Journal of Optimization Theory and Applications (2023) 196:391–414
https://doi.org/10.1007/s10957-022-02100-4

Control of Partial Differential Equations via Physics-Informed Neural Networks

Carlos J. García-Cervera[1] · Mathieu Kessler[2] · Francisco Periago[2]

Received: 30 March 2022 / Accepted: 22 August 2022 / Published online: 17 September 2022

Abstract
This paper addresses the numerical resolution of controllability problems for partial differential equations (PDEs) by using physics-informed neural networks. Error estimates for the generalization error for both state and control are derived from classical observability inequalities and energy estimates for the considered PDE. These error bounds, that apply to any exact controllable linear system of PDEs and in any dimension, provide a rigorous justification for the use of neural networks in this field. Preliminary numerical simulation results for three different types of PDEs are carried out to illustrate the performance of the proposed methodology.

Keywords Controllability of partial differential equations · Physics-informed neural networks · Error estimates

Mathematics Subject Classification 93B05 · 93C20 · 65N15

1 Introduction

Since the pioneering theoretical works by Russell [37] and Lions [22], the numerical resolution of controllability problems for PDEs has faced a range of challenging

Communicated by Lorenz Biegler.

✉ Francisco Periago
f.periago@upct.es

Carlos J. García-Cervera
cgarcia@ucsb.edu

Mathieu Kessler
mathieu.kessler@upct.es

[1] Department of Mathematics, University of California, Santa Barbara, CA 93106, USA

[2] Department of Applied Mathematics and Statistics, Technical University of Cartagena, Campus Muralla del Mar, 30202 Cartagena, Murcia, Spain

difficulties which have been solved by using a number of sophisticated techniques (see, e.g., [10, 11, 15, 29, 32, 46], among many others). All these methods require the numerical approximation of some suitable PDEs, a task which is done by using classical methods in numerical analysis, mainly finite differences and finite elements. As a consequence, the available methods for solving numerically controllability problems for PDEs suffer from the well-known *curse of dimensionality* phenomenon. In plain words, curse of dimensionality reflects the fact that doubling the number of degrees of freedom in each spatial direction increases the solution complexity by a factor of 2^d, with d being the spatial dimension. This makes classical numerical methods for solving PDEs impractical when the spatial dimension is large.

On the other hand, during the last few years there has been a deep research effort in developing numerical methods for solving PDEs by using techniques from machine learning (ML) and artificial intelligence (AI). The main motivation for exploring the use of these new techniques in approximating PDEs is not to try to find methods that outperform classical methods (finite differences, finite elements, finite volumes, etc.) in low spatial dimensions ($d = 1, 2, 3$), but to solve numerically high-dimensional PDEs ($d > 3$), where the above-mentioned classical methods get stuck by the curse of dimensionality.

Examples of fields where high-dimensional PDEs arise are, among others: radioactive transport equation ($d \geq 5$), kinetic models, e.g., the Boltzmann kinetic equations ($d = 6$), computational finance, e.g., the nonlinear Black–Scholes equation for pricing derivatives ($d \gg 1$), computational quantum chemistry, e.g., the nonlinear Schrödinger equation in the quantum many-body problem ($d \gg 1$), and game theory, e.g., the Hamilton–Jacobi–Bellman equation in dynamic programming [17]. Control problems for parametric PDEs is another field where high dimension plays a crucial role [20, 25, 26, 47].

Among the different deep-learning-based methods that have been recently proposed to approximate numerically the solution of PDEs, it is worth to mention the following: physics-informed neural networks (PINNs) [34], deep Ritz method [41], methods based on the Feynman–Kac formula [5], and methods based on the solution of backward stochastic differential equations [17]. See also [4, 6] for recent reviews. Although these methods have shown an excellent performance at the level of numerical simulation, the error analysis theory for these methods is essentially missing.

The above deep learning-based numerical schemes can be adapted to solve numerically not only forward problems for PDEs, but also a number of related problems involving PDEs such as inverse problems [27, 34] or random PDEs [43], among others.

Up to the best knowledge of the authors, the numerical resolution of controllability problems for PDEs by using ML has not been addressed so far.

The main goal of this paper is to explore the use of PINNs to approximate numerically the solution of controllability problems for PDEs. More precisely, a PINNs-based algorithm that applies to both linear and nonlinear PDEs is proposed in Sect. 2. Then, fostered by [27], error estimates for the so-called generalization error are provided (Theorem 3.1). The proof of this result is based on energy estimates for the solution of the considered PDE and on observability inequalities for its associated adjoint system. From these error bounds, a convergence result of the control and state obtained by using

the proposed method to the control and state of the continuous problem is established (Corollary 3.1). For the sake of clarity, in this preliminary work we focus on the case of boundary Dirichlet control, but in a straightforward manner the ideas and methods here proposed extend to other types of control actions. Also, for pedagogical reasons, instead of presenting an abstract general framework, the methods and proofs are first described for the two emblematic systems of the wave and heat equations and then extended to more general PDE systems. Preliminary numerical experiments for three different PDEs illustrate the performance of the proposed method. More precisely, the accuracy of the method is tested on a simple model for the wave equation for which an analytical solution is available. In a second experiment, a high-dimensional controllability problem for the heat equation is considered. The third experiment concerns a semilinear PDE.

As the title indicates, this work is just a first step toward the numerical resolution of controllability problems for high-dimensional PDEs and so further research is needed to achieve a deeper understanding of the type of problems considered here.

Finally, for the sake of completeness, it is important to point out that the connection between different architectures of deep neural networks and controlled ordinary differential equations has been recently studied in [9]. This is a novel research line that also includes the analysis of the so-called neural differential equations by using techniques coming from continuous control theory [1, 12, 13, 35, 36].

2 Problem Setup and Description of the PINNs Algorithm

From now on in this paper, $\Omega \subset \mathbb{R}^d$, $d \in \mathbb{N}$, denotes a bounded domain with a smooth boundary which is decomposed into two disjoint parts Γ_D and Γ_C. For any positive time T, we denote by $Q_T := \Omega \times (0, T)$. As is usual $\Delta := \sum_{j=1}^{d} \frac{\partial^2}{\partial x_j^2}$ stands for the Laplacian operator.

2.1 Wave Equation

Given initial data (y^0, y^1) in suitable function spaces, the null controllability problem for the wave equation amounts to finding a positive time $T > 0$ and a control function $u(x, t)$ such that the solution $y(x, t)$ of the system

$$\begin{aligned} &y_{tt} - \Delta y = 0, && \text{in } Q_T \\ &y(x, 0) = y^0(x), && \text{in } \Omega \\ &y_t(x, 0) = y^1(x) && \text{in } \Omega \\ &y(x, t) = 0, && \text{on } \Gamma_D \times (0, T) \\ &y(x, t) = u(x, t) && \text{on } \Gamma_C \times (0, T) \end{aligned} \tag{1}$$

satisfies

$$y(x, T) = y_t(x, T) = 0 \quad x \in \Omega. \tag{2}$$

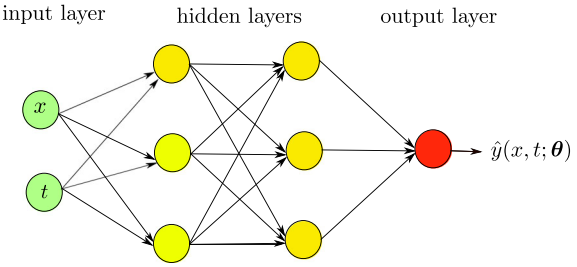

Fig. 1 Illustration of a fully connected deep feedforward neural network with two input channels $\boldsymbol{x} = (x, t)$, two hidden layers (each one with 3 neurons) and a scalar output $\hat{y}(x, t; \boldsymbol{\theta})$. Input data pass through the net by following (3). The set of free parameters of the network is denoted by $\boldsymbol{\theta}$

It is well known [22] that if the domain Ω satisfies the so-called geometrical controllability condition (GCC) introduced by Bardos, Lebeau, and Rauch [3] and if $\left(y^0, y^1\right) \in L^2(\Omega) \times H^{-1}(\Omega)$, then, for T large enough, problem (1)–(2) has a solution $u \in L^2\left(\Gamma_C \times (0, T)\right)$.

The PINNs approach for solving direct and inverse problems for PDEs [34] is next adapted to approximate numerically the control $u(x, t)$ of problem (1)–(2). Roughly speaking, in the PINNs approach the solution is approximated by a neural network and the equations are imposed, in the least square sense, at a collection of nodal points. In the machine learning language, PINNs approach is composed of the following four main steps: (1) design an artificial neural network $\hat{y}\,(x, t; \boldsymbol{\theta})$ as a surrogate of the true solution $y(x, t)$, (2) consider a training set that is used to train the neural network, (3) define an appropriate loss function which accounts for residuals of the PDE, initial, boundary, and final conditions, and (4) train the network by minimizing the cost function defined in the previous step. From the training process, optimal parameters defining the neural network $\hat{y}\,(x, t; \boldsymbol{\theta})$ are computed and eventually are used to get predictions about the state $y(x, t)$ and the control $u(x, t)$, which is approximated as the trace of $\hat{y}\,(x, t; \boldsymbol{\theta})$ on the boundary Γ_C. Next, we give details on these steps:

Step 1: Neural Network Among different possibilities, we consider a deep feedforward neural network (also known in the literature as a multilayer perceptron (MLP)) with $d + 1$ input channels $\boldsymbol{x} = (x, t) \in \mathbb{R}^{d+1}$ and a scalar output $\hat{y}$ (see Fig. 1). More specifically, $\hat{y}\,(x, t; \boldsymbol{\theta})$ is constructed as

$$\begin{aligned}
&\text{input layer:} \quad \mathcal{N}^0(\boldsymbol{x}) = \boldsymbol{x} = (x, t) \in \mathbb{R}^{d+1} \\
&\text{hidden layers:} \ \mathcal{N}^\ell(\boldsymbol{x}) = \sigma\left(\boldsymbol{W}^\ell \mathcal{N}^{\ell-1}(\boldsymbol{x}) + \boldsymbol{b}^\ell\right) \in \mathbb{R}^{N_\ell}, \quad \ell = 1, \cdots, L-1 \\
&\text{output layer:} \quad \hat{y}\,(\boldsymbol{x}; \boldsymbol{\theta}) = \mathcal{N}^L(\boldsymbol{x}) = \boldsymbol{W}^L \mathcal{N}^{L-1}(\boldsymbol{x}) + \boldsymbol{b}^L \in \mathbb{R},
\end{aligned} \tag{3}$$

where

- $\mathcal{N}^\ell : \mathbb{R}^{d_{in}} \to \mathbb{R}^{d_{out}}$ is the ℓ layer with N_ℓ neurons,
- $\boldsymbol{W}^\ell \in \mathbb{R}^{N_\ell \times N_{\ell-1}}$ and $\boldsymbol{b}^\ell \in \mathbb{R}^{N_\ell}$ are, respectively, the weights and biases so that $\boldsymbol{\theta} = \left\{\boldsymbol{W}^\ell, \boldsymbol{b}^\ell\right\}_{1 \le \ell \le L}$ are the parameters of the neural network, and
- σ is an activation function, which acts component-wise. Throughout this paper, we consider smooth activation functions such as hyperbolic tangent $\sigma(s) = \tanh(s)$, with $s \in \mathbb{R}$.

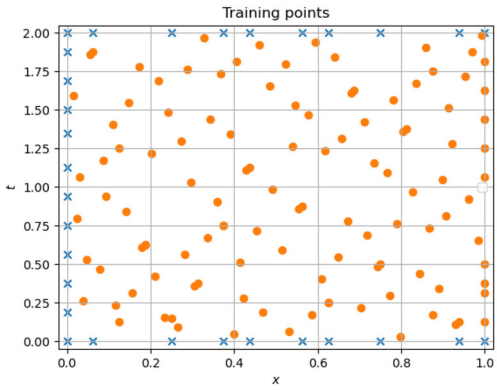

Fig. 2 Illustration of a training dataset (based on Sobol points) in the domain $Q_2 = (0, 1) \times (0, 2)$. Interior points are marked with circles and boundary points in blue color

Step 2: Training Dataset A dataset $\mathcal{T}$ of scattered data is selected in the interior domain $\mathcal{T}_{\text{int}} \subset Q_T$ and on the boundaries $\mathcal{T}_{\Gamma_D} \subset \Gamma_D \times (0, T)$, $\mathcal{T}_{t=0} \subset \Omega \times \{0\}$, $\mathcal{T}_{t=T} \subset \Omega \times \{T\}$. Thus, $\mathcal{T} = \mathcal{T}_{\text{int}} \cup \mathcal{T}_{\Gamma_D} \cup \mathcal{T}_{t=0} \cup \mathcal{T}_{t=T}$ (see Fig. 2 for an illustration). The number of selected points in $\mathcal{T}_{\text{int}}$ is denoted by N_{int}. Analogously, N_b is the number of points on the boundary Γ_D, and N_0 and N_T stand for the number of points in $\mathcal{T}_{t=0}$ and $\mathcal{T}_{t=T}$, respectively. The total number of collocation nodes is denoted by N, and we write $\mathcal{T}_N$ instead of $\mathcal{T}$ to indicate clearly the number of points N used hereafter.

Step 3: Loss Function A weighted summation of the L^2 norm of residuals for the equation, boundary, initial, and final conditions is considered as the loss function to be minimized during the training process. It is composed of the following six terms: given a neural network approximation $\hat{y}$ (as constructed in (3)), define

$$\mathcal{L}_{\text{int}}(\boldsymbol{\theta}; \mathcal{T}_{\text{int}}) = \sum_{j=1}^{N_{\text{int}}} w_{j,\text{int}} |\hat{y}_{tt}(\boldsymbol{x}_j; \boldsymbol{\theta}) - \Delta \hat{y}(\boldsymbol{x}_j; \boldsymbol{\theta})|^2, \quad \boldsymbol{x}_j \in \mathcal{T}_{\text{int}},$$

$$\mathcal{L}_{\Gamma_D}(\boldsymbol{\theta}; \mathcal{T}_{\Gamma_D}) = \sum_{j=1}^{N_b} w_{j,b} |\hat{y}(\boldsymbol{x}_j; \boldsymbol{\theta})|^2, \quad \boldsymbol{x}_j \in \mathcal{T}_{\Gamma_D},$$

$$\mathcal{L}^{\text{pos}}_{t=0}(\boldsymbol{\theta}; \mathcal{T}_{t=0}) = \sum_{j=1}^{N_0} w_{j,0} |\hat{y}(\boldsymbol{x}_j; \boldsymbol{\theta}) - y^0(\boldsymbol{x}_j)|^2, \quad \boldsymbol{x}_j \in \mathcal{T}_{t=0},$$

$$\mathcal{L}^{\text{vel}}_{t=0}(\boldsymbol{\theta}; \mathcal{T}_{t=0}) = \sum_{j=1}^{N_0} w_{j,0} |\hat{y}_t(\boldsymbol{x}_j; \boldsymbol{\theta}) - y^1(\boldsymbol{x}_j)|^2, \quad \boldsymbol{x}_j \in \mathcal{T}_{t=0},$$

$$\mathcal{L}^{\text{pos}}_{t=T}(\boldsymbol{\theta}; \mathcal{T}_{t=T}) = \sum_{j=1}^{N_T} w_{j,T} |\hat{y}(\boldsymbol{x}_j; \boldsymbol{\theta})|^2, \quad \boldsymbol{x}_j \in \mathcal{T}_{t=T},$$

$$\mathcal{L}^{\text{vel}}_{t=T}(\boldsymbol{\theta}; \mathcal{T}_{t=T}) = \sum_{j=1}^{N_T} w_{j,T} |\hat{y}_t(\boldsymbol{x}_j; \boldsymbol{\theta})|^2, \quad \boldsymbol{x}_j \in \mathcal{T}_{t=T},$$

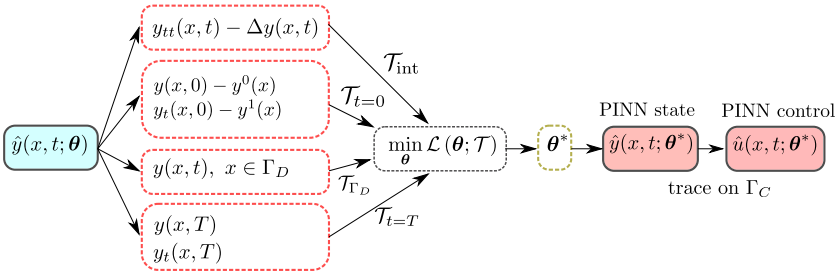

Fig. 3 PINN algorithm for approximating the exact state and control for the wave equation. The neural network $\hat{y}(x,t,\boldsymbol{\theta})$ is required to satisfy, in the least squares sense, the PDE, initial conditions, boundary condition and exact controllability conditions. Then, the residual on training points $\mathcal{L}(\boldsymbol{\theta};\mathcal{T})$ is minimized to get the optimal set of parameters $\boldsymbol{\theta}^*$ of the neural network. This leads to the PINN exact state $\hat{y}(x,t;\boldsymbol{\theta}^*)$. Finally, the PINN exact control $\hat{u}(x,t;\boldsymbol{\theta}^*)$ is obtained as the trace of the PINN state on the boundary control region Γ_C

where $w_{j,\text{int}}$, $w_{j,b}$, $w_{j,0}$, and $w_{j,T}$ are the weights of the quadrature rules.

The loss function used for training is

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta};\mathcal{T}) = &\mathcal{L}_{\text{int}}(\boldsymbol{\theta};\mathcal{T}_{\text{int}})\\ &+\mathcal{L}_{\Gamma_D}\left(\boldsymbol{\theta};\mathcal{T}_{\Gamma_D}\right)\\ &+\mathcal{L}^{\text{pos}}_{t=0}(\boldsymbol{\theta};\mathcal{T}_{t=0}) + \mathcal{L}^{\text{vel}}_{t=0}(\boldsymbol{\theta};\mathcal{T}_{t=0})\\ &+\mathcal{L}^{\text{pos}}_{t=T}(\boldsymbol{\theta};\mathcal{T}_{t=T}) + \mathcal{L}^{\text{vel}}_{t=T}(\boldsymbol{\theta};\mathcal{T}_{t=T})\,.\end{aligned} \tag{4}$$

Notice that no boundary condition is imposed on Γ_C. As is usual in the field of machine learning, all the derivatives included in the loss function are computed by using automatic differentiation [2].

Step 4: Training Process The final step in the PINN algorithm amounts to minimizing (4), i.e.,

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta};\mathcal{T})\,. \tag{5}$$

The approximation $\hat{u}(t;\boldsymbol{\theta}^*)$ of the control $u(x,t)$ is then obtained as the restriction of $\hat{y}(x,t;\boldsymbol{\theta}^*)$ to the boundary Γ_C, i.e.,

$$\hat{u}(x,t;\boldsymbol{\theta}^*) = \hat{y}(x,t;\boldsymbol{\theta}^*), \quad x \in \Gamma_C,\ 0 \le t \le T. \tag{6}$$

See Fig. 3 for an schematic diagram of the proposed algorithm.

Remark 2.1 Notice that the PINN algorithm proposed above is, in spirit, in the same line as the one considered in [30], where an error function is minimized in the sense of least squares. As a consequence, if this error function reaches the zero value, then the controllability condition is satisfied. A major difference with respect to classical numerical methods for control of PDEs is that the PINN-based approach is mesh-free as it does not require a (finite element) mesh for numerical approximation. Moreover,

the function that is used for numerical approximation is a neural network as opposed to (piece-wise) polynomials, that are the usual models of choice.

Remark 2.2 As is well known, the different terms that appear in the loss function (4) do not have the same strength, in general. At the practical level, this difficulty may be overcome by introducing additional weighting parameters in front of those terms. These would be new hyperparameters that the machine-learning-based algorithm ought to learn. It is clear that the introduction of these parameters does not affect the convergence results in the next section.

2.2 Heat Equation

Similar to the case of the wave equation, given an initial datum y^0 in a suitable function space, the null controllability problem for the heat equation amounts to finding a positive time $T > 0$ and a control function $u(x, t)$ such that the solution $y(x, t)$ of the system

$$\begin{array}{ll} y_t - \Delta y = 0, & \text{in } Q_T \\ y(x, 0) = y^0(x), & \text{in } \Omega \\ y(x, t) = 0, & \text{on } \Gamma_D \times (0, T) \\ y(x, t) = u(x, t) & \text{on } \Gamma_C \times (0, T) \end{array} \tag{7}$$

satisfies

$$y(x, T) = 0, \quad x \in \Omega. \tag{8}$$

It is well known [21] that if $y^0 \in L^2(\Omega)$, then, for any $T > 0$, problem (7)–(8) has a solution $u \in L^2(\Gamma_C \times (0, T))$.

The numerical approximation of problem (7)–(8) follows the same steps 1–4 as in the case of the wave equation. The only element to be modified is the loss function, which in this case is defined as the sum of

$$\mathcal{L}_{\text{int}}(\boldsymbol{\theta}; \mathcal{T}_{\text{int}}) = \sum_{j=1}^{N_{\text{int}}} w_{j,\text{int}} |\hat{y}_t(\boldsymbol{x}_j; \boldsymbol{\theta}) - \Delta \hat{y}(\boldsymbol{x}_j; \boldsymbol{\theta})|^2, \quad \boldsymbol{x}_j \in \mathcal{T}_{\text{int}},$$

$$\mathcal{L}_{\Gamma_D}(\boldsymbol{\theta}; \mathcal{T}_{\Gamma_D}) = \sum_{j=1}^{N_b} w_{j,b} |\hat{y}(\boldsymbol{x}_j; \boldsymbol{\theta})|^2, \quad \boldsymbol{x}_j \in \mathcal{T}_{\Gamma_D},$$

$$\mathcal{L}_{t=0}(\boldsymbol{\theta}; \mathcal{T}_{t=0}) = \sum_{j=1}^{N_0} w_{j,0} |\hat{y}(\boldsymbol{x}_j; \boldsymbol{\theta}) - y^0(\boldsymbol{x}_j)|^2, \quad \boldsymbol{x}_j \in \mathcal{T}_{t=0},$$

$$\mathcal{L}_{t=T}(\boldsymbol{\theta}; \mathcal{T}_{t=T}) = \sum_{j=1}^{N_T} w_{j,T} |\hat{y}(\boldsymbol{x}_j; \boldsymbol{\theta})|^2, \quad \boldsymbol{x}_j \in \mathcal{T}_{t=T}.$$

2.3 Extension to General Evolution PDE Systems

Consider now a general evolution system of the form

$$\begin{array}{ll} y_t + Ay = 0, & \text{in } Q_T \\ y(x,0) = y^0(x), & \text{in } \Omega \\ y(x,t) = 0, & \text{on } \Gamma_D \times (0,T) \\ y(x,t) = u(x,t) & \text{on } \Gamma_C \times (0,T)\,, \end{array} \tag{9}$$

where A is a generic (linear or nonlinear) operator, and the state $y = y(x,t)$ is, in general, a vector function.

As in the preceding two cases, the goal is to find a positive time T and a control function $u(x,t)$ such that the solution to (9) satisfies

$$y(x,T) = 0, \quad x \in \Omega. \tag{10}$$

The PINNs algorithm described above for the wave and heat equations also applies in this general framework with few changes. Actually, the only step that must be updated is the one corresponding to the loss function that now takes the form:

$$\begin{aligned} \mathcal{L}_{\text{int}}(\boldsymbol{\theta}; \mathcal{T}_{\text{int}}) &= \sum_{j=1}^{N_{\text{int}}} w_{j,\text{int}} \|\hat{y}_t(\boldsymbol{x}_j;\boldsymbol{\theta}) - \Delta\hat{y}(\boldsymbol{x}_j;\boldsymbol{\theta})\|^2, \quad \boldsymbol{x}_j \in \mathcal{T}_{\text{int}}, \\ \mathcal{L}_{\Gamma_D}\left(\boldsymbol{\theta}; \mathcal{T}_{\Gamma_D}\right) &= \sum_{j=1}^{N_b} w_{j,b} \|\hat{y}(\boldsymbol{x}_j;\boldsymbol{\theta})\|^2, \quad \boldsymbol{x}_j \in \mathcal{T}_{\Gamma_D}, \\ \mathcal{L}_{t=0}(\boldsymbol{\theta}; \mathcal{T}_{t=0}) &= \sum_{j=1}^{N_0} w_{j,0} \|\hat{y}(\boldsymbol{x}_j;\boldsymbol{\theta}) - y^0(\boldsymbol{x}_j)\|^2, \quad \boldsymbol{x}_j \in \mathcal{T}_{t=0}, \\ \mathcal{L}_{t=T}(\boldsymbol{\theta}; \mathcal{T}_{t=T}) &= \sum_{j=1}^{N_T} w_{j,T} \|\hat{y}(\boldsymbol{x}_j;\boldsymbol{\theta})\|^2, \quad \boldsymbol{x}_j \in \mathcal{T}_{t=T}, \end{aligned}$$

where $\|\cdot\|$ stands for the Euclidean norm.

3 Estimates on Generalization Error

This section aims at obtaining error estimates for the so-called generalization error for both control and state. The generalization error for the control variable u is defined by

$$\mathcal{E}_{\text{gener}}(u) := \|u - \hat{u}\|, \tag{11}$$

where $u = u(x,t)$ is the exact control of minimal L^2-norm of the continuous problem, $\hat{u} = \hat{u}\left(x,t;\boldsymbol{\theta}^*\right)$ is its numerical approximation obtained from the algorithm proposed

above, and $\|\cdot\|$ is an appropriate norm. The generalization error for the state variable is similarly defined.

The generalization error (11) is typically decomposed into *approximation error*, which is due to the choice of the hypothesis space (two-layer, multilayer, residual, convolutional neural networks, etc.), and *estimation error*, due to the fact that the surrogate control $\hat{u}$ is computed from a finite dataset. Of course, the generalization error also depends on a crucial way on the specific algorithm proposed for training. In particular, PINN solutions obtained from the proposed method are obtained by solving highly nonconvex optimization problems that typically get stuck in local minima. Estimating this optimization error is a very challenging open problem.

Error estimates for the approximation error of some hypothesis spaces are by now well known. For instance, for the case of Barron space of two-layer neural networks, the approximation error in the L^2-norm scales as $\mathcal{O}\left(m^{-1/2}\right)$, with m being the number of neurons in the network, and independently of the dimension d. As for the estimation error, it is also known that the Rademacher complexity of Barron space, which controls the estimation error, is controlled by a Monte Carlo rate $\mathcal{O}\left(N^{-1/2}\right)$, where N is the number of sampling points used for training. We refer the reader to [42] and the references therein for more details on this issue. In particular, these results support the choice of multilayer neural networks of Sect. 2.

Regarding the PINNs algorithm for solving PDEs, convergence results w.r.t. the number of sampling points used for training have been recently obtained in [38] for the case of second-order linear elliptic and parabolic equations with smooth solutions. It is also worth to mentioning article [27] where error estimates, in terms of training error and the number of sampling points, are derived for the generalization error of a class of data assimilation problems.

Following [27], we next prove error estimates for control and state and for the class of controllability problems considered here. The two key ingredients to get such error bounds are observability inequalities and error estimates for quadrature rules. The precise observability inequalities that are needed in our cases will be detailed in the next subsections. Concerning quadrature error estimates, these are very well known in the literature but for the sake of completeness, we now recall some basic concepts and results on this issue.

3.1 Error Estimates for Quadrature Rules

For a given function $f : \mathcal{D} \subset \mathbb{R}^d \to \mathbb{R}$, a quadrature rule approximating the integral

$$\overline{f} := \int_{\mathcal{D}} f(x)\, dx$$

is defined by

$$\overline{f}_N := \sum_{j=1}^{N} w_j f(x_j),$$

where (x_j, w_j), $1 \leq j \leq N$, are the nodes and weights of the quadrature rule. Quadrature errors depend on the specific rule used, on the smoothness of the function f and on the dimension d. For regular functions and low dimensions, one typically may use Gauss or Clenshaw–Curtis rules. Rules based on low discrepancy sequences such as Sobol sequences are the rules of choice for intermediate dimensions [39]. In both cases, error estimates for these quadrature rules take the general form

$$|\overline{f} - \overline{f}_N| \leq C_q N^{-\alpha}, \quad \alpha > 0, \tag{12}$$

where α depends on the regularity of f and the constant $C_q = C_q(d)$, which also depends on f and its derivatives, explodes as $d \to \infty$. Monte Carlo integration is immune to the curse of dimensionality and applies to non-smooth integrands. As is well known, the error estimate in that case is as in (12) where C_q is independent of the dimension d and $\alpha = 1/2$.

3.2 Wave Equation

The generalization error in the control variable u due to the PINN algorithm proposed in Sect. 2.1 is defined as

$$\mathcal{E}_{\text{gener}}(u) := \|u - \hat{u}\|_{L^2(\Gamma_C;(0,T))}, \tag{13}$$

where $u = u(t)$ is the exact control of the continuous problem (1)–(2) and $\hat{u} = \hat{u}\left(t; \boldsymbol{\theta}^*\right)$ is its numerical approximation given by (6).

Similarly, the generalization error for the state variable is defined by

$$\mathcal{E}_{\text{gener}}(y) := \|y - \hat{y}\|_{C(0,T;L^2(\Omega))\cap C^1(0,T;H^{-1}(\Omega))}. \tag{14}$$

As is usual in machine learning's terminology, the so-called training error for PINNs algorithm is given by

$$\begin{aligned}\mathcal{E}_{\text{train}} := \mathcal{E}_{\text{train, int}} + \mathcal{E}_{\text{train, boundary}} + \mathcal{E}_{\text{train, initialpos}} + \mathcal{E}_{\text{train, initialvel}} \\ + \mathcal{E}_{\text{train, finalpos}} + \mathcal{E}_{\text{train, finalvel}},\end{aligned} \tag{15}$$

where

$$\begin{aligned}
\mathcal{E}_{\text{train, int}} &= \left(\mathcal{L}_{\text{int}}\left(\boldsymbol{\theta}^*; \mathcal{T}_{\text{int}}\right)\right)^{1/2} \\
\mathcal{E}_{\text{train, boundary}} &= \left(\mathcal{L}_{\Gamma_D}\left(\boldsymbol{\theta}^*; \mathcal{T}_{\Gamma_D}\right)\right)^{1/2} \\
\mathcal{E}_{\text{train, initialpos}} &= \left(\mathcal{L}^{\text{pos}}_{t=0}\left(\boldsymbol{\theta}^*; \mathcal{T}_{t=0}\right)\right)^{1/2} \\
\mathcal{E}_{\text{train, initialvel}} &= \left(\mathcal{L}^{\text{vel}}_{t=0}\left(\boldsymbol{\theta}^*; \mathcal{T}_{t=0}\right)\right)^{1/2} \\
\mathcal{E}_{\text{train, finalpos}} &= \left(\mathcal{L}^{\text{pos}}_{t=T}\left(\boldsymbol{\theta}^*; \mathcal{T}_{t=T}\right)\right)^{1/2} \\
\mathcal{E}_{\text{train, finalvel}} &= \left(\mathcal{L}^{\text{vel}}_{t=T}\left(\boldsymbol{\theta}^*; \mathcal{T}_{t=T}\right)\right)^{1/2}
\end{aligned} \tag{16}$$

and $\boldsymbol{\theta}^*$ is as in (5).

Next, we recall classical observability and energy inequalities for the wave equation:

Lemma 3.1 *Let us assume that the domain Ω satisfies the geometrical controllability condition [3], and let $T > 0$ be large enough. Given initial and final conditions $(z_0^0, z_0^1), (z_T^0, z_T^1) \in L^2(\Omega) \times H^{-1}(\Omega)$, there exists a control function $v \in L^2(\Gamma_C; (0, T))$ such that the solution $z(x, t)$ of the system*

$$\begin{aligned} &z_{tt} - \Delta z = 0, && \text{in } Q_T \\ &z(x, 0) = z_0^0(x), && \text{in } \Omega \\ &z_t(x, 0) = z_0^1(x) && \text{in } \Omega \\ &z(x, t) = 0, && \text{on } \Gamma_D \times (0, T) \\ &z(x, t) = v(x, t) && \text{on } \Gamma_C \times (0, T) \end{aligned} \tag{17}$$

satisfies

$$z(x, T) = z_T^0(x), \quad z_t(x, T) = z_T^1(x, T), \quad x \in \Omega. \tag{18}$$

Moreover,

$$\|v\|_{L^2(\Gamma_C;(0,T))} \leq C_o \left(\|z_0^0\|_{L^2(\Omega)} + \|z_0^1\|_{H^{-1}(\Omega)} + \|z_T^0\|_{L^2(\Omega)} + \|z_T^1\|_{H^{-1}(\Omega)} \right), \tag{19}$$

for a positive constant $C_o = C_o(\Omega, T)$ which depends on Ω and T, but is independent of the initial and final data.

Lemma 3.2 *Let $\left(z_0^0, z_0^1\right) \in L^2(\Omega) \times H^{-1}(\Omega)$ and $g \in L^2(\partial\Omega \times (0, T))$. Consider the non-homogeneous system*

$$\begin{aligned} &z_{tt} - \Delta z = f(x, t), && \text{in } Q_T \\ &z(x, 0) = z_0^0(x), && \text{in } \Omega \\ &z_t(x, 0) = z_0^1(x) && \text{in } \Omega \\ &z(x, t) = g(x, t), && \text{on } \partial\Omega \times (0, T). \end{aligned} \tag{20}$$

Then, there exists a positive constant $C_e = C_e(\Omega, T)$ such that

$$\begin{aligned} &\|z\|_{C(0,T;L^2(\Omega))} + \|z_t\|_{C(0,T;H^{-1}(\Omega))} \\ &\quad \leq C_e \left(\|z_0^0\|_{L^2(\Omega)} + \|z_0^1\|_{H^{-1}(\Omega)} + \|g\|_{L^2(\partial\Omega\times(0,T))} \right). \end{aligned} \tag{21}$$

We are now in a position to estimate the generalization error for our PINNs-based algorithm.

Theorem 3.1 *Let $y = y(x, t) \in C^2\left(\overline{Q_T}\right)$ be a classical solution of (1)–(2), and let $\hat{y} = \hat{y}(x, t; \boldsymbol{\theta}^*)$ its PINN approximation obtained by the method proposed in Sect. 2.1.*

It is assumed that $\hat{y} \in C^2\left(\overline{Q_T}\right)$. *Let* $u = u(x,t)$ *and* $\hat{u} = \hat{u}\left(x,t;\boldsymbol{\theta}^*\right)$ *be the exact control of the continuous system* (1)–(2) *and its PINN approximation, respectively. Then, the following estimate for the generalization error in the control variable holds:*

$$\begin{aligned}\mathcal{E}_{gener}(u) \leq C\Big(&\mathcal{E}_{train,\,int} + C_{q_{int}}^{1/2} N_{int}^{-\alpha_{int}/2}\\ &+\mathcal{E}_{train,\,boundary} + C_{qb}^{1/2} N_b^{-\alpha_b/2}\\ &+\mathcal{E}_{train,\,initialpos} + C_{qip}^{1/2} N_0^{-\alpha_{ip}/2}\\ &+\mathcal{E}_{train,\,initialvel} + C_{qiv}^{1/2} N_0^{-\alpha_{iv}/2}\\ &+\mathcal{E}_{train,\,finalpos} + C_{qfp}^{1/2} N_T^{-\alpha_{fp}/2}\\ &+\mathcal{E}_{train,\,finalvel} + C_{fv}^{1/2} N_T^{-\alpha_{fv}/2}\Big),\end{aligned} \tag{22}$$

where $C = C(\Omega, T)$, *and consequently* $C = C(d)$ *also depends on the spatial dimension* d. *The constants* C_{q-} *and exponents* α_- *are the ones associated with quadrature rules as in* (12).

A similar estimate, with different constants, also holds for the generalization error in the state variable, as given by (14).

Proof Let $\overline{y} = y - \hat{y}$ and $\overline{u} = u - \hat{u}$ be the error in the state and control variables, respectively. By linearity, $\overline{y}$ solves

$$\begin{aligned}&\overline{y}_{tt} - \Delta\overline{y} = \hat{y}_{tt} - \Delta\hat{y}, && \text{in } Q_T\\ &\overline{y}(x,0) = y^0(x) - \hat{y}(x,0), && \text{in } \Omega\\ &\overline{y}_t(x,0) = y^1(x) - \hat{y}_t(x,0) && \text{in } \Omega\\ &\overline{y}(x,T) = \hat{y}(x,T), && \text{in } \Omega\\ &\overline{y}_t(x,T) = \hat{y}_t(x,T) && \text{in } \Omega\\ &\overline{y}(x,t) = \hat{y}(x,t), && \text{on } \Gamma_D \times (0,T)\\ &\overline{y}(x,t) = u(x,t) - \hat{y}(x,t) && \text{on } \Gamma_C \times (0,T).\end{aligned} \tag{23}$$

Again by linearity, $\overline{y}(x,t;\boldsymbol{\theta})$ is decomposed as $\overline{y} = \overline{y}^1 + \overline{y}^2$, where $\overline{y}^1$ and $\overline{y}^2$ are, respectively, solutions to

$$\begin{aligned}&\overline{y}^1_{tt} - \Delta\overline{y}^1 = 0, && \text{in } Q_T\\ &\overline{y}^1(x,0) = y^0(x) - \hat{y}(x,0), && \text{in } \Omega\\ &\overline{y}^1_t(x,0) = y^1(x) - \hat{y}_t(x,0) && \text{in } \Omega\\ &\overline{y}^1(x,t) = 0, && \text{on } \Gamma_D \times (0,T)\\ &\overline{y}^1(x,t) = u(x,t) - \hat{y}(x,t) && \text{on } \Gamma_C \times (0,T)\end{aligned} \tag{24}$$

and

$$
\begin{aligned}
&\overline{y}^2_{tt} - \Delta \overline{y}^2 = \hat{y}_{tt} - \Delta \hat{y}, && \text{in } Q_T \\
&\overline{y}^2(x,0) = 0, && \text{in } \Omega \\
&\overline{y}^2_t(x,0) = 0 && \text{in } \Omega \\
&\overline{y}^2(x,T) = \hat{y}(x,T) - \overline{y}^1(x,T), && \text{in } \Omega \\
&\overline{y}^2_t(x,T) = \hat{y}_t(x,T) - \overline{y}^1_t(x,T), && \text{in } \Omega \\
&\overline{y}^2(x,t) = \hat{y}(x,t), && \text{on } \Gamma_D \times (0,T) \\
&\overline{y}^2(x,t) = 0 && \text{on } \Gamma_C \times (0,T).
\end{aligned} \tag{25}
$$

By applying the observability inequality (19) to system (24), and the energy estimate (21) to (25),

$$
\begin{aligned}
&\|u - \hat{u}\|_{L^2(\Gamma_C;(0,T))} \\
&\quad \le C_o \left(\|y^0 - \hat{y}(0)\|_{L^2(\Omega)} + \|y^1 - \hat{y}_t(0)\|_{H^{-1}(\Omega)} + \|\overline{y}^1(T)\|_{L^2(\Omega)} + \|\overline{y}^1_t(T)\|_{H^{-1}(\Omega)} \right) \\
&\quad \le C_o \Big(\|y^0 - \hat{y}(0)\|_{L^2(\Omega)} + \|y^1 - \hat{y}_t(0)\|_{L^2(\Omega)} + \|\hat{y}(T)\|_{L^2(\Omega)} + \|\hat{y}_t(T)\|_{L^2(\Omega)} \\
&\qquad + \|\overline{y}^2(T)\|_{L^2(\Omega)} + \|\overline{y}^2_t(T)\|_{H^{-1}(\Omega)} \Big) \\
&\quad \le C_o \Big(\|y^0 - \hat{y}(0)\|_{L^2(\Omega)} + \|y^1 - \hat{y}_t(0)\|_{L^2(\Omega)} + \|\hat{y}(T)\|_{L^2(\Omega)} + \|\hat{y}_t(T)\|_{L^2(\Omega)} \\
&\qquad + C_e \left(\|\hat{y}\|_{L^2(\Gamma_D \times (0,T))} + \|\hat{y}_{tt} - \Delta \hat{y}\|_{L^2(0,T;L^2(\Omega))} \right) \Big).
\end{aligned} \tag{26}
$$

Estimate (22) then follows by applying (12). The corresponding estimate for the generalization error (14) is an immediate consequence of (21) and (22). □

Although it has not been written explicitly hereinabove, it is clear that generalization errors depend on the specific type and size of neural network as well as on the type and number of quadrature nodes which are selected from the very beginning. Thus, denoting by $\mathcal{H}_m$ the hypothesis space considered for numerical approximation, where m denotes the number of neurons (or free parameters) in the neural network, and by N the number of collocation points used for quadrature, to make explicit this dependence we write

$$
\mathcal{E}_{\text{gener}}(u) = \mathcal{E}^{m,N}_{\text{gener}}(u) \quad \text{and} \quad \mathcal{E}_{\text{gener}}(y) = \mathcal{E}^{m,N}_{\text{gener}}(y).
$$

Next, the behavior of the generalization errors is analyzed when the size m of single-layer neural networks goes to infinity and so does the sampling size ($N \to \infty$).

Let us consider the hypothesis space of single-layer neural nets

$$
\mathcal{H}_m := \left\{ y_m(\boldsymbol{x}) := \sum_{i=1}^{m} a_i \sigma\left(\boldsymbol{\omega}_i \boldsymbol{x} + b_i\right) : \boldsymbol{x}, \boldsymbol{\omega}_i \in \mathbb{R}^{d+1}, a_i, b_i \in \mathbb{R} \right\}.
$$

The training process (5) may be rewritten in the equivalent form

$$
\hat{y}_{m,N} = \arg \min_{y_m \in \mathcal{H}_m} \mathcal{L}\left(y_m; \mathcal{T}_N\right). \tag{27}
$$

From now on it is assumed that the optimization problem (27) has a solution. Otherwise, one can always add a regularization term of the form $\|\boldsymbol{\theta}\|^2$.

We now recall the following *universal approximation theorem* due to Pinkus [33, Th. 4.1].

Theorem 3.2 *Let $f \in C^k(\mathbb{R}^{d+1})$. Assume that the activation function $\sigma \in C^k(\mathbb{R})$ is not a polynomial. Then, for any compact set $K \subset \mathbb{R}^{d+1}$ and any $\varepsilon > 0$ there exists $m \in \mathbb{N}$ and $y_m \in \mathcal{H}_m$ such that*

$$\max_{\boldsymbol{x}\in K} |D^l f(\boldsymbol{x}) - D^l y_m(\boldsymbol{x})| \leq \varepsilon$$

for all multiindex $l \leq k$.

Corollary 3.1 *Assume that the activation function $\sigma \in C^k(\mathbb{R})$ is not a polynomial. With the same assumptions as in Theorem 3.1 and considering subsequences, still labeled by m and N, one has*

$$\lim_{N\to\infty} \lim_{m\to\infty} \mathcal{E}^{m,N}_{gener}(u) = \lim_{N\to\infty} \lim_{m\to\infty} \mathcal{E}^{m,N}_{gener}(y) = 0. \tag{28}$$

Proof Let us fix $\varepsilon > 0$. We apply Theorem 3.2 for $K = \overline{Q_T}$ and $f = y$, solution of the controllability problem (1)–(2). Then, there exist $m = m(\varepsilon) \in \mathbb{N}$ and corresponding $y_m \in \mathcal{H}_m$ such that

$$\begin{aligned} &\|(y_m)_{tt} - \Delta y_m\|_{L^2(0,T;L^2(\Omega))} + \|y_m\|_{L^2(\Gamma_D\times(0,T))} \\ &\quad + \|y^0 - y_m(0)\|_{L^2(\Omega)} + \|y^1 - (y_m)_t(0)\|_{L^2(\Omega)} \\ &\quad + \|y_m(T)\|_{L^2(\Omega)} + \|(y_m)_t(T)\|_{L^2(\Omega)} \\ &\leq \varepsilon/2. \end{aligned} \tag{29}$$

Each of the terms in the left-hand side of (29) is now expressed by using a quadrature rule with collocation nodes $\mathcal{T}_N$. Then, taking into account the optimality of $\hat{y}_{m,N}$, as given by (27), one deduces that the sum of training errors that appear in the right-hand side of (22) is less than or equal to $\varepsilon/2$.

Moreover, for fixed $m = m(\varepsilon)$, there exists N such that the sum of quadrature errors in (22) is also less than or equal to $\varepsilon/2$. Thus, $\mathcal{E}^{m,N}_{\text{gener}}(u) \leq \varepsilon$. The arbitrariness of ε gives the result for the generalization error in the control variable. The case of the state variable is completely analogous. □

3.3 Extension to Other PDE Systems and Neural Network Architectures

It is clear that the arguments and conclusions of Theorem 3.1 and Corollary 3.1 extend to any linear system of PDEs for which observability as well as energy inequalities similar to those in (19) and (21) hold. Linearity of the PDE is used in an essential way in the proof of Theorem 3.1. Thus, a different argument is needed to extend this result to the case of nonlinear PDEs.

The proof of Corollary 3.1 relies on the universal approximation theorem by Pinkus for the case of single-layer neural networks. Thus, the conclusion of Corollary 3.1 also holds for other neural network architectures for which such a density result is true.

4 Numerical Experiments

In this section, we test the performance of the proposed method in three exact controllability problems. The first experiment aims at checking the accuracy of the method on a very simple controllability problem for the wave equation for which an exact solution is explicitly known. In the second experiment, the high-dimensional situation is tested on a controllability problem for the heat equation. The last experiment considers a semilinear wave equation.

As indicated at the beginning of Sect. 3, the optimization error due to the gradient-based algorithms used for training is a key ingredient in the total error associated with the proposed PINN algorithm. This error has not been accounted for in Theorem 3.1 and Corollary 3.1. However, the numerical simulation results presented in this section do incorporate this error. As a consequence, simulation results are unable to illustrate with accuracy the theoretical findings of Sect. 3. The gap between the theoretical error estimates and the simulation results is accounted for by the optimization error in the training process.

In all experiments that follow, a multilayer neural network, as described in Sect. 2, with the tanh as activation function, is used. Sobol quadrature nodes [39] are employed for training the neural network. The training process, i.e., minimization of $\mathcal{L}(\boldsymbol{\theta}; \mathcal{T}_N)$, is carried out with the ADAM optimizer [19] with learning rate 10^{-3} for the first 20000 epochs. Then, a L-BFGS optimizer [7] is employed to accelerate convergence. The required gradients are computed by using automatic differentiation [2]. The descent algorithm is initialized with Glorot uniform [14]. As is well known, results obtained from gradient-based optimizers depend on initialization. A common practice to deal with this issue is to perform an emsemble training [24]. However, the use of this and other more sophisticated techniques (residual-based adaptive refinement (RAR) [23], dropout [40], batch normalization [18], etc.) is not the purpose of this paper which aims at illustrating the possible use of PINNs in the topic of controllability of PDEs.

4.1 Experiment 1: Linear Wave Equation

We consider the control system (1)–(2) in the domain $\Omega = (0, 1)$ for the data

$$y^0(x) = \sin(\pi x), \quad y^1(x) = 0, \quad 0 \leq x \leq 1,$$

and for the control time $T = 2$. An explicit solution of the problem is easily obtained by using D'Alembert formula. Indeed, by considering the function

$$\tilde{y^0}(x) = \begin{cases} \sin(\pi x), & -1 \le x \le 1 \\ 0, & \text{elsewhere,} \end{cases}$$

the explicit exact state is given by

$$y(x,t) = \frac{1}{2}\left(\tilde{y^0}(x-t) + \tilde{y^0}(x+t)\right), \quad 0 \le x \le 1, \ 0 \le t \le 2, \tag{30}$$

and the exact control is

$$u(t) = \begin{cases} \frac{1}{2}y^0(1-t), & 0 \le t \le 1 \\ -\frac{1}{2}y^0(t-1), & 1 \le t \le 2. \end{cases} \tag{31}$$

Remark 4.1 We notice that the control given by (31) is the one of minimal L^2-norm. This is no longer true if the initial velocity y^1 is different from zero (see [16], Section 4.1 for details).

The efficiency of the proposed PINN-based algorithm in approximating the solution to this problem is analyzed next. The generalization error in the control variable $\mathcal{E}_{\text{gener}}(u) := \|u - \hat{u}\|_{L^2(0,T)}$, L^2-relative error $\|u - \hat{u}\|_{L^2(0,T)}/\|u\|_{L^2(0,T)}$, and total training error $\mathcal{E}_{\text{train}} := \mathcal{L}\left(\boldsymbol{\theta}^*; \mathcal{T}_N\right)$ are computed for several values of the total number N of training points and several architectures of the neural network. The effect of regularization, where the term $\lambda_{\text{reg}}\|\boldsymbol{\theta}\|_2^2$ is added to the loss function (4), with $\lambda_{\text{reg}} > 0$, is also studied.

Once the training process is finished and the optimal set of parameters $\boldsymbol{\theta}^*$ is obtained, the PINN control $\hat{u}(t;\boldsymbol{\theta}^*) = \hat{y}\left(1,t;\boldsymbol{\theta}^*\right)$ is computed on a uniform mesh of size $h = 0.02$ in the segment $(1,t)$, $0 \le t \le 2$. Both the generalization error and the L^2-relative error are then approximated by using the same mesh. The training points are split into interior and boundary points as follows: for a given positive integer N_0, $3N_0$ points are located on the boundary and N_0^2 in the interior domain. Thus, $N = N_0^2 + 3N_0$.

Tables 1 and 2 collect simulation results for a multilayer neural network composed of 4 hidden layers and 50 neurons in each layer. It is observed that both the generalization error and the L^2-relative error slowly decrease as the number of training points increases. The comparison between Tables 1 and 2 shows that regularization does not increase the level of accuracy. Table 3 displays simulation results for the case of a single-layer architecture having the same number of neurons as in the multilayer neural network considered in Tables 1 and 2. It is observed that the single-layer architecture provides slightly less accurate results.

Figure 4 shows the exact control (31) and the PINN control $\hat{u}\left(t;\boldsymbol{\theta}^*\right)$, and the error between exact and PINN states.

The effect of increasing the depth (number of hidden layers) and width (number of neurons for layer) of the neural network has been also tested. We have observed that the level of accuracy in the solutions is not improved significantly. This is in agreement with previous studies (see, e.g., [23]) that show that a relative small neural network is able to approximate with accuracy of smooth solutions of PDEs.

Table 1 Experiment 1 (linear wave equation): No regularization. Number of training points N versus generalization error $\mathcal{E}_{\text{gener}}(u)$, L^2- relative error and training error $\mathcal{E}_{\text{train}}$ for a multilayer neural network composed of 4 hidden layers and 50 neurons in each layer

	$N = 130$	$N = 700$	$N = 2650$	$N = 5850$	$N = 10300$
$\mathcal{E}_{\text{gener}}(u)$	9.08×10^{-1}	1.365×10^{-1}	1.20×10^{-1}	6.98×10^{-2}	2.37×10^{-2}
$\frac{\|u-\hat{u}\|_{L^2(0,T)}}{\|u\|_{L^2(0,T)}}$	2.58×10^{-1}	3.88×10^{-2}	3.39×10^{-2}	1.98×10^{-2}	6.74×10^{-3}
$\mathcal{E}_{\text{train}}$	4.72×10^{-7}	4.68×10^{-6}	2.16×10^{-6}	4.40×10^{-6}	3.86×10^{-6}

Table 2 Experiment 1 (linear wave equation): Regularization with $\lambda_{\text{reg}} = 10^{-7}$. Number of training points N versus generalization error $\mathcal{E}_{\text{gener}}(u)$, L^2- relative error and training error $\mathcal{E}_{\text{train}}$ for a multilayer neural network composed of 4 hidden layers and 50 neurons in each layer

	$N = 130$	$N = 700$	$N = 2650$	$N = 5850$	$N = 10300$
$\mathcal{E}_{\text{gener}}(u)$	1.09×10^{0}	1.44×10^{-1}	1.20×10^{-1}	9.48×10^{-2}	6.51×10^{-2}
$\frac{\|u-\hat{u}\|_{L^2(0,T)}}{\|u\|_{L^2(0,T)}}$	3.11×10^{-1}	4.10×10^{-2}	2.08×10^{-2}	2.69×10^{-2}	1.85×10^{-2}
$\mathcal{E}_{\text{train}}$	2.37×10^{-7}	6.10×10^{-7}	1.06×10^{-6}	1.47×10^{-6}	1.41×10^{-6}

Table 3 Experiment 1 (linear wave equation): No regularization. Number of training points N versus generalization error $\mathcal{E}_{\text{gener}}(u)$, L^2- relative error and training error $\mathcal{E}_{\text{train}}$ for a single-layer neural network composed of 200 neurons

	$N = 130$	$N = 700$	$N = 2650$	$N = 5850$	$N = 10300$
$\mathcal{E}_{\text{gener}}(u)$	3.06×10^{-1}	2.35×10^{-1}	2.10×10^{-1}	3.33×10^{-1}	1.68×10^{-1}
$\frac{\|u-\hat{u}\|_{L^2(0,T)}}{\|u\|_{L^2(0,T)}}$	8.69×10^{-2}	6.68×10^{-2}	5.95×10^{-2}	9.47×10^{-2}	4.79×10^{-2}
$\mathcal{E}_{\text{train}}$	6.95×10^{-5}	2.73×10^{-4}	5.72×10^{-4}	1.39×10^{-3}	4.23×10^{-4}

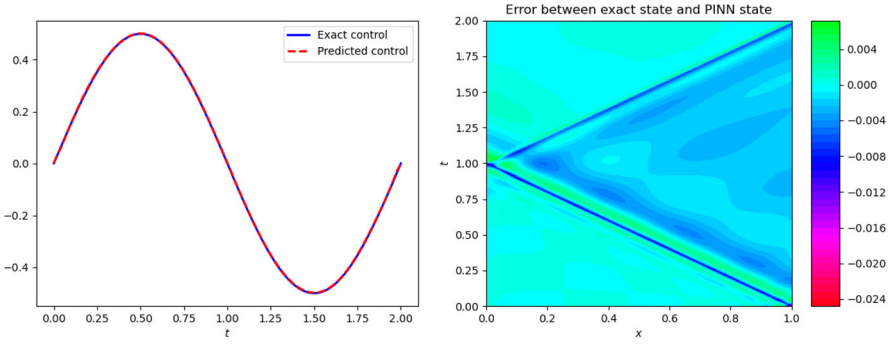

Fig. 4 Experiment 1 (linear wave equation). Comparison between exact control $u(t)$ and PINN (or predicted) control $\hat{u}(t;\boldsymbol{\theta}^*)$ (left), and error between exact state and PINN state, i.e., $y(x,t) - \hat{y}(x,t;\boldsymbol{\theta}^*)$ (right) Neural network composed of 4 hidden layers and 50 neurons in each layer. No regularization. Number of training points $N = 10300$

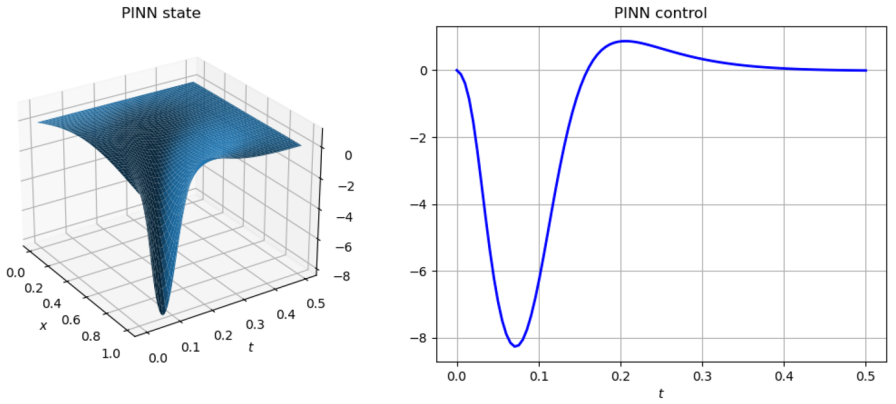

Fig. 5 Experiment 2 (linear heat equation). PINN (or predicted) state (left) and PINN control (right). Neural network composed of 5 hidden layers and 100 neurons in each layer. Number of training points $N = 10300$

4.2 Experiment 2: Linear Heat Equation

In this experiment, we consider the heat system (7)–(8) for $\Omega = (0, 1)^d$ and $d = 1, 5, 10$, and 20.

The One-Dimensional case For comparison purposes, the case $d = 1$ is addressed next. The first mode of the Laplacian $y^0(x) = \sin(\pi x)$, $0 < x < 1$, is taken as the initial condition. On $x = 0$, a zero Dirichlet boundary condition is imposed. The control function acts on the extreme $x = 1$. In order to have a better control of the diffusion, the Laplacian Δ is replaced by $\kappa\Delta$, with $\kappa = 0.25$. The control time is $T = 0.5$. This experiment has been previously considered in [30, Subsection 5.1]. Figure 5 shows the predicted state (left) and control (right) obtained from the PINN algorithm described in Sect. 2.2, and for a feedforward neural network composed of 5 hidden layers and 100 neurons in each layer. The number of training points is $N = 10300$. Once the training process is completed, the training error for the controllability condition $y(x, T) = 0$, $0 < x < 1$, which provides an approximation of $\|y(\cdot, T)\|_{L^2(\Omega)}$ is 1.17×10^{-5}. It is observed in Fig. 6 that both the PINN control and state have a similar profile as in [30, Figures 2 and 4 (left)]. However, no oscillations near the final time appear in the PINN control. This is not contradictory with the results in [30] since it is well known that no uniqueness of null controls holds.

The Multi-dimensional Case In order to check the accuracy of the proposed method in high dimensions, we consider the following control to the trajectory problem for $\Omega = (0, 1)^d$ and $T = 1$:

$$\begin{array}{ll} y_t - \Delta y = 0, & \text{in } Q_T := \Omega \times (0, T) \\ y(x, 0) = \frac{\|x\|^2}{d}, & \text{in } \Omega \\ y(x, t) = u(x, t) & \text{on } \partial\Omega \times (0, T) \\ y(x, T) = \frac{\|x\|^2}{d} + 2 & \text{in } \Omega. \end{array} \tag{32}$$

Table 4 Experiment 2 (linear heat equation): Dimension versus L^2-relative error in the state variable and training error $\mathcal{E}_{\text{train}}$. Multilayer neural network composed of 4 hidden layers and 50 neurons in each layer. Number of training points $N = 23000$

dimension d	1	5	10	20
$\frac{\|y-\hat{y}\|_{L^2(\Omega\times(0,T))}}{\|y\|_{L^2(\Omega\times(0,T))}}$	9.6×10^{-4}	1.45×10^{-3}	2.86×10^{-3}	3.2×10^{-2}
$\mathcal{E}_{\text{train}}$	1.54×10^{-7}	2.58×10^{-7}	3.15×10^{-7}	1.26×10^{-7}

This problem has an explicit solution [28], which is given by $y(x,t) = \frac{\|x\|^2}{d} + 2t$, $x \in \Omega$. The control function is obtained as the trace of y on $\partial\Omega$. Table 4 displays simulation results for L^2- relative error in the state variable and training error. It is observed that even for high dimensions the relative error in the state variable is very low. Accuracy is similar to the one obtained for forward PDEs via PINNs [28].

4.3 Experiment 3: A Semilinear Wave Equation

Next, we consider a nonlinear situation, precisely the case of a semilinear wave equation. Positive results on the exact controllability for semilinear wave equations have been obtained, among others, in [31, 44, 45].

In this experiment, the following null controllability problem for a semilinear wave equation is considered:

$$\begin{array}{ll} y_{tt} - y_{xx} = 4y^2, & \text{in } (0,1)\times(0,2) \\ y(x,0) = 1.5\sin(3\pi x), & \text{in } (0,1) \\ y_t(x,0) = x^2 & \text{in } (0,1) \\ y(0,t) = 0, & \text{on } (0,2) \\ y(1,t) = u(t) & \text{on } (0,2), \\ y(x,2) = y_t(x,2) = 0 & \text{in } (0,1). \end{array} \tag{33}$$

This problem has been previously studied in [8, Subsection 4.2.1].

The proposed PINN-based algorithm has been tested for different neural network architectures and number of training points. Table 5 collects the simulation results for all contributions in training error as in (16). Recall that $\mathcal{E}_{\text{train, int}}$ is the training error associated with the residual of the PDE, $\mathcal{E}_{\text{train, boundary}}$ corresponds to the boundary condition at $x = 0$, $\mathcal{E}_{\text{train, initialpos}}$ and $\mathcal{E}_{\text{train, initialvel}}$ are, respectively, the training errors for initial position and velocity, and finally, $\mathcal{E}_{\text{train, finalpos}}$ and $\mathcal{E}_{\text{train, initialvel}}$ are the training errors for the controllability condition at the control time $T = 2$. It is observed in Table 5 that increasing the number of training points does not reduce significantly training errors. This is in accordance with previous studies (see, e.g., numerical experiments in [27]). Recall that the training error includes the optimization error due to the gradient-based descent algorithms used for minimization of the highly nonconvex loss function (4) for which we have no information.

Figure 6 displays numerical simulation results obtained for a multilayer neural network composed of 5 hidden layers and 100 neurons in each layer. For this particular

Table 5 Experiment 3 (semilinear wave equation): Training error versus number of training points N for a neural network composed of 5 hidden layers and 100 neurons per layer

	$N = 700$	$N = 2650$	$N = 5850$	$N = 10300$
$\mathcal{E}_{\text{train, int}}$	4.6×10^{-5}	5.67×10^{-5}	3.74×10^{-5}	6.36×10^{-5}
$\mathcal{E}_{\text{train, boundary}}$	6.0×10^{-5}	7.05×10^{-5}	5.76×10^{-5}	9.12×10^{-5}
$\mathcal{E}_{\text{train, initialpos}}$	1.05×10^{-4}	1.05×10^{-4}	8.44×10^{-5}	1.22×10^{-4}
$\mathcal{E}_{\text{train, initialvel}}$	5.08×10^{-2}	5.08×10^{-2}	5.0×10^{-2}	5.0×10^{-2}
$\mathcal{E}_{\text{train, finalpos}}$	3.01×10^{-6}	3.87×10^{-6}	2.18×10^{-7}	4.02×10^{-7}
$\mathcal{E}_{\text{train, initialvel}}$	1.47×10^{-5}	2.17×10^{-5}	3.27×10^{-5}	6.43×10^{-5}

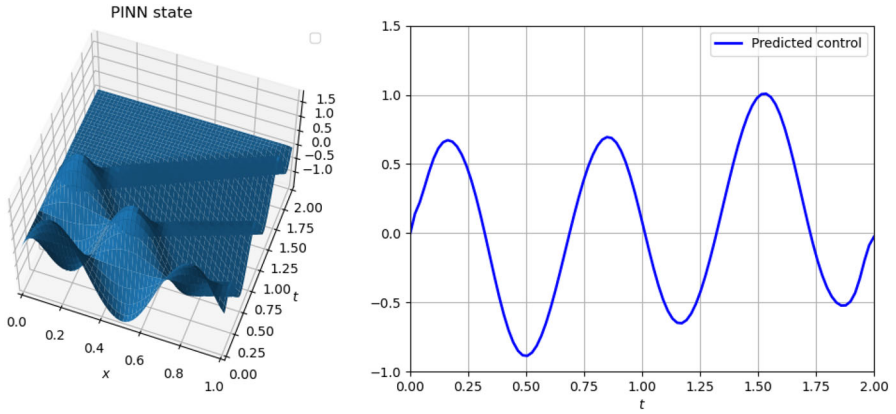

Fig. 6 Experiment 3 (semilinear wave equation). PINN (or predicted) state $\hat{y}(x, t; \boldsymbol{\theta}^*)$ (left) and PINN control $\hat{u}(t; \boldsymbol{\theta}^*)$ (right). Neural network composed of 5 hidden layers and 100 neurons in each layer. Number of training points $N = 5850$

example, no explicit solution is known so that it is not possible to check the accuracy of the method. In addition, as it was mentioned in the preceding experiment, the control is not unique. Nonetheless, comparison between Figs. 6 and 2 in [8] shows that the results are very similar.

5 Conclusions

Even though highly accurate methods are available for approximating numerically a wide range of controllability problems for PDEs, the applicability of these methods to high-dimensional problems is questionable due to the well-known *curse of dimensionality* phenomenon.

The present paper provides a first attempt to overcome this difficulty. It relies on the use of modern deep-learning-based methods, in particular on the so-called physics-informed neural networks. More precisely, a PINNs-based method has been proposed for the numerical approximation of controllability problems for PDEs both linear and

nonlinear. The problem is formulated as the minimization, in the sense of least squares, of a loss function that accounts for the residual of the PDE and its initial, boundary, and final conditions. The main novelties here with respect to more classical numerical methods in control of PDEs are as follows: (i) a feedforward neural network is used for approximating both the state and the control variables, and (ii) the method is mesh-free. In addition, it is important to emphasize that although deep-learning-based methods have found great success in many applications, no theoretical results have appeared in the literature in this field so far. In this respect, estimates for the generalization error (in both control and state variables) in terms of training and quadrature errors have been obtained in this paper. It is also proved that the training error vanishes as the size of the neural network and the number of training points go to infinity. An important feature of these theoretical results is that they apply to any controllability problem for a linear PDE and in any dimension and so PINNs qualify as a promising tool to deal with high-dimensional problems.

The accuracy in our numerical experiments is similar to the one obtained by using the PINN algorithm [34] for solving forward problems for PDEs. This is not surprising since the proposed method is a PINN-based algorithm for solving PDEs where final conditions are added to the picture and the control is obtained as the trace of the solution of the PDE.

There are many interesting questions that remain open. Some of them are:

- Since the constants that appear in our estimates on generalization error are based on energy and observability inequalities, they depend on the spatial dimension d. To what extent these estimates break the curse of dimensionality is a very interesting open problem.
- Although it was proved that training error converges to zero as the size of network and the number of the training points increase, up to the best knowledge of the authors, estimating training error is also a very challenging open problem. It is clear that training errors can be estimated *a posteriori*. Nonetheless, a posteriori estimates of training errors are, in general, not sharp as the training error incorporates errors due to the numerical approximation of highly nonconvex optimization problems whose solutions get stuck in local minima. This issue has been observed in our numerical experiments where increasing the size of the neural network and the number of training points produces a very slow decreasing of the training error.
- Proving error estimates for generalization error in the case of controllability problems for nonlinear PDEs and other types of control actions (e.g., distributed control) are also interesting open problems.

Reproducibility

The implementation of the numerical experiments presented in Sect. 4 has been performed with the user-friendly Python library DeepXDE [23], which is available at https://github.com/lululxvi/deepxde. Python scripts for the three experiments can be downloaded from https://github.com/fperiago/deepcontrol.

Acknowledgements This research was supported by Fundación Séneca (Agencia de Ciencia y Tecnología de la Región de Murcia (Spain)) under contract 20911/PI/18 and grant number 21503/EE/21 (mobility program Jiménez de la Espada). F. Periago acknowledges the hospitality of the Mathematics Department at University of California, Santa Barbara, where part of this work was carried out. The authors also thank professor Lu Lu for very fruitful comments on the use of DeepXDE.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature.

References

1. Bárcenas-Petisco, J.A.: Optimal control for neural ode in a long time horizon and applications to the classification and simultaneous controllability problems. https://hal.archives-ouvertes.fr/hal-03299270/ (2022)
2. Baydin, A.G., Pearlmutter, B.A., Radul, A.A., Siskind, J.M.: Automatic differentiation in Machine Learning: a survey. J. Mach. Learn. Res. **18**, 1–43 (2018)
3. Bardos, C., Lebeau, G., Rauch, J.: Sharp sufficient conditions for the observation, control, and stabilization of waves from the boundary. SIAM J. Control Optim. **30**(5), 1024–1065 (1992)
4. Beck, C., Martin, H., Jentzen, A., Benno, K.: An overview on deep learning-based approximation methods for partial differential equations. arXiv:2012.12348 (2021)
5. Beck, C., Becker, S., Grohs, P., Jaafari, N., Jentzen, A.: Solving the Kolmogorov PDE by means of deep learning. J. Sci. Comput. **88**(3), 1–28 (2021)
6. Blechschmidt, J., Ernst, O.G.: Three ways to solve partial differential equations with neural networks—a review. GAMM-Mitt **44**(2), 1–29 (2021)
7. Byrd, R.H., Lu, P., Nocedal, J., Zhu, C.: A limited memory algorithm for bound constrained optimization. SIAM J. Sci. Comput. **16**, 1190–1208 (1995)
8. Cavalcanti, M., Cavalcanti, V.D., Rosier, C., Rosier, L.: Numerical control of a semilinear wave equation on an interval. In: Auriol, J., Deutscher, J., Mazanti, G., Valmorbida, G. (eds.) Advances in Distributed Parameter Systems, pp. 69–89. Springer, Cham (2022)
9. Cuchiero, C., Larsson, M., Teichmann, J.: Deep neural networks, generic universal interpolation, and controlled ODEs. SIAM J. Math. Data Sci. **2**(3), 901–919 (2020)
10. Castro, C., Micu, S.: Boundary controllability of a linear semi-discrete 1-D wave equation derived from a mixed finite element method. Numer. Math. **102**(3), 413–462 (2006)
11. Ervedoza, S., Zuazua, E.: Numerical Approximation of Exact Controls for Waves. Springer Briefs in Mathematics, vol. 38. Springer, Berlin (2013)
12. Esteve, C., Geshkovski, B., Pighin, D., Zuazua, E.: Large-time asymptotics in deep learning. arXiv:2008.02491 (2021)
13. Esteve-Yagüe, C., Geshkovski, B.: Sparse approximation in learning via neural odes. arXiv:2102.13566 (2021)
14. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. AISTATS (2010)
15. Glowinski, R., Li, C., Lions, J.L.: A numerical approach to the exact boundary controllability of the wave equation (I). Dirichlet controls: description of the numerical methods. Jpn. J. Appl. Math. **7**, 1–76 (1990)
16. Gugat, M.: Optimal boundary control and boundary stabilization of hyperbolic systems. Springer Briefs in Control, Automation and Robotics. Springer (2015)

17. Han, J., Jentzen, A., Weinan, E.: Solving high-dimensional partial differential equations using deep learning. PANS **115**(34), 8505–8510 (2018)
18. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. PMLR **37**, 448–456 (2015)
19. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: 3rd International Conference on Learning Representations (2015)
20. Lazar, M., Zuazua, E.: Greedy controllability of finite dimensional linear systems. Automatica **74**, 327–340 (2016)
21. Lebeau, G., Robbiano, L.: Contrôle exact de L'equation de la chaleur. Commun. Partial Differ. Equ. **20**(1–2), 335–356 (1995)
22. Lions, J.L.: Controllabilité exacte, perturbations et stabilization de systémes distribués, vol. I. Masson, Paris (1988)
23. Lu, L., Meng, X., Mao, Z., Karniadakis, G.E.: DeepXDE: a deep learning library for solving differential equations. SIAM Rev. **63**(1), 208–228 (2021)
24. Lye, K.O., Mishra, S., Ray, D.: Deep learning observables in computational fluid dynamics. J. Comput. Phys. **410**, 109339 (2020)
25. Marín, F.J., Martínez-Frutos, J., Periago, F.: Robust averaged control of vibrations for the Bernoulli–Euler beam equation. J. Optim. Theory Appl. **174**(2), 428–454 (2017)
26. Martínez-Frutos, J., Periago, F.: Optimal control of PDEs under uncertainty. An introduction with application to optimal shape design of structures. Springer Briefs in Mathematics. BCAM Springer Briefs. Springer (2018)
27. Mishra, S., Molinaro, R.: Estimates on generalization error of physics-informed neural networks for approximating a class of inverse problems for PDEs. IMA J. Numer. Anal. **00**, 1–42 (2021)
28. Mishra, S., Molinaro, R.: Estimates on generalization error of physics-informed neural networks for approximating PDEs. IMA J. Numer. Anal. (2022). https://doi.org/10.1093/imanum/drab093
29. Münch, A.: A uniformly controllable and implicit scheme for the 1-D wave equation. Math. Model. Numer. Anal. **39**(2), 377–418 (2006)
30. Münch, A., Pedregal, P.: Numerical null controllability of the heat equation through a least squares and variational approach. Eur.J. Appl. Math. **25**(3), 277–306 (2014)
31. Münch, A., Trélat, E.: Constructive exact control of semilinear 1D wave equations by a least-squares approach. SIAM J. Control Optim. **60**(2), 652–673 (2022)
32. Pedregal, P., Periago, F., Villena, J.: A numerical method of local energy decay for the boundary controllability of time-reversible distributed parameter systems. Stud. Appl. Math. **121**(1), 27–47 (2008)
33. Pinkus, A.: Approximation theory of the MLP model in neural networks. Stud. Acta Numer. **8**, 143–195 (1999)
34. Raissi, M., Perdikaris, P., Karniadakis, G.E.: 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 (2019)
35. Ruiz-Balet, D., Zuazua, E.: Neural ode control for classification, approximation and transport. arXiv:2104.05278 (2021)
36. Ruiz-Balet, D., Affili, E., Zuazua, E.: Interpolation and approximation via momentum resnets and neural odes. Syst. Control Lett. **162**, 105182 (2022)
37. Russell, D.L.: A unified boundary controllability theory for hyperbolic and parabolic partial differential equations. Stud. Appl. Math. LI **I**(3), 189–211 (1973)
38. Shin, Y., Darbon, J., Karniadakis, G.E.: On the convergence of physics informed neural networks for linear second-order elliptic and parabolic type PDEs. Commun. Comput. Phys. **28**(5), 2042–2074 (2020)
39. Sobol', I.M.: On the distribution of points in a cube and the approximate evaluation of integrals. Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki **7**(4), 784–802 (1967)
40. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**, 1929–1958 (2014)
41. Weinan, E., Yu, B.: The deep Riesz method: a deep learning-based numerical algorithm for solving variational problems. Commun. Math. Stat. **6**, 1–12 (2018)
42. Weinan, E., Chao, M., Wojtowytsch, S., Lei, W.: Towards a mathematical understanding of neural network-based machine learning: what we know and what we don't. CSIAM Trans. Appl. Math. **1**(4), 561–615 (2020)

43. Zhang, D., Lu, L., Guo, L., Karniadakis, G.E.: Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems. J. Comput. Phys. **397**, 108850 (2019)
44. Zuazua, E.: Exact controllability for the semilinear wave equation. J. Math. Pures Appl. **69**(9), 1–31 (1990)
45. Zuazua, E.: Exact boundary controllability for the semilinear wave equation. In: Nonlinear partial Differential Equations and Their Applications. Vol. 220 of Pitman Res. Notes Math. Ser., Longman Sci. Tech., Harlow, 357–391 (1991)
46. Zuazua, E.: Propagation, observation, control and numerical approximation of waves approximated by finite difference methods. SIAM Rev. **47**(2), 197–243 (2005)
47. Zuazua, E.: Averaged control. Automatica **50**, 3077–3087 (2014)