Computational and Applied Mathematics (2023) 42:178
https://doi.org/10.1007/s40314-023-02323-9

A decoupled physics-informed neural network for recovering a space-dependent force function in the wave equation from integral overdetermination data

Aydin Sarraf[1] · Fatemeh Baharifard[2] · Kamal Rashedi[3]

Received: 1 September 2022 / Revised: 13 April 2023 / Accepted: 26 April 2023 /
Published online: 11 May 2023

Abstract
In this paper, we study the inverse problem of approximating a space-dependent wave source in a one-dimensional wave equation. The problem is converted to a nonclassical second-order hyperbolic equation and is later reduced to the solution of a linear system of algebraic equations by employing the Ritz collocation method. The obtained Ritz approximation is used to train a physics-informed neural network (PINN). However, unlike the conventional training in PINN, the loss function of the proposed method is not additive and the loss terms are decoupled in the sense that the neural network is re-trained separately according to each loss term. Hence, we name the proposed method decoupled PINN or D-PINN for short. Since the function of the initial condition is nonlinear and continuously differentiable, it is used, up to a scalar, as the activation function as opposed to using an off-the-shelf activation function such as hyperbolic tangent. Experiments indicate that D-PINN is capable of approximating the solution more accurately than Ritz and PINN.

Keywords Physics-informed neural networks · Inverse wave problem · Ritz approximation

Mathematics Subject Classification 35R30 · 35L20 · 65M70 · 68T07

✉ Aydin Sarraf
aydin.sarraf@ericsson.com

Fatemeh Baharifard
f.baharifard@ipm.ir

Kamal Rashedi
k.rashedi@mazust.ac.ir

1 Global Artificial Intelligence Accelerator, Ericsson, Montreal, QC H4S 0B6, Canada

2 School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran

3 Department of Mathematics, University of Science and Technology of Mazandaran, Behshahr, Iran

1 Introduction

Consider the problem of simultaneous determination of the functions $u(x,t)$ and $f(x)$ satisfying the wave equation,

$$u_{tt}(x,t) - u_{xx}(x,t) = f(x)g(x,t) + h(x,t), \quad in \quad (0,L)\times(0,T), \tag{1.1}$$

along with the initial conditions

$$u(x,0) = u_0(x), \quad u_t(x,0) = u_1(x), \quad 0 \le x \le L, \tag{1.2}$$

and the boundary conditions

$$u(0,t) = b_0(t), \quad u(L,t) = b_1(t), \quad 0 < t < T, \tag{1.3}$$

and the overdetermination data

$$\int_0^T \omega(t)u(x,t)dt = u_T(x), \quad x \in [0,L]. \tag{1.4}$$

In the system of Eqs. (1.1)–(1.4), the functions

$$u_0(x),\ u_1(x),\ b_0(t),\ b_1(t),\ \omega(t),\ u_T(x), \tag{1.5}$$

are given. We further assume that the following compatibility conditions hold

$$b_0(0) = u_0(0),\ u_1(0) = b_0'(0),\ b_1(0) = u_0(L),\ b_1'(0) = u_1(L), \tag{1.6}$$

$$\int_0^T \omega(t)b_0(t)dt = u_T(0),\ \int_0^T \omega(t)b_1(t)dt = u_T(L). \tag{1.7}$$

Theoretical results concerning the existence and uniqueness of the solution for the inverse problem (1.1)–(1.4) were presented in Lesnic et al. (2016).

From a practical point of view, the inverse problem (1.1)–(1.4) and other similar problems are important in applications related to unknown force loads (Lesnic et al. 2016) such as designing final states or time-averaged displacement by controlling the wave source. The problem of recovering the space-dependent force function in the second order hyperbolic equation using various boundary conditions has been discussed in several articles and useful theoretical and numerical results have been obtained (Cannon and Dunninger 1970; Engl et al. 1994; Hussein and Lesnic 2014, 2016; Lesnic et al. 2016; Nguyen 2019; Yamamoto 1995). In Cannon and Dunninger (1970), the authors considered the Neumann boundary condition at $x = 0$ as an extra specification and established the existence and uniqueness of a classical solution for the inverse problem. In Yamamoto (1995), the author presented a regularized reconstruction formula for the space-dependent force function from the Neumann boundary data. In Engl et al. (1994), the authors established uniqueness proofs for the solution of inverse problems of identifying the force function from Dirichlet condition in linear partial differential equations including the wave equation. In Lesnic et al. (2016), the problem of determining the source of the wave equation is discussed and necessary conditions for the uniqueness of the solution are provided when the additional condition either has an integral form or is a wave displacement in the finite time. In Nguyen (2019), the authors considered the problem of reconstructing sources from the lateral Cauchy data of the wave field on the boundary of a domain and employed the quasi-reversibility method to find its regularized solution. In Hussein and Lesnic (2014, 2016), the boundary element method (BEM) is combined with a regularized method of separating variables to determine an unknown space-dependent force function.

In Rashedi et al. (2022), the authors considered the problem of recovering a space-dependent force function in the wave equation for two different cases of extra conditions, involving measurements of the flux at the boundary $x = 0$ and the wave displacement at the final time which are different from (1.1)–(1.4), and solved the problem numerically using the Ritz-collocation method. For more background on inverse wave problems and their solution methods from both theoretical and numerical point of view, we refer the interested reader to Isakov (2006); Prilepko et al. (2000); Samarskii and Vabishchevich (2007).

Numerical solutions of direct and inverse problems for differential equations can be divided into two main categories: classical numerical methods and scientific machine learning (SciML) methods. Some well-known classical numerical methods include finite difference methods, finite volume methods, finite element methods, and spectral methods. Scientific machine learning methods involve the application of machine learning models such as support vector machines and neural networks to solve direct and inverse problems for differential equations. A hybrid category can also be considered as the combination of both classical numerical methods and SciML methods. Our approach in this paper falls into this hybrid category as we use both the Ritz method and PINN to solve certain inverse problems. Arguably, the most popular SciML method for solving differential equations is the physics-informed neural network (PINN). Although the precursors of PINN date back to early 1990 s, the term PINN was coined in 2017 (Blechschmidt and Ernst 2021), and since then the number of published papers related to PINN is in the thousands (Cuomo et al. 2022, Fig. 1). Applications of PINN are wide-ranging including ordinary, partial, integro-, and stochastic differential equations. Some examples of the problems that can be solved by PINN are incompressible Navier–Stokes equations (Jin et al. 2021), parameterized steady-state PDEs on the irregular domain (Gao et al. 2021), fractional telegraph equation, Euler equations that model high-speed aerodynamic flows (Mao et al. 2020), and various inverse problems (Jagtap et al. 2022; Gao et al. 2022; Jagtap et al. 2020c; Mishra and Molinaro 2022; Zhang et al. 2019; Yuan 2022).

In the most general form, PINN is capable of solving the following differential equations (Cuomo et al. 2022):

$$\begin{aligned}\mathcal{F}(u(z),\gamma) &= f(z), \quad z \in \Omega\\ \mathcal{B}(u(z)) &= g(z), \quad z \in \partial\Omega\end{aligned} \tag{1.8}$$

where $\Omega \subset \mathbb{R}^d$, $\partial\Omega$ is the boundary of Ω, $z = (x_1, ..., x_d, t)$ is the space-time coordinate vector, u is the unknown solution, γ are the parameters related to the physics, f is the function identifying the data of the problem, g is the boundary function, $\mathcal{F}$ is the nonlinear differential operator, and $\mathcal{B}$ is the operator indicating initial or boundary conditions. In PINN, the unknown solution u is approximated by a neural network, typically a multilayer perceptron, where the parameters of the neural network are obtained by minimizing the following loss function (Cuomo et al. 2022):

$$\theta^* = \underset{\theta}{\operatorname{argmin}}(w_{\mathcal{F}}\mathcal{L}_{\mathcal{F}}(\theta) + w_{\mathcal{B}}\mathcal{L}_{\mathcal{B}}(\theta) + w_{\mathcal{D}}\mathcal{L}_{\mathcal{D}}(\theta)), \tag{1.9}$$

where $\mathcal{L}_{\mathcal{F}}$ is the physics-informed loss function (PI-loss), $\mathcal{L}_{\mathcal{B}}$ is the boundary and initial loss function (BI-loss), $\mathcal{L}_{\mathcal{D}}$ is the data loss function (D-loss), and $w_{\mathcal{F}}$, $w_{\mathcal{B}}$, $w_{\mathcal{D}}$ are their corresponding weights. For direct (forward) problems, PINN typically includes the first two terms of the loss function, namely $\mathcal{L}_{\mathcal{F}}$ and $\mathcal{L}_{\mathcal{B}}$. Therefore, for direct problems, PINN is an unsupervised learning method because the first two loss terms do not use any external data for training. In contrast, for inverse problems, PINN can be regarded as a weakly-supervised learning method as the third loss term requires some noisy (approximation) data for training. If the data is not noisy and comes from the exact solution in a region of the domain, then PINN can be considered as a supervised learning method. In our approach, the data comes from an approximation (Ritz method) instead of the exact solution. Therefore, our learning method belongs to the class of weakly-supervised learning methods. Originally, the loss terms of PINN did not include any weights (Raissi et al. 2019), i.e. $w_{\mathcal{F}}$, $w_{\mathcal{B}}$, and $w_{\mathcal{D}}$ were set to 1. Later, the loss formulation of PINN was generalized by including weights and designing algorithms to find the optimal weights (Meer et al. 2022). In this paper, we follow a completely different training approach. Our approach is based on pre-training a multilayer perceptron with $\mathcal{L}_{\mathcal{B}}$ as the loss function, fine-tuning the first model (by loading the weights of the model pre-trained with the BI-loss) with $\mathcal{L}_{\mathcal{F}}$ as the loss function, and finally fine-tuning the second model (by loading the weights of the model fine-tuned by the PI-loss) with $\mathcal{L}_{\mathcal{D}}$ as the loss function. A relatively similar training approach for PINN has been employed in Moseley et al. (2020) to solve a direct problem for the wave equation.

Apart from our training method, our activation functions are also different from the typical activation functions used in PINN. Traditionally, certain off-the-shelf activation functions such as hyperbolic tangent are used for training PINN (Raissi et al. 2019). Adaptive activation functions where an additional hyperparameter is used to adapt a well-known activation function to a problem, e.g. $\tanh(az)$ instead of $\tanh(z)$, have also been considered in the literature (Jagtap et al. 2020a, b). In general, the must-have properties of an activation function are non-linearity and continuous differentiability. Non-linearity is essential because a neural network with linear activation functions is a linear function. Similarly, some level of differentiability (ideally, continuous differentiability) is required as calculation of gradients, usually through automatic differentiation (AD) (Baydin et al. 2017), is needed for training a neural network. In addition to the must-have properties, various nice-to-have properties are suggested by different authors such as monotonicity and approximation of the identity function near the origin (Murray et al. 2022). Recently, periodic activation functions such as sinusoidal representation networks or SIRENs (Meronen et al. 2021; Sitzmann 2020) are introduced to represent a signal's spatial and temporal derivatives given that these derivatives are essential to many physical signals defined implicitly as the solution to partial differential equations (Sitzmann 2020). Since the initial condition of a PDE can be viewed as a snapshot of its exact solution at $t = 0$, we decided to use the function of the initial condition, up to a constant, as our activation function. Of course, this option might not always be available because an activation function should at least possess the must-have properties, i.e. nonlinearity and continuous differentiability. If the function of the initial condition is not nonlinear or continuously differentiable, we suggest using a periodic activation function following the recommendation of Sitzmann (2020).

In this article, we propose a hybrid method based on the application of the Ritz collocation method and neural networks to solve the inverse problem given by equations (1.1)–(1.4). Our main goal is to obtain approximate solutions with high accuracy and low computational cost. In addition, when dealing with perturbed input data, a regularization method is used to obtain stable numerical derivatives. The rest of the paper is organized as follows. In Sect. 2, we discuss the theoretical background including the reconstruction algorithm. In Sect. 3, we discuss the numerical implementation of the proposed technique. In Sect. 4, we summarize the results and conclude the paper.

2 Approximation method

We define the following,

$$G_\omega(x) := \int_0^T \omega(t)g(x,t)dt, \quad H_\omega(x) := \int_0^T \omega(t)h(x,t)dt. \tag{2.1}$$

In addition to the conditions of the existence of the solution (Lesnic et al. 2016), we further assume that

$$\forall\, x \in\ [0,L],\ G_\omega(x) \geq constant > 0, \quad u_T(x) \in C^2[0,L]. \tag{2.2}$$

By using the extra condition (1.4) we get the following,

$$f(x) = \frac{\int_0^T \omega(t)u_{tt}(x,t)dt - H_\omega(x) - u_T''(x)}{G_\omega(x)}. \tag{2.3}$$

Thus, the main problem is reformulated as the following PDE

$$\begin{aligned} u_{tt}(x,t) - u_{xx}(x,t) = g(x,t)&\frac{\int_0^T \omega(t)u_{tt}(x,t)dt - H_\omega(x) - u_T''(x)}{G_\omega(x)} \\ &+h(x,t), \quad in \quad (0,L)\times(0,T), \end{aligned} \tag{2.4}$$

along with the initial and boundary conditions (1.2)–(1.3). Next, we aim to construct an auxiliary function, the so-called satisfier function which fulfills all the initial and boundary conditions (1.2)–(1.3) exactly (Rashedi 2021, 2022a, b). By applying the interpolation techniques and supposing that the initial and boundary conditions are continuous in corners, i.e. the consistency conditions (1.6)–(1.7) hold, the satisfier function $S(x,t)$ is obtained as follows

$$A_1(x,t) = u_0(x) + tu_1(x), \quad A_2(x,t) = b_0(t) + \frac{x}{L}(b_1(t) - b_0(t)), \tag{2.5}$$

$$S(x,t) := A_1(x,t) + A_2(x,t) - \left(A_2(x,0) + t\left.\frac{\partial A_2(x,t)}{\partial t}\right|_{t=0}\right) \tag{2.6}$$

$$\begin{aligned} &= u_0(x) + t\{u_1(x) - b_0'(0)\} + b_0(t) - b_0(0) \\ &\quad + \frac{x}{L}\left(b_1(t) - b_0(t) - b_1(0) + b_0(0) - tb_1'(0) + tb_0'(0)\right). \end{aligned} \tag{2.7}$$

Then, denoting the well-known Legendre polynomials of degree m which are defined over the interval $[-1,1]$ as $l_m(z)$ and can be determined utilizing the following recurrence formula,

$$l_0(z) = 1,\ l_1(z) = z,\ l_{m+1}(z) = \frac{2m+1}{m+1}zl_m(z) - \frac{m}{m+1}l_{m-1}(z),\ m \in \mathbb{N}, \tag{2.8}$$

we define $\phi_i^*(x) := l_i(\frac{2x}{L} - 1)$ as the shifted Legendre polynomial of degree i in the interval $[0,L]$ and $\psi_j^*(t) := l_j(\frac{2t}{T} - 1)$ as the shifted Legendre polynomial of degree j in the interval $[0,T]$. Therefore, the orthonormal Legendre basis functions (OLBFs) corresponding to the intervals $[0,L]$ and $[0,T]$, respectively can be obtained as

$$\phi_i(x) = \frac{\phi_i^*(x)}{\sqrt{\int_0^L (\phi_i^*(x))^2dx}}, \quad \psi_i(t) = \frac{\psi_i^*(t)}{\sqrt{\int_0^T (\psi_i^*(t))^2dt}}. \tag{2.9}$$

We consider the following vectors of the OLBFs[1]

$$\phi(x)=[\phi_0(x),\phi_1(x),...,\phi_N(x)]^{tr},\quad \psi(t)=[\psi_0(t),\psi_1(t),...,\psi_{N'}(t)]^{tr},\qquad(2.10)$$

and introduce the integration and differentiation of the vectors $\phi(x)$ and $\psi(t)$ as follows

$$\int_0^x \phi(z)dz \simeq P_{N,L}\phi(x),\quad \frac{d}{dx}\phi(x)=D_{N,L}\phi(x),\quad 0\le x\le L,\qquad(2.11)$$

$$\int_0^t \psi(z)dz \simeq P_{N',T}\psi(t),\quad \frac{d}{dt}\psi(t)=D_{N',T}\psi(t),\quad 0\le t\le T,\qquad(2.12)$$

where $P_{N,L}$ and $D_{N,L}$ are called the $(N+1)\times(N+1)$ operational matrices of integration and differentiation (Rashedi 2021, 2022a, b) corresponding to the basis functions $\phi_i(x)$. Similarly, $P_{N',T}$ and $D_{N',T}$ are called the $(N'+1)\times(N'+1)$ operational matrices of integration and differentiation corresponding to the basis functions $\psi_i(t)$. Denoting the entries of the matrices $P_{N,L},\ P_{N',T}$ by $p_{ij}^{N,L},\ p_{ij}^{N',T}$, respectively, we can get

$$\begin{aligned} p_{ij}^{N,L} &= \int_0^L R_i^{\phi}(x)\phi_j(x)dx,\ i,\ j=0,...,N,\\ p_{ij}^{N',T} &= \int_0^T R_i^{\psi}(t)\psi_j(t)dt,\ i,\ j=0,...,N',\end{aligned}\qquad(2.13)$$

where

$$R_i^{\phi}(x):=\int_0^x \phi_i(s)ds,\quad R_i^{\psi}(t):=\int_0^t \psi_i(s)ds.\qquad(2.14)$$

Furthermore, utilizing the properties of the Legendre polynomials (Bell 2004; Hochstadt 1986), the following relations are obtained

$$\phi_i'(x)=\frac{2\sqrt{2i+1}}{L}\Big(\sqrt{2i-1}\phi_{i-1}(x)+\sqrt{2i-5}\phi_{i-3}(x)+\sqrt{2i-9}\phi_{i-5}(x)+...\qquad(2.15)$$

$$\psi_i'(t)=\frac{2\sqrt{2i+1}}{T}\Big(\sqrt{2i-1}\psi_{i-1}(t)+\sqrt{2i-5}\psi_{i-3}(t)+\sqrt{2i-9}\psi_{i-5}(t)+...\qquad(2.16)$$

which can be used directly to calculate the entries of the matrices $D_{N,L}$ and $D_{N',T}$. For example, $D_{N,L}$ and $P_{N,L}$ with $N=9$ are presented as follows

$$D_{9,L}=\begin{pmatrix}
0&0&0&0&0&0&0&0&0&0\\
\frac{2\sqrt{3}}{L}&0&0&0&0&0&0&0&0&0\\
0&\frac{2\sqrt{15}}{L}&0&0&0&0&0&0&0&0\\
\frac{2\sqrt{7}}{L}&0&\frac{2\sqrt{35}}{L}&0&0&0&0&0&0&0\\
0&\frac{6\sqrt{3}}{L}&0&\frac{6\sqrt{7}}{L}&0&0&0&0&0&0\\
\frac{2\sqrt{11}}{L}&0&\frac{2\sqrt{55}}{L}&0&\frac{6\sqrt{11}}{L}&0&0&0&0&0\\
0&\frac{2\sqrt{39}}{L}&0&\frac{2\sqrt{91}}{L}&0&\frac{2\sqrt{143}}{L}&0&0&0&0\\
\frac{2\sqrt{15}}{L}&0&\frac{10\sqrt{3}}{L}&0&\frac{6\sqrt{15}}{L}&0&\frac{2\sqrt{195}}{L}&0&0&0\\
0&\frac{2\sqrt{51}}{L}&0&\frac{2\sqrt{119}}{L}&0&\frac{2\sqrt{187}}{L}&0&\frac{2\sqrt{255}}{L}&0&0\\
\frac{2\sqrt{19}}{L}&0&\frac{2\sqrt{95}}{L}&0&\frac{6\sqrt{19}}{L}&0&\frac{2\sqrt{247}}{L}&0&\frac{2\sqrt{323}}{L}&0
\end{pmatrix},\qquad(2.17)$$

[1] In this work we represent the transpose operator by notation tr.

$$P_{9,L}=\begin{pmatrix} \frac{L}{2} & \frac{L}{2\sqrt{3}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ \frac{-L}{2\sqrt{3}} & 0 & \frac{L}{2\sqrt{15}} & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & \frac{-L}{2\sqrt{15}} & 0 & \frac{L}{2\sqrt{35}} & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & \frac{-L}{2\sqrt{35}} & 0 & \frac{L}{6\sqrt{7}} & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & \frac{-L}{6\sqrt{7}} & 0 & \frac{L}{6\sqrt{11}} & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & \frac{-L}{6\sqrt{11}} & 0 & \frac{L}{2\sqrt{143}} & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & \frac{-L}{2\sqrt{143}} & 0 & \frac{L}{2\sqrt{195}} & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{-L}{2\sqrt{195}} & 0 & \frac{L}{2\sqrt{255}} & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{-L}{2\sqrt{255}} & 0 & \frac{L}{2\sqrt{323}}\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{-L}{2\sqrt{323}} & 0 \end{pmatrix}. \tag{2.18}$$

The Ritz approximation $u_{N,N'}(x,t)$, for the unknown function $u(x,t)$, based on the OLBFs is sought in the form of the following truncated series

$$\begin{aligned} u_{N,N'}(x,t) &:= t^2x(x-L)\phi^{tr}(x)C\psi(t)+S(x,t)\\ &=\sum_{i=0}^{N}\sum_{j=0}^{N'}c_{ij}t^2x(x-L)\phi_i(x)\psi_j(t)+S(x,t), \end{aligned} \tag{2.19}$$

where the unknown matrix C is given by

$$C=\begin{pmatrix} c_{00} & \cdots & c_{0N'}\\ \vdots & & \vdots\\ c_{N0} & \cdots & c_{NN'} \end{pmatrix}. \tag{2.20}$$

Obviously, the approximation $u_{N,N'}(x,t)$ given by Eq. (2.19) fulfills the initial and boundary conditions (1.2)–(1.3). Nevertheless, we call (2.19) as the Ritz approximate solution of the system of equations (2.4) and (1.2)–(1.3) if $u_{N,N'}(x,t)$ also approximately satisfies the following residual function defined over $(x,t)\in[0,L]\times[0,T]$

$$\begin{aligned} R(x,t,u(x,t)) &:= u_{tt}(x,t)-u_{xx}(x,t)\\ &\quad -g(x,t)\frac{\int_0^T\omega(t)u_{tt}(x,t)dt-H_\omega(x)-u''_T(x)}{G_\omega(x)}-h(x,t)=0. \end{aligned} \tag{2.21}$$

In this respect, by taking (2.19) into account and using the operational matrices of differentiation $D_{N,L}$ and $D_{N',T}$, we calculate the approximations of $u_{tt}(x,t)$ and $u_{xx}(x,t)$ based on OLBFs as

$$\begin{aligned} u_{tt}(x,t) &\simeq x(x-L)\phi^{tr}(x)C\Big(2\psi(t)+4tD_{N',T}\psi(t)+t^2(D_{N',T})^2\psi(t)\Big)\\ &\quad +b''_0(t)+\frac{x}{L}(b''_1(t)-b''_0(t)), \end{aligned} \tag{2.22}$$

$$\begin{aligned} u_{xx}(x,t) &\simeq t^2\Big(2\phi^{tr}(x)+(4x-2L)\phi^{tr}(x)D^{tr}_{N,L}\\ &\quad +x(x-L)\phi^{tr}(x)(D^{tr}_{N,L})^2\Big)C\psi(t)+u''_0(x)+tu''_1(x). \end{aligned} \tag{2.23}$$

Meanwhile, if we define the column vector α and the function $\beta(x)$ as

$$\begin{aligned}\alpha &:= \int_0^T \omega(t)\{2\psi(t)+4tD_{N',T}\psi(t)+t^2(D_{N',T})^2\psi(t)\}dt,\\ \beta(x) &= \int_0^T \omega(t)\{b_0''(t)+\frac{x}{L}(b_1''(t)-b_0''(t))\}dt,\end{aligned} \tag{2.24}$$

then we get

$$\int_0^T \omega(t)u_{tt}(x,t)dt \simeq x(x-L)\phi^{tr}(x)C\alpha+\beta(x). \tag{2.25}$$

Finally, by substituting the approximations (2.22)–(2.25) in Eq. (2.21) we have

$$\begin{aligned}R(x,t,u_{N,N'}(x,t)) \simeq{}& x(x-L)\phi^{tr}(x)C\Big(2\psi(t)+4tD_{N',T}\psi(t)+t^2(D_{N',T})^2\psi(t)\Big)\\ &+b_0''(t)+\frac{x}{L}(b_1''(t)-b_0''(t))\\ &-t^2\Big(2\phi^{tr}(x)+(4x-2L)\phi^{tr}(x)D_{N,L}^{tr}+x(x-L)\phi^{tr}(x)(D_{N,L}^{tr})^2\Big)\\ &\times C\psi(t)-u_0''(x)-tu_1''(x)\\ &-\frac{g(x,t)}{G_\omega(x)}\Big(x(x-L)\phi^{tr}(x)C\alpha+\beta(x)\\ &-H_\omega(x)-u_T''(x)\Big)-h(x,t)=0.\end{aligned} \tag{2.26}$$

By collocating the residual function (2.26) at the points

$$x_i=\frac{iL}{N+2},\ t_j=\frac{jT}{N'+2},\ i=1,...,N+1,\ j=1,...,N'+1, \tag{2.27}$$

we obtain the following system of algebraic equations

$$R(x_i,t_j,u_{N,N'}(x_i,t_j))=0,\ i=1,...,N+1,\ j=1,...,N'+1. \tag{2.28}$$

By solving the system of linear equations (2.28) for the elements c_{ij} utilizing the Newton's iterative method (Stoer and Bulirsch 1980), the unknown matrix C and approximation (2.19) are specified. It should be noted that the approximation constructed by Eq. (2.19)–(2.28) is valid as long as the input initial and boundary data of the problem are free of errors. Otherwise, appropriate instructions should be adopted so that the errors in the input data are controlled. To do so, we apply a numerical differentiation method to find stable numerical derivatives of the perturbed data. Thus, in the case of inaccurate boundary data, assume $u_T^\sigma(x)$ is perturbed subject to

$$\frac{1}{M}\sum_{k=1}^{M}(u_T(x_k)-u_T^\sigma(x_k))^2 \le \sigma^2,\quad x_k\in[0,L],\ M\in\mathbb{N}. \tag{2.29}$$

It has been shown that the solution (Wei and Li 2006) of the following minimization problem

$$\begin{aligned}\min_{u_T\in\Gamma_3}\Omega(u_T) &= \frac{1}{M}\sum_{r=1}^{M}(u_T(x_r)-u_T^\sigma(x_r))^2+\mu^*\|u_T'''\|_{L^2(\mathbb{R})}\ s.t\ \Gamma_3\\ &=\{u_T|u_T\in C^2(\mathbb{R}),\ u_T'''\in L^2(\mathbb{R})\},\end{aligned} \tag{2.30}$$

is given by

$$u_T^{\mu^*}(x) = \sum_{j=1}^{M} \Theta_j |x - x_j|^5 + \sum_{j=1}^{3} d_j x^{j-1}, \tag{2.31}$$

where the coefficients $\{\Theta_j\}_{j=1}^M$ and $\{d_j\}_{j=1}^3$ satisfy the following system of equations

$$u_T^{\mu^*}(x_i) - 240\mu^* M \Theta_i - u_T^{\sigma}(x_i) = 0, \quad i = 1, ..., M, \tag{2.32}$$

$$\sum_{j=1}^{3} d_j x_j^i = 0, \quad i = 0, 1, 2, \tag{2.33}$$

and μ^* is the regularization parameter that can be chosen by either a priori rule which takes $\mu^* = \sigma^2$ or employing the Morozov's discrepancy principle to get μ^* from the following equation

$$\frac{1}{M} \sum_{k=1}^{M} (u_T^{\mu^*}(x_k) - u_T^{\sigma}(x_k))^2 = \sigma^2. \tag{2.34}$$

Thus, in equation (2.3) we employ $\frac{d^2}{dx^2}\left(u_T^{\mu^*}(x)\right)$ instead of $u_T^{''}(x)$.

As mentioned in the introduction, the continuous time model physics informed neural network (CTM-PINN) (Raissi et al. 2019) typically comprises of two loss terms. The boundary and initial loss function, $\mathcal{L}_{\mathcal{B}}$ or BI-loss, and the physics informed loss function, $\mathcal{L}_{\mathcal{F}}$ or PI-loss. Moreover, the aforementioned loss functions are added together to provide a single loss function to train a CTM-PINN.

In this paper, we use the Ritz approximation to achieve two main objectives. The first objective is to convert the inverse problem to an integro-differential equation and provide an approximation for the force function which makes the definition of a PI-loss function possible. For the problem given by equations (1.1)–(1.4), the PI-loss function $\mathcal{L}_{\mathcal{F}}$ is defined as follows

$$\text{PI-loss} = \frac{1}{n} \sum_{i=1}^{n} (u_{tt}(x_i, t_i) - u_{xx}(x_i, t_i) - f(x_i) g(x_i, t_i) - h(x_i, t_i))^2, \tag{2.35}$$

where (x_i, t_i) are collocation points. The PI-loss function cannot be used directly as it contains an unknown function $f(x_i)$. Nevertheless, the relations (2.3) and (2.25) allow us to replace the unknown $f(x)$ with an approximation, namely

$$\tilde{f}(x) = \frac{\left(x(x-L)\phi^T(x) C\alpha + \beta(x) - H_\omega(x) - u_T^{''}(x)\right)}{G_\omega(x)}. \tag{2.36}$$

The second objective is to decrease the error of the neural network approximator. To decrease the error, a third loss function $\mathcal{L}_{\mathcal{D}}$ is defined which is the mean square error (MSE) between the Ritz approximation and the neural network output. We call this loss function, data loss or D-loss for short. In summary, instead of defining our loss function by $\mathcal{L}_{\mathcal{B}} + \mathcal{L}_{\mathcal{F}}$ and training our neural network in a single step, we consider three loss functions, i.e. BI-loss, PI-loss, D-loss, and train our neural network with each loss function separately in three steps. In Algorithm 1, we gave an overview of the steps involved in our solution.

Algorithm 1 A step-by-step overview of the solution

```
1) Convert the inverse problem to an integro-differential equation given by (2.4)
2) Convert the integro-differential equation to a linear system of algebraic equations using the Ritz collocation method and solve the system
3) Train a multilayer perceptron using a boundary and initial loss function $\mathcal{L}_{\mathcal{B}}$
4) Fine-tune the multilayer perceptron of Step 3 using a physics-informed loss function $\mathcal{L}_{\mathcal{F}}$ where the unknown function $f(x)$ is approximated by (2.36)
5) Fine-tune the multilayer perceptron of Step 4 using a data loss function $\mathcal{L}_{\mathcal{D}}$ where the data comes from the Ritz approximation in Step 2
```

By not adding the three loss functions together as in CTM-PINN, we can have a disentangled representation where the contribution of each loss function can be analyzed separately and the loss functions do not impact each other negatively. The proposed three-step training strategy also provides more flexibility. As shown in Sect. 3, it allows us to choose separate hyperparameters such as learning rate and number of epochs for each training step.

Another difference between this work and CTM-PINN is the choice of activation functions. If we knew the exact solution, we could design the activation functions by taking into account the nonlinear functions that appear in the exact solution. In practical applications of PINN, the exact solution is not known. Nevertheless, some nonlinear and continuously differentiable functions might be present in the boundary and initial conditions of the problem. As discussed in the introduction, in the absence of such functions in the boundary and initial conditions, we recommend using some periodic functions such as sine or cosine.

3 Numerical experiments

In this section, we evaluate the proposed method on two examples. Moreover, the second example has two variations: i) without any noise, and ii) with the noise level of $\sigma = 0.002$ where σ is defined in (2.29). Throughout this section, MSE, and RMSE stand for mean squared error, and root mean squared error respectively. The code for the experiments is written in Python 3.10.6, and PyTorch 1.12.1 is used as the machine learning library for neural network training and automatic differentiation.

In all examples, the neural network has one hidden layer, the output layer has one neuron with an identity activation function, and the input layer has two neurons, one for x and another for t. We generate three datasets, namely a training set, a validation set, and a test set. For each dataset, the intervals where points are sampled from are as follows, $\text{TRI}_x = [0+\epsilon, 1-\epsilon]$, $\text{TRI}_t = [0+\epsilon, 0.8]$, $\text{VAI}_x = [0+\epsilon, 1-\epsilon]$, $\text{VAI}_t = [0.8+\epsilon, 0.825]$, $\text{TEI}_x = [0+\epsilon, 1-\epsilon]$, $\text{TEI}_t = [0.825+\epsilon, 0.85]$ where TRI stands for training interval, VAI stands for validation interval, TEI stands for test interval, and ϵ is set to 0.001. We generate 250,000 equidistant initial training points from TRI_x and 250,000 equidistant boundary training points from TRI_t. We generate collocation points by Latin hypercube sampling. In particular, we generate 500,000 collocation training points from $(\text{TRI}_x, \text{TRI}_t)$, 50,000 collocation validation points from $(\text{VAI}_x, \text{VAI}_t)$, and 25000 collocation test points from $(\text{TEI}_x, \text{TEI}_t)$.

In all examples of D-PINN, the training consists of three steps:

1. training the neural network with the training boundary and initial points where the loss function $\mathcal{L}_{\mathcal{B}}$ is the MSE between the output of the neural network and the values given by the boundary and initial conditions,
2. training the neural network with the collocation training points where the loss is the physics informed loss function $\mathcal{L}_{\mathcal{F}}$, c.f. (2.35), and
3. training the neural network with the collocation training points where the loss function $\mathcal{L}_{\mathcal{D}}$ is the MSE between the output of the neural network and the Ritz approximation values.

The validation data is only used in step 3 for learning rate scheduling and saving the best model to the memory to be used for testing. More precisely, we set the patience value to 2 and the factor to 0.1 in learning rate scheduling, i.e. the learning rate is multiplied by 0.1 if the validation RMSE does not decrease after two epochs. The best model is the model that has the smallest validation RMSE value.

In all steps, the optimization algorithm is RMSProp. In step 1 and 2, RMSProp has the default parameters in PyTorch 1.12.1. In step 3, the smoothing constant α is set to 0.98, and ϵ is set to 1e-07. The number of epochs and the learning rate for step 1 and 2 are set to 1 and 0.0001 respectively. The experiments are conducted on a laptop with Intel(R) Core(TM) i5-1145G7 CPU, no CUDA-Enabled GPU, and 32 GB of RAM.

Example 1

As the first example, consider the problem given by equations (1.1)–(1.4) with the following properties,

$$u_0(x) = u_1(x) = e^x \cos(x),\ u_T(x) = e^{1+x}\cos(x),\ x \in [0, 1], \tag{3.1}$$

$$\omega(t) = 1 + t,\ b_0(t) = e^t,\ b_1(t) = \cos(1)e^{1+t},\ t \in (0, 1), \tag{3.2}$$

$$g(x, t) = e^{x+t},\ h(x, t) = 0, \quad (x, t) \in (0, 1) \times (0, 1), \tag{3.3}$$

i.e. the Eq. (1.1) is $u_{tt}(x, t) - u_{xx}(x, t) = f(x)e^{x+t}$. The exact solution of the problem is as follows,

$$u(x, t) = \cos(x)e^{x+t},\ f(x) = \cos(x) + 2\sin(x), \quad (x, t) \in (0, 1) \times (0, 1). \tag{3.4}$$

By taking the exact initial and boundary conditions, we solve the problem using the numerical technique presented in Sect. 2, with $N = N' = 2$. In this example, the hidden layer has 32 neurons, and the activation function of the hidden layer is $u(x, 0) = e^x \cos(x)$. In all steps, batch size is set to 32. The number of epochs and the learning rate for step 3 of training are 30 and 0.001 respectively.

The results are shown in Table 1 and Figs. 1, 2 and 3. The loss function in PINN without $\mathcal{L}_{\mathcal{D}}$ is $\mathcal{L}_{\mathcal{B}}$ + $\mathcal{L}_{\mathcal{F}}$ and the activation function is tanh. The loss function in PINN with $\mathcal{L}_{\mathcal{D}}$ is $\mathcal{L}_{\mathcal{B}}$ + $\mathcal{L}_{\mathcal{F}}$+$\mathcal{L}_{\mathcal{D}}$ and the activation function is tanh. In PINN experiments with or without $\mathcal{L}_{\mathcal{D}}$, learning rate scheduling and validation data are not used, batch size is the total training data, the number of epochs is 30, the learning rate is 0.0001, and the optimization algorithm is RMSProp with the default parameters in PyTorch 1.12.1.

As it can be seen in Table 1 and Figs. 1 and 2, the proposed neural network method D-PINN produces a more accurate numerical approximation, i.e. it is closer to the exact solution than the Ritz method on the test set. More precisely, in the region $\mathrm{TEI}_x \times \mathrm{TEI}_t$, D-PINN is 32% more accurate than the Ritz method in terms of MSE and 18% more accurate than the Ritz

Table 1 This table compares PINN (with (2.36) as the approximation of $f(x)$, but without $\mathcal{L}_{\mathcal{D}}$), PINN with $\mathcal{L}_{\mathcal{D}}$, D-PINN without $\mathcal{L}_{\mathcal{D}}$, D-PINN with tanh instead of $e^x\cos(x)$, D-PINN without $\mathcal{L}_{\mathcal{F}}$, D-PINN without $\mathcal{L}_{\mathcal{B}}$, Ritz, and D-PINN on the test set

Method	MSE	RMSE
PINN	2.91990465	1.70877285
PINN with $\mathcal{L}_{\mathcal{D}}$	2.53943817	1.59356147
D-PINN without $\mathcal{L}_{\mathcal{D}}$	1.22036661	1.10470205
D-PINN with tanh	0.00014811	0.01217014
D-PINN without $\mathcal{L}_{\mathcal{F}}$	0.00004136	0.00643130
D-PINN without $\mathcal{L}_{\mathcal{B}}$	0.00001561	0.00395047
Ritz	0.00001094	0.00330777
D-PINN	0.00000743	0.00272654

In each experiment, we calculate the MSE and RMSE between the approximate solution and the exact solution. This ablation study demonstrates the contribution of each component in D-PINN

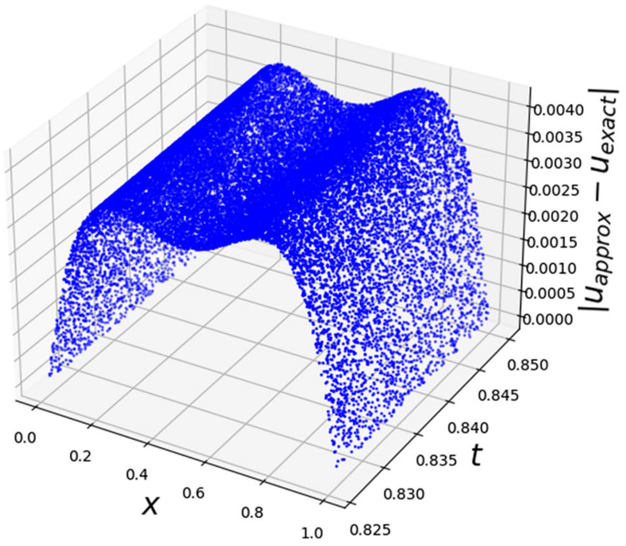

Fig. 1 The difference between Ritz and the exact solution for Example 1

method in terms of RMSE. In Fig. 3, the difference between approximation of $f(x)$ by Ritz and $f(x)$ is shown.

To investigate the impact of the number of hidden layers on the approximation error, we added more hidden layers, each with 32 neurons, to D-PINN for Example 1. As it can be seen in Table 2, adding more hidden layers degrades the performance.

Example 2

As the second example (Lesnic et al. 2016), consider the problem given by equations (1.1)-(1.4) with the following properties:

$$u_0(x) = 2\sin(\pi x),\ u_1(x) = 0,\ u_T(x) = \sin(\pi x),\ x \in [0, 1], \tag{3.5}$$

$$\omega(t) = 1,\ b_0(t) = b_1(t) = 0,\ t \in (0, 1), \tag{3.6}$$

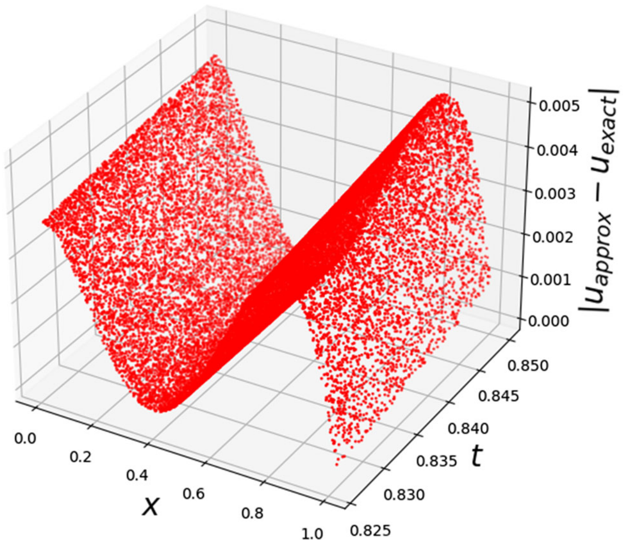

Fig. 2 The difference between D-PINN and the exact solution for Example 1

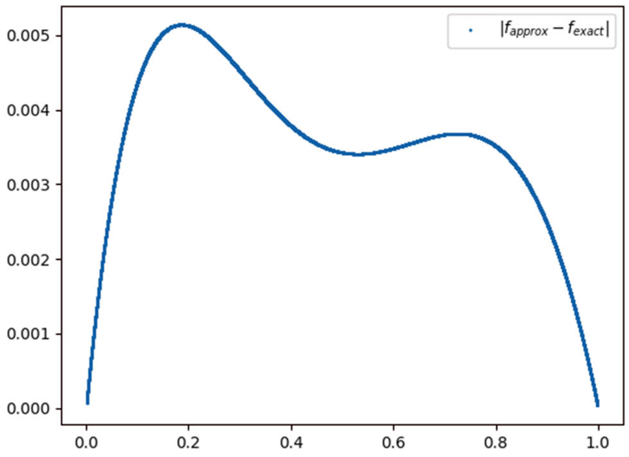

Fig. 3 The difference between $f(x)$ and its approximation by Ritz for Example 1

$$g(x,t) = 1, \; h(x,t) = 0, \quad (x,t) \in (0,1) \times (0,1), \tag{3.7}$$

i.e. the Eq. (1.1) is $u_{tt}(x,t) - u_{xx}(x,t) = f(x)$. The exact solution of the problem is as follows,

$$u(x,t) = \sin(\pi x)(1+\cos(\pi t)), \quad f(x) = \pi^2 \sin(\pi x), \quad (x,t) \in (0,1) \times (0,1). \tag{3.8}$$

We solve the problem by applying the numerical scheme proposed in Sect. 2 in the presence of exact initial and boundary data with $N = N' = 4$. In this example and its noisy variation, the hidden layer has 16 neurons, batch size is set to 16 in all training steps, and the activation function of the hidden layer is $\frac{u(x,0)}{2} = \sin(\pi x)$. To ensure that the absolute value of the range of the activation function is less than 1, given that the domain is $(0,1) \times (0,1)$, we

Table 2 This table shows the impact of adding more hidden layers to D-PINN for Example 1

Method	MSE	RMSE
D-PINN with one hidden layer	0.00000743	0.00272654
D-PINN with two hidden layers	0.00105702	0.03251189
D-PINN with three hidden layers	0.00016358	0.01279003

The MSE and RMSE are calculated between the approximate solution and the exact solution

Table 3 This table shows the comparison between the Ritz approximation and D-PINN approximation for Example 2 without noise

Method	MSE	RMSE
Ritz	0.00000841	0.00290061
D-PINN	0.00000790	0.00281021

The MSE and RMSE are calculated between the approximate solution and the exact solution

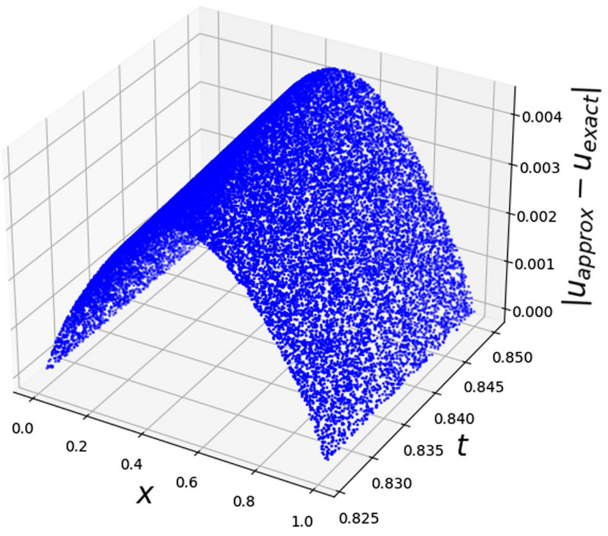

Fig. 4 The difference between Ritz and the exact solution for Example 2 without noise

divided $u(x, 0)$ by two. In the absence of noise, the number of epochs and the learning rate for step 3 of training are 8 and 0.01 respectively. In the presence of noise, the number of epochs and the learning rate for step 3 of training are 10 and 0.01 respectively.

As it can be seen in Table 3 and Figs. 4 and 5, the proposed neural network method D-PINN produces a more accurate numerical approximation than Ritz on the test set. In Table 3, in the region $\mathrm{TEI}_x \times \mathrm{TEI}_t$, D-PINN is 6% more accurate than the Ritz method in terms of MSE and 3% more accurate than the Ritz method in terms of RMSE. In Fig. 6, the difference between approximation of $f(x)$ by Ritz and $f(x)$ is shown.

To investigate the issue of numerical stability of the approximate solution with respect to the small perturbations of the input boundary data, we use the following rule (Wen et al. 2013)

$$u_T^{\sigma}(x_i) = u_T(x_i)(1 + \sigma \Pi_i), \quad \sigma = r \times 10^{-3}, \; r \in \mathbb{R}, \tag{3.9}$$

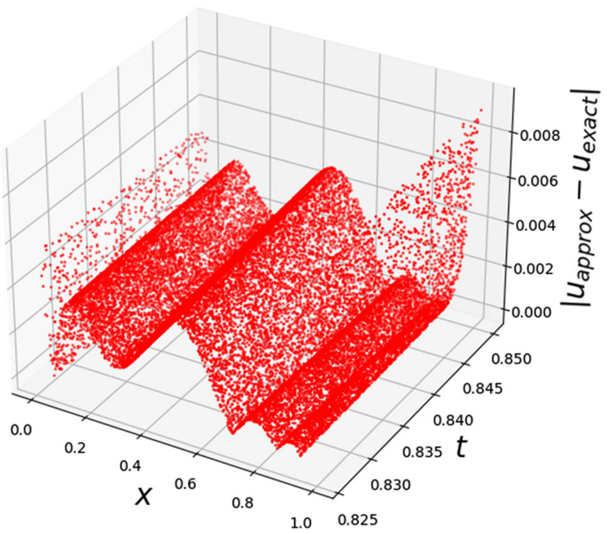

Fig. 5 The difference between D-PINN and the exact solution for Example 2 without noise

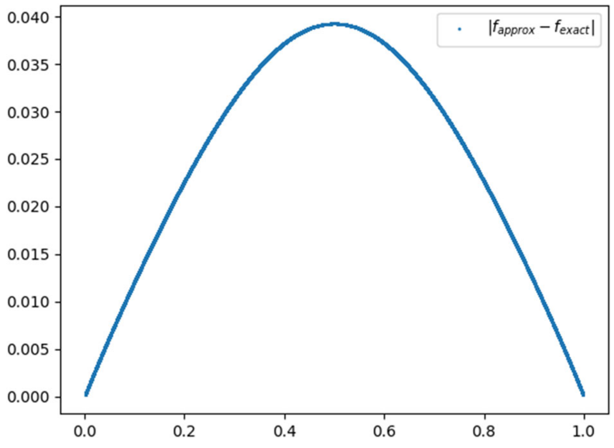

Fig. 6 The difference between $f(x)$ and its approximation by Ritz for Example 2 without noise

to generate artificial errors, where Π_i is a random number uniformly distributed between $[-1, 1]$. We take $r = 2$ and apply the technique presented by Eqs. (2.31)–(2.33) with $\mu^* = 2 \times 10^{-8}$, $M = 40$ to obtain the stable approximation of $(u_T^{\sigma})''(x)$. The outcome of this experiment is presented in Figs. 7, 8 and 9 and Table 4. As it can be seen, our proposed technique performs well in the presence of perturbed input data. In Fig. 9, approximation of $f(x)$ by Ritz and $f(x)$ are shown.

As shown in Table 4, in the region $\text{TEI}_x \times \text{TEI}_t$, the proposed method is 6% more accurate than the Ritz method in terms of MSE and 3% more accurate than the Ritz method in terms of RMSE. The performance gap between PINN with an additive loss function and tanh activation function and D-PINN for Example 2 and its noisy variation is similar to the performance gap in Table 1 of Example 1, but we did not include the results here for brevity.

The training time of D-PINN is shown in Table 5.

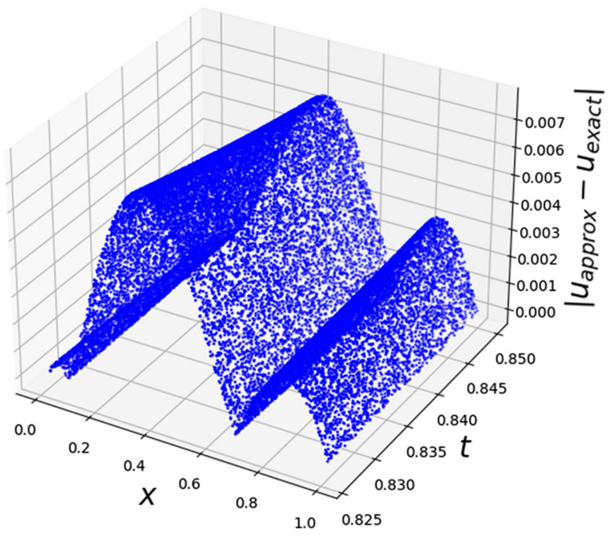

Fig. 7 The difference between Ritz and the exact solution for Example 2 with noise

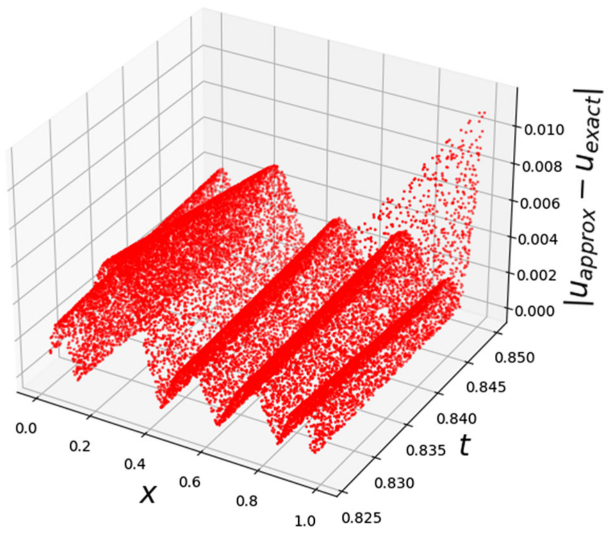

Fig. 8 The difference between D-PINN and the exact solution for Example 2 with noise

Table 4 This table shows the comparison between the Ritz approximation and D-PINN approximation for Example 2 with noise

Method	MSE	RMSE
Ritz	0.00001516	0.00389332
D-PINN	0.00001429	0.00378041

The MSE and RMSE are calculated between the approximate solution and the exact solution

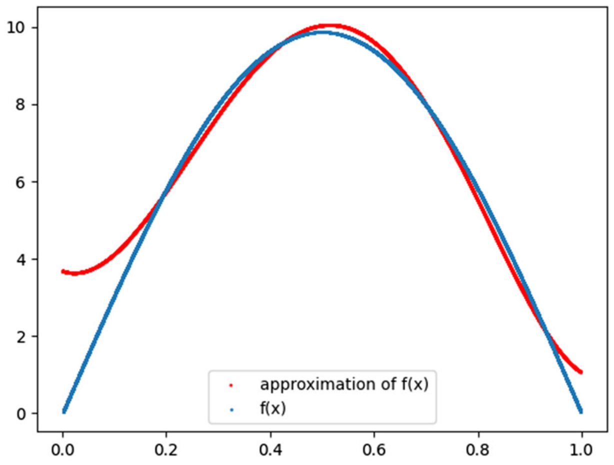

Fig. 9 The function $f(x)$ and its approximation by Ritz for Example 2 with noise

Table 5 This table shows the total training time of D-PINN in second for Example 1, Example 2 without noise and Example 2 with noise

Problem	Training time in second
Example 1	449
Example 2 without noise	199
Example 2 with noise	214

4 Conclusion

In this paper, we provided an accurate and stable solution for the inverse problem of approximating a space-dependent wave source in a one-dimensional wave equation. The solution is based on the combination of the Ritz method and physics-informed neural networks (PINN). The Ritz method is used to convert the inverse problem to an integro-differential equation, provide an approximation for the force function, and provide training and validation data to train a decoupled PINN (D-PINN) with a data loss function. The D-PINN is trained in a novel way where three loss functions are used to train the PINN in three separate stages. Furthermore, validation data is used for learning rate scheduling and avoiding overfitting to the training data. The activation functions of the proposed D-PINN are custom-designed based on the initial conditions of the inverse problems to improve the accuracy of the approximation. The experiments show that D-PINN outperforms Ritz and PINN on the test data and is numerically stable against perturbations in the boundary data.

A distinguishing factor between the present work and the prior art concerning PINN is that the unknown function $f(x)$ of the inverse problems involves integral terms. In contrast to automatic differentiation, deep learning libraries typically cannot perform automatic integration. Therefore, there is no straightforward way to apply PINN to the inverse problems considered in this paper. The proposed D-PINN is a hybrid approach where a classical method such as Ritz is used in conjunction with PINN to solve a class of inverse problems in a novel way.

A limitation of the present work is that the problems considered in the experiments are not high dimensional and the domains are simple. In higher dimensions, achieving accurate results with Ritz approximation requires a large number of bases which is quite computeintensive. We believe that D-PINN shines best in high-dimensional problems because one can employ a less accurate Ritz approximation, i.e. with a fewer number of bases, as the input data for the data loss function, and obtain a more accurate approximation by D-PINN.

Acknowledgements The authors are very grateful to the referees for carefully reading the paper and for their comments and suggestions which have led to improvements of the paper.

Author Contributions AS designed the numerical algorithm, implemented the algorithm, and conducted numerical experiments for the neural network approximation. FB worked on writing the introduction, and contributed to the main methodology (the neural network aspect). KR worked on the Ritz approximation of the inverse wave problems, and contributed to the main methodology (the Ritz aspect) and conducted numerical experiments for the Ritz approximation. All authors reviewed the manuscript.

Funding We confirm that the authors did not receive any funding for this research.

Availability of data and materials The data is generated by a numerical method and it is available.

Declarations

Conflict of interest We confirm that there are no competing interests.

Ethical approval and consent to participate All authors gave their consent for participation. There are no other participants other than the authors.

Consent for publication All authors gave their consent for publication.

Human and animal ethics Not applicable.

References

Baydin AG, Pearlmutter BA, Radul AA, Siskind JM (2017) Automatic differentiation in machine learning: a survey. J Mach Learn Res 18(1):5595–5637

Bell WW (2004) Special functions for scientists and engineers. Dover Publications, New York

Blechschmidt J, Ernst OG (2021) Three ways to solve partial differential equations with neural networks: a review. GAMM Mitt 44(2):e202100006

Cannon JR, Dunninger DR (1970) Determination of an unknown forcing function in a hyperbolic equation from overspecified data. Ann Mat Pura Appl 1:49–62

Chen P, Liu, Aihara K, Chen L (2020) Autoreservoir computing for multistep ahead prediction based on the spatiotemporal information transformation. Nat Commun 11(1):1–15

Cuomo S, Di Cola VS, Giampaolo F, Rozza G, Raissi M, Piccialli F (2022)Scientific machine learning through physics-informed neural networks: where we are and what's next. arXiv preprint arXiv:2201.05624

Engl HW, Scherzer O, Yamamoto M (1994) Uniqueness and stable determination of forcing terms in linear partial differential equations with overspecified boundary data. Inverse Probl 10:1253–1276

Gao H, Sun L, Wang J-X (2021) PhyGeoNet: physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state PDEs on irregular domain. J Comput Phys 428:110079

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

Hajimohammadi Z, Parand K (2021) Numerical learning approximation of time-fractional sub diffusion model on a semi-infinite domain. Chaos Solitons Fractals 142:110435

Hochstadt H (1986) The functions of mathematical physics. Dover Publications Inc., New York

Hussein SO, Lesnic D (2014) Determination of a space-dependent source function in the one-dimensional wave equation. Electron J Bound Elem 12:1–26

Hussein SO, Lesnic D (2016) Determination of forcing functions in the wave equation. Part I: the space-dependent case. J Eng Math 96:115–133

Isakov V (2006) Inverse problems for partial differential equations. Springer, New York

Jagtap AD, Kawaguchi K, Karniadakis GE (2020a) Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. J Comput Phys 404:109136

Jagtap AD, Kawaguchi K, Karniadakis GE (2020b) Locally adaptive activation functions with slope recovery for deep and physics-informed neural networks. Proc R Soc A 476:20200334

Jagtap AD, Kharazmi E, Karniadakis GE (2020c) Conservative physics-informed neural networks on discrete domains for conservation laws: applications to forward and inverse problems. Comput Methods Appl Mech Eng 365:113028

Jagtap AD, Mao Z, Adams N, Karniadakis GE (2022) Physics-informed neural networks for inverse problems in supersonic flows. arXiv preprint arXiv:2202.11821

Jin X, Cai S, Li H, Karniadakis GE (2021) NSFnets (Navier–Stokes flow nets): Physics-informed neural networks for the incompressible Navier–Stokes equations. J Comput Phys 426:109951

Lesnic D, Hussein SO, Johansson BT (2016) Inverse space-dependent force problems for the wave equation. J Comput Appl Math 306:10–39

Mao Z, Jagtap AD, Karniadakis GE (2020) Physics-informed neural networks for high-speed flows. Comput Methods Appl Mech Eng 360:112789

Meer R, Oosterlee CW, Borovykh A (2022) Optimally weighted loss functions for solving pdes with neural networks. J Comput Appl Math 405:113887

Meronen L, Trapp M, Solin A (2021) Periodic activation functions induce stationarity. Adv Neural Inf Process Syst 34:1673–1685

Mishra S, Molinaro R (2022) Estimates on the generalization error of physics-informed neural networks for approximating a class of inverse problems for PDEs. IMA J Numer Anal 42(2):981–1022

Moseley B, Markham A, Meyer TN (2020) Solving the wave equation with physics-informed deep learning. arXiv preprint arXiv:2006.11894

Murray M, Abrol V, Tanner J (2022) Activation function design for deep networks: linearity and effective initialisation. Appl Comput Harmon Anal 59:117–154

Nguyen LH (2019) An inverse space-dependent source problem for hyperbolic equations and the Lipschitz-like convergence of the quasi-reversibility method. Inverse Probl 35(35):035007. https://doi.org/10.1088/1361-6420/aafe8f

Prilepko AI, Orlovsky DG, Vasin IA (2000) Methods for solving inverse problems in mathematical physics. Marcel Dekker Inc, New York

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

Rashedi K (2021) A numerical solution of an inverse diffusion problem based on operational matrices of orthonormal polynomials. Math Method Appl Sci 44:12980–12997

Rashedi K (2022a) Recovery of coefficients of a heat equation by Ritz collocation method. Kuwaut J Sci. https://doi.org/10.48129/kjs.18581

Rashedi K (2022b) A spectral method based on Bernstein orthonormal basis functions for solving an inverse Roseneau equation. Comput Appl Math. https://doi.org/10.1007/s40314-02226401908-0

Rashedi K, Baharifard F, Sarraf A (2022) Stable recovery of a space-dependent force function in a one-dimensional wave equation via Ritz collocation method. J Math Model 10:463–480

Samarskii AA, Vabishchevich AN (2007) Numerical methods for solving inverse problems of mathematical physics. Walter de Gruyter, Berlin

Sitzmann V et al (2020) Implicit neural representations with periodic activation functions. Adv Neural Inf Process Syst 33:7462–7473

Stoer J, Bulirsch R (1980) Introduction to numerical analysis. Springer, New York

Wei T, Li M (2006) High order numerical derivatives for one-dimensional scattered noisy data. Appl Math Comput 175:1744–1759

Wen J, Yamamoto M, Wei T (2013) Simultaneous determination of a time-dependent heat source and the initial temperature in an inverse heat conduction problem. Inverse Probl Sci Eng 21:485–499

Yamamoto M (1995) Stability, reconstruction formula and regularization for an inverse source hyperbolic problem by a control method. Inverse Probl 1:481–496

Yuan L et al (2022) A-PINN: auxiliary physics informed neural networks for forward and inverse problems of nonlinear integro-differential equations. J Comput Phys 462:111260

Zhang D, Lu L, Guo L, Karniadakis GE (2019) Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems. J Comput Phys 397:108850