comput. Phys. 397:108852
28. Shojaei A, Mudric T, Zaccariotto M, Galvanetto U (2016) A coupled meshless finite point/Peridynamic method for 2D dynamic fracture analysis. Int J Mech Sci 119:419–431
29. Silling SA (2000) Reformulation of elasticity theory for discontinuities and long-range forces. J Mech Phys Solids 48(1):175–209
30. Silling S, Askari E (2005) A meshfree based on the peridynamic model of solid mechanics. Comput Struct 83(17–18):1526–1535
31. Sukumar N, Srivastava A (2022) Exact imposition of boundary conditions with distance functions in physics-informed deep neural networks. Comput Methods Appl Mech Eng 389:114333
32. Taylor JM, Pardo D, Muga I (2023) A deep fourier residual method for solving PDEs using neural networks. Comput Methods Appl Mech Eng 405:115850
33. Turner DZ, van Bloemen Waanders BG, Parks ML (2015) Inverse problems in heterogeneous and fractured media using peridynamics. J Mech Materi Struct 10(5)

34. Vitullo P, Colombo A, Franco NR, Manzoni A, Zunino P (2024) Nonlinear model order reduction for problems with microstructure using mesh informed neural networks. Finite Elements Anal Design 229:104068
35. Wang L, Jafarzadeh S, Mousavi F, and Bobaru F (2023) PeriFast/Corrosion: A 3D Pseudospectral Peridynamic MATLAB Code for Corrosion. J Peridynamics Nonlocal Model:1–25
36. Weckner O, Abeyaratne R (2005) The effect of long-range forces on the dynamics of a bar. J Mech Phys Solids 53(3):705–728
37. Yuan L, Ni YQ, Deng XY, Hao S (2022) A-PINN: auxiliary physics informed neural networks for forward and inverse problems of nonlinear integro-differential equations. J Comput Phys 462
38. Yuyao Chen LL, Karniadakis GE, Dal Negro L (2020) Physics-informed neural networks for inverse problems in nano-optics and metamaterials. Opt Express 28(8):11618–11633
39. Zaccariotto M, Mudric T, Tomasi D, Shojaei A, Galvanetto U (2018) Coupling of FEM meshes with Peridynamic grids. Comput Methods Appl Mech Eng 330:471–497
40. Zhou Z, Wang L, Yan Z (2023) Deep neural networks learning forward and inverse problems of two-dimensional nonlinear wave equations with rational solitons. Comput Math Appl 151:164–171