A first step towards numerical approximation of controllability problems via Deep- Learning-based methods

Francisco Periago

Universidad Politécnica de Cartagena. Spain
Supported by Fundación Séneca-Agencia de Ciencia y Tecnología de la Región de Murcia. Mobility program Jiménez de la Espada.

Ongoing work in collaboration with Carlos J. García Cervera (University of California, Santa Barbara), and Mathieu Kessler (Universidad Politécnica de Cartagena)

Workshop on New Bridges between Mathematics and Data Science
November $8^{th} - 11^{th}$, Valladolid, Spain

Goals, outline and references

Goals

Goals, outline and references

Goals

- Explore the use of Deep-learning-based algorithms to approximate numerically controllability problems for PDEs

Goals, outline and references

Goals

- Explore the use of Deep-learning-based algorithms to approximate numerically controllability problems for PDEs
- Provide error estimates for the so-called generalization error

Goals, outline and references

Goals

- Explore the use of Deep-learning-based algorithms to approximate numerically controllability problems for PDEs
- Provide error estimates for the so-called generalization error

Outline

Goals, outline and references

Goals

- Explore the use of Deep-learning-based algorithms to approximate numerically controllability problems for PDEs
- Provide error estimates for the so-called generalization error

Outline

- Toy model: boundary controllability of the linear wave equation

Goals, outline and references

Goals

- Explore the use of Deep-learning-based algorithms to approximate numerically controllability problems for PDEs
- Provide error estimates for the so-called generalization error

Outline

- Toy model: boundary controllability of the linear wave equation
- We adapt Physics-Informed-Neural-Networks (PINNs) to approximate numerically the above toy model

Goals, outline and references

Goals

- Explore the use of Deep-learning-based algorithms to approximate numerically controllability problems for PDEs
- Provide error estimates for the so-called generalization error

Outline

- Toy model: boundary controllability of the linear wave equation
- We adapt Physics-Informed-Neural-Networks (PINNs) to approximate numerically the above toy model
- Analysis of error estimates for generalization error

Goals, outline and references

Goals

- Explore the use of Deep-learning-based algorithms to approximate numerically controllability problems for PDEs
- Provide error estimates for the so-called generalization error

Outline

- Toy model: boundary controllability of the linear wave equation
- We adapt Physics-Informed-Neural-Networks (PINNs) to approximate numerically the above toy model
- Analysis of error estimates for generalization error
- Numerical implementation via DeepXDE Python library

Goals, outline and references

Goals

- Explore the use of Deep-learning-based algorithms to approximate numerically controllability problems for PDEs
- Provide error estimates for the so-called generalization error

Outline

- Toy model: boundary controllability of the linear wave equation
- We adapt Physics-Informed-Neural-Networks (PINNs) to approximate numerically the above toy model
- Analysis of error estimates for generalization error
- Numerical implementation via DeepXDE Python library
- Numerical simulation results

Goals, outline and references

Goals

- Explore the use of Deep-learning-based algorithms to approximate numerically controllability problems for PDEs
- Provide error estimates for the so-called generalization error

Outline

- Toy model: boundary controllability of the linear wave equation
- We adapt Physics-Informed-Neural-Networks (PINNs) to approximate numerically the above toy model
- Analysis of error estimates for generalization error
- Numerical implementation via DeepXDE Python library
- Numerical simulation results
- Extension to other PDEs, mainly where high-dimensionality plays a role

Goals, outline and references

Goals

- Explore the use of Deep-learning-based algorithms to approximate numerically controllability problems for PDEs
- Provide error estimates for the so-called generalization error

Outline

- Toy model: boundary controllability of the linear wave equation
- We adapt Physics-Informed-Neural-Networks (PINNs) to approximate numerically the above toy model
- Analysis of error estimates for generalization error
- Numerical implementation via DeepXDE Python library
- Numerical simulation results
- Extension to other PDEs, mainly where high-dimensionality plays a role

References

Raisi, M., Perdikaris, P. and Karniadakis, G: Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Physics **378**, 686-707, 2019.

Lu, L., Meng, X., Mao, Z. and Karniadakis, G.: DeepXDE: A Deep Learning Library for Solving Differential Equations, SIAM Review, **63** (1), 208-228, 2021.

A toy model: controllability of the linear wave equation

The exact controllability problem: given initial data $(y^0(x), y^1(x))$ and a positive time $T > 0$ find a boundary control $u(t)$ such that the solution $y(x,t)$ of the system

$$\begin{cases} y_{tt} = c^2 y_{xx}, & \text{in } (0,1)\times(0,T) \\ y(x,0) = y^0(x), & \text{in } (0,1) \\ y_t(x,0) = y^1(x) & \text{in } (0,1) \\ y(0,t) = 0, \quad y(1,t) = u(t) & \text{on } (0,T) \end{cases} \tag{1}$$

A toy model: controllability of the linear wave equation

The exact controllability problem: given initial data $(y^0(x), y^1(x))$ and a positive time $T > 0$ find a boundary control $u(t)$ such that the solution $y(x,t)$ of the system

$$\begin{cases} y_{tt} = c^2 y_{xx}, & \text{in } (0,1) \times (0,T) \\ y(x,0) = y^0(x), & \text{in } (0,1) \\ y_t(x,0) = y^1(x) & \text{in } (0,1) \\ y(0,t) = 0, \quad y(1,t) = u(t) & \text{on } (0,T) \end{cases} \tag{1}$$

satisfies

$$y(x,T) = y_t(x,T) = 0 \quad \text{in } (0,1). \tag{2}$$

A toy model: controllability of the linear wave equation

The exact controllability problem: given initial data $(y^0(x), y^1(x))$ and a positive time $T > 0$ find a boundary control $u(t)$ such that the solution $y(x,t)$ of the system

$$\begin{cases} y_{tt} = c^2 y_{xx}, & \text{in } (0,1) \times (0,T) \\ y(x,0) = y^0(x), & \text{in } (0,1) \\ y_t(x,0) = y^1(x) & \text{in } (0,1) \\ y(0,t) = 0, \quad y(1,t) = u(t) & \text{on } (0,T) \end{cases} \tag{1}$$

satisfies

$$y(x,T) = y_t(x,T) = 0 \quad \text{in } (0,1). \tag{2}$$

For $T = 2/c$ this problem has the explicit solution

$$u(t) = \begin{cases} \frac{1}{2} y^0(1-ct) + \frac{1}{2c} \int_{1-ct}^{1} y^1(s)\, ds & 0 \le t \le 1/c \\ \\ -\frac{1}{2} y^0(ct-1) + \frac{1}{2c} \int_{ct-1}^{1} y^1(s)\, ds & 1/c \le t \le 2/c \end{cases} \tag{3}$$

Numerical approximation of the control via PINNs

Main steps of the PINNs algorithm

Numerical approximation of the control via PINNs

Main steps of the PINNs algorithm

1. **design an artificial neural network $\hat{y}(x, t; \theta)$ as a surrogate of the true solution $y(x, t)$**

Numerical approximation of the control via PINNs

Main steps of the PINNs algorithm

1. **design an artificial neural network $\hat{y}(x, t; \theta)$ as a surrogate of the true solution $y(x, t)$**
2. **Choose a training dataset in the space-time domain $(0, 1) \times (0, T)$**

Numerical approximation of the control via PINNs

Main steps of the PINNs algorithm

1. design an artificial neural network $\hat{y}(x, t; \theta)$ as a surrogate of the true solution $y(x, t)$
2. Choose a training dataset in the space-time domain $(0, 1) \times (0, T)$
3. Consider a loss function: a weighted summation of the L^2 norm of residuals for the equation, boundary, initial and final conditions

Numerical approximation of the control via PINNs

Main steps of the PINNs algorithm

1. design an artificial neural network $\hat{y}(x, t; \theta)$ as a surrogate of the true solution $y(x, t)$
2. Choose a training dataset in the space-time domain $(0, 1) \times (0, T)$
3. Consider a loss function: a weighted summation of the L^2 norm of residuals for the equation, boundary, initial and final conditions
4. Train the network by minimizing the loss function defined in the previous step

Numerical approximation of the control via PINNs

Main steps of the PINNs algorithm

1. **design an artificial neural network $\hat{y}(x, t; \boldsymbol{\theta})$ as a surrogate of the true solution $y(x, t)$**
2. **Choose a training dataset in the space-time domain $(0, 1) \times (0, T)$**
3. **Consider a loss function: a weighted summation of the L^2 norm of residuals for the equation, boundary, initial and final conditions**
4. **Train the network by minimizing the loss function defined in the previous step**

   From the training process, optimal parameters $\boldsymbol{\theta}$ 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 $u(t)$, which is approximated as the trace of $\hat{y}(x, t; \boldsymbol{\theta})$ on the boundary $x = 1$, i.e., the surrogate control $\hat{u}(t; \boldsymbol{\theta}) = \hat{y}(1, t; \boldsymbol{\theta})$

Numerical approximation of the control via PINNs: the details

Numerical approximation of the control via PINNs: the details

Step 1: Neural network.

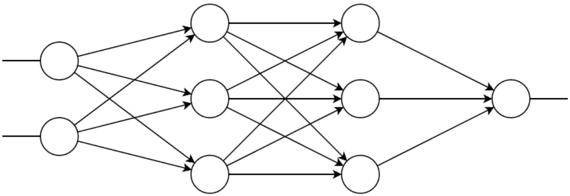
Numerical approximation of the control via PINNs: the details

Step 1: Neural network. We consider a Multilayer Perceptron (MLP) with two input canals $\boldsymbol{x} = (x, t) \in \mathbb{R}^2$ and an scalar output $\hat{y}$. Precisely, $\hat{y}\,(x, t; \boldsymbol{\theta})$ is constructed as

$$\begin{cases} \text{input layer:} & \mathcal{N}^0(\boldsymbol{x}) = \boldsymbol{x} = (x, t) \in \mathbb{R}^2 \\ \text{hidden layers:} & \mathcal{N}^\ell(\boldsymbol{x}) = \sigma\left(\boldsymbol{W}^\ell \mathcal{N}^{\ell-1}(\boldsymbol{x}) + \boldsymbol{b}^\ell\right) \in \mathbb{R}^{N_\ell} \\ \text{output layer:} & \hat{y}\,(\boldsymbol{x}; \boldsymbol{\theta}) = \mathcal{N}^L(\boldsymbol{x}) = \boldsymbol{W}^L \mathcal{N}^{L-1}(\boldsymbol{x}) + \boldsymbol{b}^L \in \mathbb{R} \end{cases} \tag{4}$$

where

- $\mathcal{N}^\ell(\boldsymbol{x}) : \mathbb{R}^{d_{in}} \to \mathbb{R}^{d_{out}}$ 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 a smooth activation function, e.g. the hyperbolic tangent $\sigma(s) = \tanh(s)$.



Numerical approximation of the control via PINNs: the details

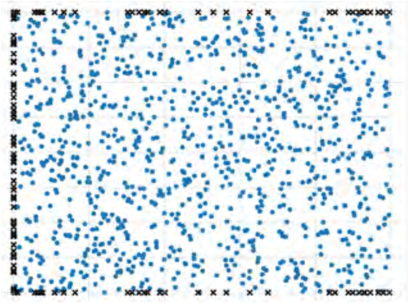
Step 2: Training dataset.

Numerical approximation of the control via PINNs: the details

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 $\mathcal{T}_{x=0} \subset \{0\} \times (0, T)$, $\mathcal{T}_{t=0} \subset (0, 1) \times \{0\}$, $\mathcal{T}_{t=T} \subset (0, 1) \times \{T\}$. Thus, $\mathcal{T} = \mathcal{T}_{\text{int}} \cup \mathcal{T}_{x=0} \cup \mathcal{T}_{t=0} \cup \mathcal{T}_{t=T}$. 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 $x = 0$, and N_0 and N_T stand for the number of points in $\mathcal{T}_{t=0}$ and $\mathcal{T}_{t=T}$, respectively.

Numerical approximation of the control via PINNs: the details

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 $\mathcal{T}_{x=0} \subset \{0\} \times (0, T)$, $\mathcal{T}_{t=0} \subset (0,1) \times \{0\}$, $\mathcal{T}_{t=T} \subset (0,1) \times \{T\}$. Thus, $\mathcal{T} = \mathcal{T}_{\text{int}} \cup \mathcal{T}_{x=0} \cup \mathcal{T}_{t=0} \cup \mathcal{T}_{t=T}$. 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 $x = 0$, and N_0 and N_T stand for the number of points in $\mathcal{T}_{t=0}$ and $\mathcal{T}_{t=T}$, respectively.



Step 3: Loss function.

Numerical approximation of the control via PINNs: the details

Step 3: Loss function. It is composed of the following six terms:

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

Numerical approximation of the control via PINNs: the details

Step 4: Training process. The final step at the PINN algorithm amounts to minimize the loss function

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta};\mathcal{T}) \quad &= \mathcal{L}_{\text{int}}(\boldsymbol{\theta};\mathcal{T}_{\text{int}}) \\ &+\mathcal{L}_{x=0}(\boldsymbol{\theta};\mathcal{T}_{x=0}) \\ &+\mathcal{L}^{\text{pos}}_{t=0}(\boldsymbol{\theta};\mathcal{T}_{t=0}) + \mathcal{L}^{\text{vel}}_{t=0}(\boldsymbol{\theta};\mathcal{T}_{t=0}) \\ &+\mathcal{L}^{\text{pos}}_{t=T}(\boldsymbol{\theta};\mathcal{T}_{t=T}) + \mathcal{L}^{\text{vel}}_{t=T}(\boldsymbol{\theta};\mathcal{T}_{t=T}).\end{aligned} \tag{5}$$

i.e., we compute

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta};\mathcal{T}). \tag{6}$$

Numerical approximation of the control via PINNs: the details

Step 4: Training process. The final step at the PINN algorithm amounts to minimize the loss function

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta};\mathcal{T}) \quad &= \mathcal{L}_{\text{int}}(\boldsymbol{\theta};\mathcal{T}_{\text{int}}) \\ &+\mathcal{L}_{x=0}(\boldsymbol{\theta};\mathcal{T}_{x=0}) \\ &+\mathcal{L}^{\text{pos}}_{t=0}(\boldsymbol{\theta};\mathcal{T}_{t=0}) + \mathcal{L}^{\text{vel}}_{t=0}(\boldsymbol{\theta};\mathcal{T}_{t=0}) \\ &+\mathcal{L}^{\text{pos}}_{t=T}(\boldsymbol{\theta};\mathcal{T}_{t=T}) + \mathcal{L}^{\text{vel}}_{t=T}(\boldsymbol{\theta};\mathcal{T}_{t=T}).\end{aligned} \tag{5}$$

i.e., we compute

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta};\mathcal{T}). \tag{6}$$

The descent algorithm **ADAM** (Adaptive with Moment) is chosen for numerical implementation. Automatic Differentation **AD**, which is included in TensorFlow, is used for computation of gradients.

Numerical approximation of the control via PINNs: the details

Step 4: Training process. The final step at the PINN algorithm amounts to minimize the loss function

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta};\mathcal{T}) \quad &= \mathcal{L}_{\text{int}}(\boldsymbol{\theta};\mathcal{T}_{\text{int}}) \\ &+\mathcal{L}_{x=0}(\boldsymbol{\theta};\mathcal{T}_{x=0}) \\ &+\mathcal{L}^{\text{pos}}_{t=0}(\boldsymbol{\theta};\mathcal{T}_{t=0}) + \mathcal{L}^{\text{vel}}_{t=0}(\boldsymbol{\theta};\mathcal{T}_{t=0}) \\ &+\mathcal{L}^{\text{pos}}_{t=T}(\boldsymbol{\theta};\mathcal{T}_{t=T}) + \mathcal{L}^{\text{vel}}_{t=T}(\boldsymbol{\theta};\mathcal{T}_{t=T}).\end{aligned} \tag{5}$$

i.e., we compute

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta};\mathcal{T}). \tag{6}$$

The descent algorithm **ADAM** (Adaptive with Moment) is chosen for numerical implementation. Automatic Differentation **AD**, which is included in TensorFlow, is used for computation of gradients.
The approximation $\hat{u}(t;\boldsymbol{\theta}^*)$ of the control $u(t)$ is then obtained as the restriction of $\hat{y}(x,t;\boldsymbol{\theta}^*)$ to the boundary $x=1$, i.e.

$$\hat{u}(t;\boldsymbol{\theta}^*) = \hat{y}(1,t;\boldsymbol{\theta}^*), \quad 0 \le t \le T. \tag{7}$$

Error estimates for generalization error

Generalization error for the control

$$\mathcal{E}_{\text{gener}}(u) := \|u - \hat{u}\|_{L^2(0,T)}, \tag{8}$$

where $u = u(t)$ is the exact control of the continuous problem and $\hat{u} = \hat{u}(t; \boldsymbol{\theta}^*)$ is its numerical approximation via PINN algo.

Error estimates for generalization error

Generalization error for the control

$$\mathcal{E}_{\text{gener}}(u) := \|u - \hat{u}\|_{L^2(0,T)}, \tag{8}$$

where $u = u(t)$ is the exact control of the continuous problem and $\hat{u} = \hat{u}(t; \boldsymbol{\theta}^*)$ is its numerical approximation via PINN algo.

Our goal is to get estimations for generalization error in terms of error estimates for quadrature and the so-called training error.

Error estimates for generalization error

Generalization error for the control

$$\mathcal{E}_{\text{gener}}(u) := \|u - \hat{u}\|_{L^2(0,T)}, \tag{8}$$

where $u = u(t)$ is the exact control of the continuous problem and $\hat{u} = \hat{u}(t; \boldsymbol{\theta}^*)$ is its numerical approximation via PINN algo.
Our goal is to get estimations for generalization error in terms of error estimates for quadrature and the so-called training error.
Quadrature errors:

$$|\overline{f} - \overline{f}_N| \leq C_q(d) N^{-\alpha}, \quad \alpha > 0, \tag{9}$$

where

$$\overline{f} := \int_{\mathcal{D}} f(x)\, dx, \quad \overline{f}_N := \sum_{j=1}^{N} w_j f(x_j)$$

Error estimates for generalization error

Generalization error for the control

$$\mathcal{E}_{\text{gener}}(u) := \|u - \hat{u}\|_{L^2(0,T)}, \tag{8}$$

where $u = u(t)$ is the exact control of the continuous problem and $\hat{u} = \hat{u}(t; \boldsymbol{\theta}^*)$ is its numerical approximation via PINN algo.
Our goal is to get estimations for generalization error in terms of error estimates for quadrature and the so-called training error.
Quadrature errors:

$$|\overline{f} - \overline{f}_N| \leq C_q(d) N^{-\alpha}, \quad \alpha > 0, \tag{9}$$

where

$$\overline{f} := \int_{\mathcal{D}} f(x)\, dx, \quad \overline{f}_N := \sum_{j=1}^{N} w_j f(x_j)$$

Training error: $\mathcal{E}_{\text{train}} := \mathcal{L}(\boldsymbol{\theta}^*; \mathcal{T})$

$$\begin{cases} \mathcal{E}_{\text{train, int}} & = \mathcal{L}_{\text{int}}(\boldsymbol{\theta}^*; \mathcal{T}_{\text{int}}) \\ \mathcal{E}_{\text{train, boundary}} & = \mathcal{L}_{x=0}(\boldsymbol{\theta}^*; \mathcal{T}_{x=0}) \\ \mathcal{E}_{\text{train, initialpos}} & = \mathcal{L}^{\text{pos}}_{t=0}(\boldsymbol{\theta}^*; \mathcal{T}_{t=0}) \\ \mathcal{E}_{\text{train, initialvel}} & = \mathcal{L}^{\text{vel}}_{t=0}(\boldsymbol{\theta}^*; \mathcal{T}_{t=0}) \\ \mathcal{E}_{\text{train, finalpos}} & = \mathcal{L}^{\text{pos}}_{t=T}(\boldsymbol{\theta}^*; \mathcal{T}_{t=T}) \\ \mathcal{E}_{\text{train, finalvel}} & = \mathcal{L}^{\text{vel}}_{t=T}(\boldsymbol{\theta}^*; \mathcal{T}_{t=T}). \end{cases} \tag{10}$$

Error estimates for generalization error

Theorem

Let $y = y(x,t) \in C^k\left(\overline{Q_T}\right)$, $k \geq 2$, *be the unique classical solution of* (1)-(2) *and let* $\hat{y} = \hat{y}(x,t;\boldsymbol{\theta}^*)$ *its PINN approximation. Let* $u = u(t)$ *and* $\hat{u} = \hat{u}\left(t;\boldsymbol{\theta}^*\right)$ *be the exact control of the continuous system* (1)-(2) *and its PINN approximation, respectively. Then, the following estimate for generalization error holds*

$$\begin{aligned}
\mathcal{E}_{gener}(u) \quad &\lesssim \mathcal{E}_{train,\ int} + CN_{int}^{-\alpha/2} \\
&+\mathcal{E}_{train,\ boundary} + CN_{b}^{-\alpha/2} \\
&+\mathcal{E}_{train,\ initialpos} + CN_{0}^{-\alpha/2} \\
&+\mathcal{E}_{train,\ initialvel} + CN_{0}^{-\alpha/2} \\
&+\mathcal{E}_{train,\ finalpos} + CN_{T}^{-\alpha/2} \\
&+\mathcal{E}_{train,\ finalvel} + CN_{T}^{-\alpha/2}
\end{aligned} \qquad (11)$$

Error estimates for generalization error

Theorem

Let $y = y(x,t) \in C^k\left(\overline{Q_T}\right)$, $k \geq 2$, be the unique classical solution of (1)-(2) *and let $\hat{y} = \hat{y}(x,t;\boldsymbol{\theta}^*)$ its PINN approximation. Let $u = u(t)$ and $\hat{u} = \hat{u}\left(t;\boldsymbol{\theta}^*\right)$ be the exact control of the continuous system* (1)-(2) *and its PINN approximation, respectively. Then, the following estimate for generalization error holds*

$$\begin{aligned} \mathcal{E}_{gener}(u) \quad & \lesssim \mathcal{E}_{train,\ int} + CN_{int}^{-\alpha/2} \\ & + \mathcal{E}_{train,\ boundary} + CN_{b}^{-\alpha/2} \\ & + \mathcal{E}_{train,\ initialpos} + CN_{0}^{-\alpha/2} \\ & + \mathcal{E}_{train,\ initialvel} + CN_{0}^{-\alpha/2} \\ & + \mathcal{E}_{train,\ finalpos} + CN_{T}^{-\alpha/2} \\ & + \mathcal{E}_{train,\ finalvel} + CN_{T}^{-\alpha/2} \end{aligned} \tag{11}$$

Main ingredients in the proof are observability inequalities and energy estimates

Error estimates for generalization error

Lemma (E. Fernández-Cara and E. Zuazua)

Let $T \geq 2$. Given initial and final conditions $(z_0^0, z_0^1), (z_T^0, z_T^1) \in L^2(0,1) \times H^{-1}(0,1)$, there exists a control function $v \in L^2(0,T)$ such that the solution $z(x,t)$ of the system

$$\begin{cases} z_{tt} = z_{xx}, & \text{in } Q_T \\ z(x,0) = z_0^0(x), & \text{in } (0,1) \\ z_t(x,0) = z_0^1(x) & \text{in } (0,1) \\ z(0,t) = 0, \quad z(1,t) = v(t) & \text{on } (0,T) \end{cases} \tag{12}$$

satisfies

$$z(x,T) = z_T^0(x), \quad z_t(x,T) = z_T^1(x,T), \quad x \in (0,1). \tag{13}$$

Moreover,

$$\|v\|_{L^2(0,T)} \leq C\left(\|z_0^0\|_{L^2(0,1)} + \|z_0^1\|_{H^{-1}(0,1)} + \|z_T^0\|_{L^2(0,1)} + \|z_T^1\|_{H^{-1}(0,1)}\right), \tag{14}$$

for a positive constant $C = C(T)$, which does not depend on the initial and final data.

Error estimates for generalization error

Lemma

Consider the non-homogeneous system

$$\begin{cases} z_{tt} = z_{xx} + f(x,t), & \text{in } Q_T \\ z(x,0) = z_0^0(x), & \text{in } (0,1) \\ z_t(x,0) = z_0^1(x) & \text{in } (0,1) \\ z(0,t) = g_0(t), \quad z(1,t) = g_1(t) & \text{on } (0,T) \end{cases}$$

Then, there exists a positive constant C such that

$$\|z\|_{C\left(0,T;L^2(0,1)\right)} + \|z_t\|_{C\left(0,T;H^{-1}(0,1)\right)} \\ \leq C\left(\|z_0^0\|_{L^2(0,1)} + \|z_0^1\|_{H^{-1}(0,1)} + \|g_0\|_{L^2(0,T)} + \|g_1\|_{L^2(0,T)} + \|f\|_{L^2\left(0,T;L^2(0,1)\right)}\right)$$

Proof of theorem on generalization error.

Proof of theorem on generalization error.

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

Error estimates for generalization error

Proof of theorem on generalization error.

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

$$\begin{cases} \overline{y}_{tt} - \overline{y}_{xx} = \hat{y}_{tt} - \hat{y}_{xx}, & \text{in } Q_T \\ \overline{y}(x,0) = y^0(x) - \hat{y}(x,0), & \text{in } (0,1) \\ \overline{y}_t(x,0) = y^1(x) - \hat{y}_t(x,0) & \text{in } (0,1) \\ \overline{y}(x,T) = \hat{y}(x,T), & \text{in } (0,1) \\ \overline{y}_t(x,T) = \hat{y}_t(x,T) & \text{in } (0,1) \\ \overline{y}(0,t) = \hat{y}(0,t), \quad \overline{y}(1,t) = u(t) - \hat{y}(1,t) & \text{on } (0,T). \end{cases} \tag{15}$$

Error estimates for generalization error

Proof of theorem on generalization error.

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

$$\begin{cases} \overline{y}_{tt} - \overline{y}_{xx} = \hat{y}_{tt} - \hat{y}_{xx}, & \text{in } Q_T \\ \overline{y}(x,0) = y^0(x) - \hat{y}(x,0), & \text{in } (0,1) \\ \overline{y}_t(x,0) = y^1(x) - \hat{y}_t(x,0) & \text{in } (0,1) \\ \overline{y}(x,T) = \hat{y}(x,T), & \text{in } (0,1) \\ \overline{y}_t(x,T) = \hat{y}_t(x,T) & \text{in } (0,1) \\ \overline{y}(0,t) = \hat{y}(0,t), \quad \overline{y}(1,t) = u(t) - \hat{y}(1,t) & \text{on } (0,T). \end{cases} \tag{15}$$

$\overline{y}(x,t;\boldsymbol{\theta})$ is decomposed as $\overline{y} = \overline{y}^1 + \overline{y}^2$, where

$$\begin{cases} \overline{y}^1_{tt} - \overline{y}^1_{xx} = 0, & \text{in } Q_T \\ \overline{y}^1(x,0) = y^0(x) - \hat{y}(x,0), & \text{in } (0,1) \\ \overline{y}^1_t(x,0) = y^1(x) - \hat{y}_t(x,0) & \text{in } (0,1) \\ \overline{y}^1(0,t) = 0, \quad \overline{y}^1(1,t) = u(t) - \hat{y}(1,t) & \text{on } (0,T). \end{cases} \tag{16}$$

$$\begin{cases} \overline{y}^2_{tt} - \overline{y}^2_{xx} = \hat{y}_{tt} - \hat{y}_{xx}, & \text{in } Q_T \\ \overline{y}^2(x,0) = 0, \quad \overline{y}^2_t(x,0) = 0 & \text{in } (0,1) \\ \overline{y}^2(x,T) = \hat{y}(x,T) - \overline{y}^1(x,T), & \text{in } (0,1) \\ \overline{y}^2_t(x,T) = \hat{y}_t(x,T) - \overline{y}^1_t(x,T), & \text{in } (0,1) \\ \overline{y}^2(0,t) = \hat{y}(0,t), \quad \overline{y}^2(1,t) = 0 & \text{on } (0,T). \end{cases} \tag{17}$$

Error estimates for generalization error

Proof of theorem on generalization error.

By applying the observability inequality to system (16) and the energy estimate to (17),

$$\|u - \hat{u}\|_{L^2(0,T)}$$

$$\lesssim \|y^0 - \hat{y}(\cdot,0)\|_{L^2(0,1)} + \|y^1 - \hat{y}_t(\cdot,0)\|_{H^{-1}(0,1)} + \|\overline{y}^1(\cdot,T)\|_{L^2(0,1)} + \|\overline{y}^1_t(\cdot,T)\|_{H^{-1}(0,1)}$$

Error estimates for generalization error

Proof of theorem on generalization error.

By applying the observability inequality to system (16) and the energy estimate to (17),

$$\|u-\hat{u}\|_{L^2(0,T)}$$

$$\lesssim \|y^0-\hat{y}(\cdot,0)\|_{L^2(0,1)}+\|y^1-\hat{y}_t(\cdot,0)\|_{H^{-1}(0,1)}+\|\overline{y}^1(\cdot,T)\|_{L^2(0,1)}+\|\overline{y}_t^1(\cdot,T)\|_{H^{-1}(0,1)}$$

$$\lesssim \|y^0-\hat{y}(\cdot,0)\|_{L^2(0,1)}+\|y^1-\hat{y}_t(\cdot,0)\|_{L^2(0,1)}+\|\hat{y}(\cdot,T)\|_{L^2(0,1)}+\|\hat{y}_t(\cdot,T)\|_{L^2(0,1)}$$

$$+\|\overline{y}^2(\cdot,T)\|_{L^2(0,1)}+\|\overline{y}_t^2(\cdot,T)\|_{H^{-1}(0,1)}$$

Error estimates for generalization error

Proof of theorem on generalization error.

By applying the observability inequality to system (16) and the energy estimate to (17),

$$\begin{aligned}
&\|u-\hat{u}\|_{L^2(0,T)}\\
&\lesssim \|y^0-\hat{y}(\cdot,0)\|_{L^2(0,1)}+\|y^1-\hat{y}_t(\cdot,0)\|_{H^{-1}(0,1)}+\|\overline{y}^1(\cdot,T)\|_{L^2(0,1)}+\|\overline{y}^1_t(\cdot,T)\|_{H^{-1}(0,1)}\\
&\lesssim \|y^0-\hat{y}(\cdot,0)\|_{L^2(0,1)}+\|y^1-\hat{y}_t(\cdot,0)\|_{L^2(0,1)}+\|\hat{y}(\cdot,T)\|_{L^2(0,1)}+\|\hat{y}_t(\cdot,T)\|_{L^2(0,1)}\\
&+\|\overline{y}^2(\cdot,T)\|_{L^2(0,1)}+\|\overline{y}^2_t(\cdot,T)\|_{H^{-1}(0,1)}\\
&\lesssim \|y^0-\hat{y}(\cdot,0)\|_{L^2(0,1)}+\|y^1-\hat{y}_t(\cdot,0)\|_{L^2(0,1)}+\|\hat{y}(\cdot,T)\|_{L^2(0,1)}+\|\hat{y}_t(\cdot,T)\|_{L^2(0,1)}\\
&+\|\hat{y}\,(0,\cdot)\,\|_{L^2(0,T)}+\|\hat{y}_{tt}-\hat{y}_{xx}\|_{L^2(0,T;L^2(0,1))}.
\end{aligned} \tag{18}$$

The result then follows by applying estimates error for quadrature (9).

A numerical experiment (preliminary results)

$$\begin{cases} y_{tt} = y_{xx}, & \text{in } (0,1) \times (0,2) \\ y(x,0) = \sin(\pi x), & \text{in } (0,1) \\ y_t(x,0) = 0 & \text{in } (0,1) \\ y(0,t) = 0, \quad y(1,t) = u(t) & \text{on } (0,2) \\ y(x,2) = y_t(x,2) = 0 & \text{in } (0,1) \end{cases}$$

A numerical experiment (preliminary results)

$$\begin{cases} y_{tt} = y_{xx}, & \text{in } (0,1) \times (0,2) \\ y(x,0) = \sin(\pi x), & \text{in } (0,1) \\ y_t(x,0) = 0 & \text{in } (0,1) \\ y(0,t) = 0, \quad y(1,t) = u(t) & \text{on } (0,2) \\ y(x,2) = y_t(x,2) = 0 & \text{in } (0,1) \end{cases}$$

Numerical implentation via DeepXDE Python library

- Multilayer perceptron with 4 hidden layers and 50 neurons in each layer
- Activation function: tanh
- Dataset for training: Sobol
- Optimizer: ADAM + L-BFGS-B
- Initializer: Glorot uniform

A numerical experiment (preliminary results)

$$\begin{cases} y_{tt} = y_{xx}, & \text{in } (0,1) \times (0,2) \\ y(x,0) = \sin(\pi x), & \text{in } (0,1) \\ y_t(x,0) = 0 & \text{in } (0,1) \\ y(0,t) = 0, \quad y(1,t) = u(t) & \text{on } (0,2) \\ y(x,2) = y_t(x,2) = 0 & \text{in } (0,1) \end{cases}$$

Numerical implentation via DeepXDE Python library

- Multilayer perceptron with 4 hidden layers and 50 neurons in each layer
- Activation function: tanh
- Dataset for training: Sobol
- Optimizer: ADAM + L-BFGS-B
- Initializer: Glorot uniform

Table: Summary of results for training errors and for 500 interior points and 50 boundary points.

$\|\hat{y}_{tt} - \hat{y}_{xx}\|$	$\|\hat{y}(0,\cdot)\|$	$\|y^0 - \hat{y}(\cdot,0)\|$	$\|y^1 - \hat{y}_t(\cdot,0)\|$	$\|\hat{y}(\cdot,T)\|$	$\|\hat{y}_t(\cdot,T)\|$
8.8×10^{-6}	2.1×10^{-6}	1.1×10^{-6}	5.7×10^{-9}	3.4×10^{-8}	7.3×10^{-8}

A numerical experiment (preliminary results)

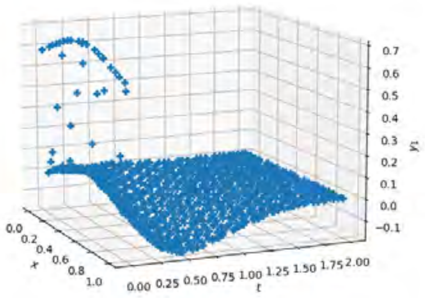


Figure: Predicted solution.

Scope of the proposed methodology

- The same approach applies to many other controllability problems for PDEs both linear and nonlinear.

Scope of the proposed methodology

- The same approach applies to many other controllability problems for PDEs both linear and nonlinear.
- The method may be adapted to averaged control of parametric PDEs where the number of parameters may be large.

Scope of the proposed methodology

- The same approach applies to many other controllability problems for PDEs both linear and nonlinear.
- The method may be adapted to averaged control of parametric PDEs where the number of parameters may be large.
- A challenging and high dimensional problem is to design a ML algorithm to approximate the initial data to control mapping

$$\left(u^0, u^1\right) \mapsto u(t)$$

Scope of the proposed methodology

- The same approach applies to many other controllability problems for PDEs both linear and nonlinear.
- The method may be adapted to averaged control of parametric PDEs where the number of parameters may be large.
- A challenging and high dimensional problem is to design a ML algorithm to approximate the initial data to control mapping

$$\left(u^0, u^1\right) \mapsto u(t)$$

.... Muchas gracias ....