Solving Neural Field Equations using Physics Informed Neural Networks

Weronika Wojtak[1, 2, 3] ✉

Estela Bicho[1]

Wolfram Erlhagen[2]

[1] Research Centre Algoritmi, University of Minho, Guimarães, Portugal

[2] Research Centre of Mathematics, University of Minho, Guimarães, Portugal

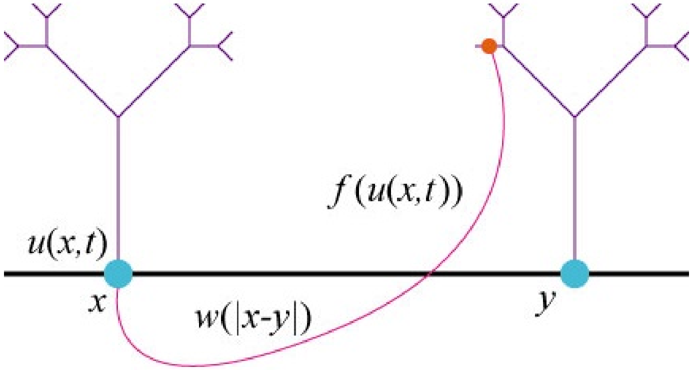
[3] Centro de Computação Gráfica, Guimarães, Portugal

✉ w.wojtak@dei.uminho.pt

Neural Field Equations

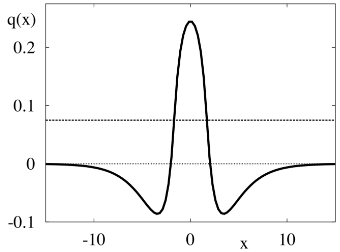
- Describe the **large-scale dynamics of neuronal populations** in the cortex.
- Typically formalized by integro-differential equations (IDEs):

$$\frac{\partial u(x,t)}{\partial t} = -u(x,t) + \int_{\Omega} w(|x-y|)\, f(u(y,t)-\kappa)\, dy$$

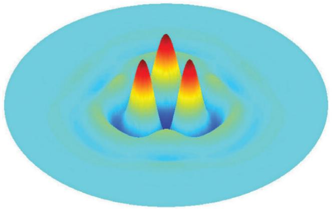


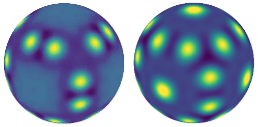
Neural Field Equations

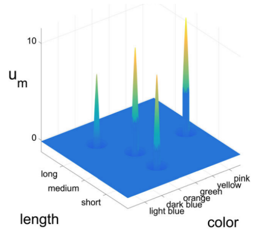
- Applications in **neuroscience** and **robotics**.



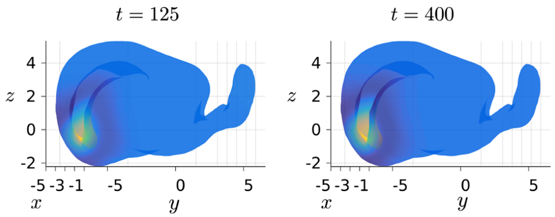
Coombes, 2005


Owen et al., 2007

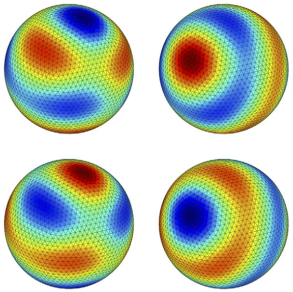

Avitabile et al., 2017

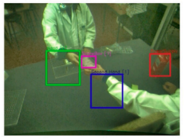


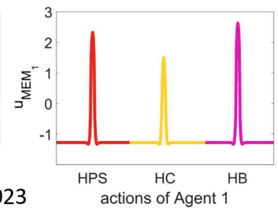
Wojtak et al., 2021



Martin et al., 2018


Visser et al., 2017

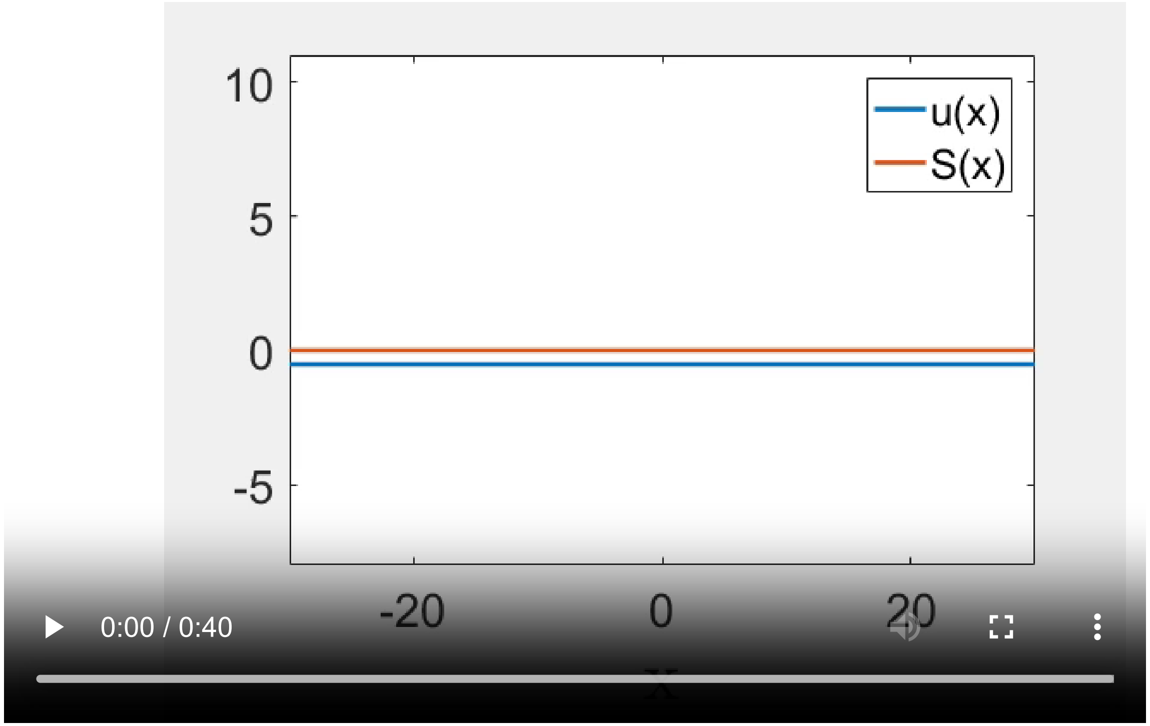




Wojtak et al., 2023

Neural Field Equations

The mathematical analysis: the existence and stability of **localized activation patterns (bumps)**.

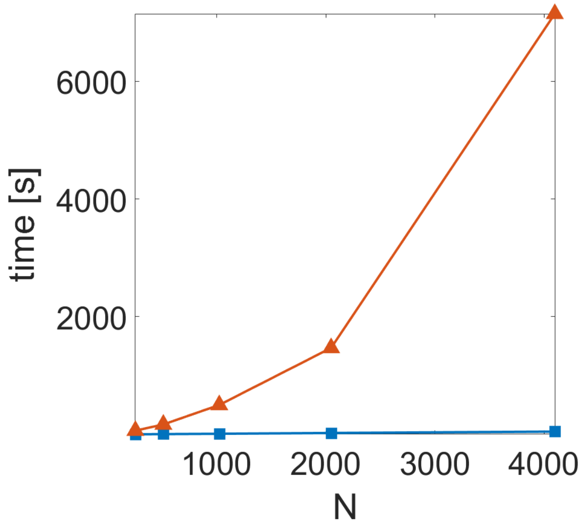


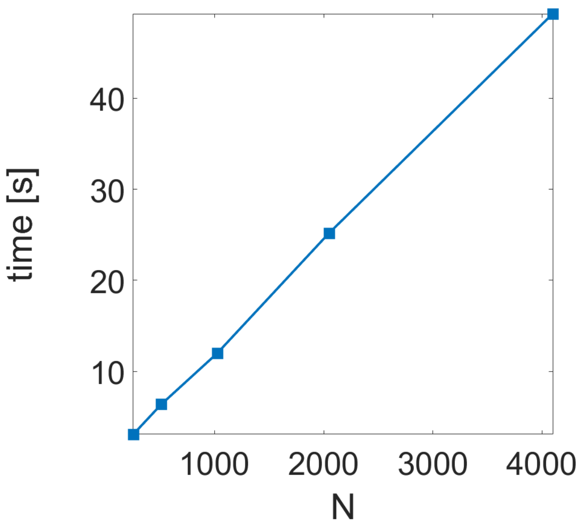
Neural Field Equations

- In general, no closed-form expressions for these patterns, the solutions must be **approximated numerically.**
- Significant computational effort, mostly due to the discretization of the **spatial convolution term.**
- Proposed approaches: **collocation techniques, Galerkin schemes, low-rank methods, quadrature rules**, etc.

Neural Field Equations

- One of the most efficient ways: **Fast Fourier Transforms (FFTs).**





orange line: trapezoidal rule

blue line: FFT

Motivation

Motivated by **the rise of scientific machine learning (SciML)**, we look for new approaches to solving NFEs.

Traditional ML

- Relies on **statistical patterns** in the data.
- Depends on **large amounts of labelled data**.
- **May lack interpretability**, considered as a "black box" in some cases.
- Applications: e.g. image recognition, natural language processing, recommendation engines.

Scientific ML

- Integrates **physical principles** into ML process.
- Can handle situations with **limited or no data**.
- **Interpretability**: incorporates prior knowledge about the system's behaviour.
- Applications: e.g. solving differential equations, inverse problems, simulating physical phenomena.

Traditional ML

Many frameworks and libraries.

PyTorch Lightning TensorFlow Keras Hugging Face fast.ai

Hugging Face | Search models, datasets, users...

Tasks Libraries Datasets Languages Licenses Other | Models 320,856

AWS Azure Google Cloud

Scientific ML

Custom implementations, some frameworks exist.

SIAM REVIEW
Vol. 63, No. 1, pp. 208–228

DeepXDE: A Deep Learning Library for Solving Differential Equations*

Lu Lu†
Xuhui Meng‡
Zhiping Mao§
George Em Karniadakis¶

ELSEVIER
Available online at www.sciencedirect.com
ScienceDirect
Comput. Methods Appl. Mech. Engrg. 373 (2021) 113552
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

SciANN: A Keras/TensorFlow wrapper for scientific computations and physics-informed deep learning using artificial neural networks

Ehsan Haghighat*, Ruben Juanes
Massachusetts Institute of Technology, Cambridge, MA, United States of America

Received 11 May 2020; received in revised form 16 October 2020; accepted 27 October 2020
Available online 26 November 2020

Physics informed neural networks

- Designed to solve problems involving **Partial Differential Equations (PDEs).**
- The idea: **add differential equations into the loss function** when training the network.
- Leverages **automatic differentiation** (known as algorithmic differentiation or AutoDiff).
- Different kinds of problems: integer-order PDEs, fractional PDEs, stochastic PDEs and integrodifferential equations (IDEs).

Physics informed neural networks

Introduced in 2019:

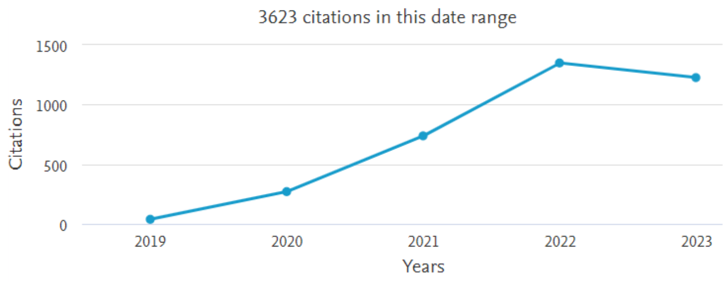
Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

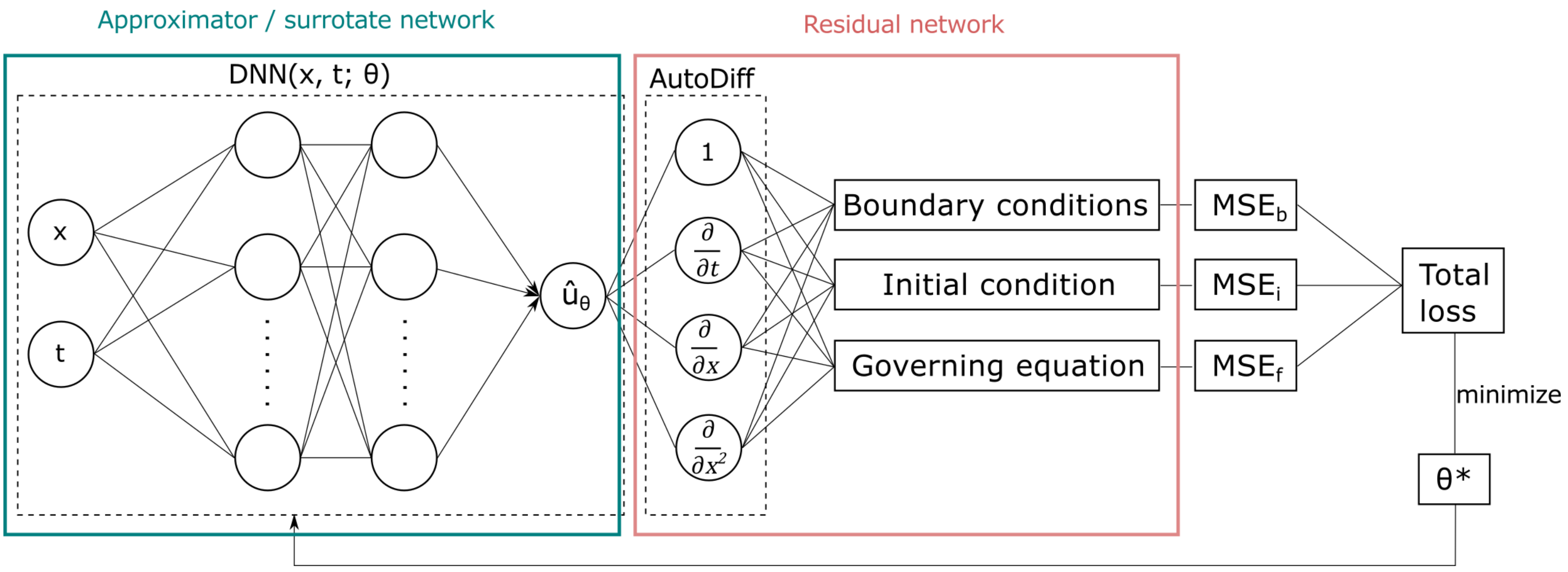
Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations

M. Raissi [a], P. Perdikaris [b,*], G.E. Karniadakis [a]



Physics informed neural networks

- The surrogate network is trained to provide an approximate solution at given collocation points.
- The residual network receives the output of the surrogate network and calculates a residual value (also called loss function).



Physics informed neural networks

PINNs can solve differential equations expressed in the form:

$$\mathcal{F}(u(z);\gamma) = f(z) \quad z \text{ in } \Omega,$$

$$\mathcal{B}(u(z)) = g(z) \quad z \text{ in } \partial\Omega,$$

defined on the domain $\Omega \subset \mathbb{R}$ with the boundary $\partial\Omega$, where:

- $z = [x_1, \ldots, x_{d-1}; t]$: the space-time coordinate vector,
- u: the unknown solution,
- γ: the parameters of the governing equation,
- f: the function identifying the data of the problem,
- $\mathcal{F}$: the non-linear differential operator.
- $\mathcal{B}$: arbitrary initial or boundary conditions,
- g: the boundary function.

Amari Equation

We solve the canonical Amari equation defined on a 1D finite domain Ω

$$\frac{\partial u(x,t)}{\partial t} = -u(x,t) + \int_\Omega w(|x-y|) f(u(y,t) - \kappa)\, dy, \quad (x,t) \text{ in } \Omega \times M, \tag{1}$$

$$u(x,0) = u_0, \quad x \text{ in } \Omega,$$

The initial condition u_0 is given by

$$u_0 = u(x,0) = A_0 e^{(-x^2/\sigma_0^2)} \tag{2}$$

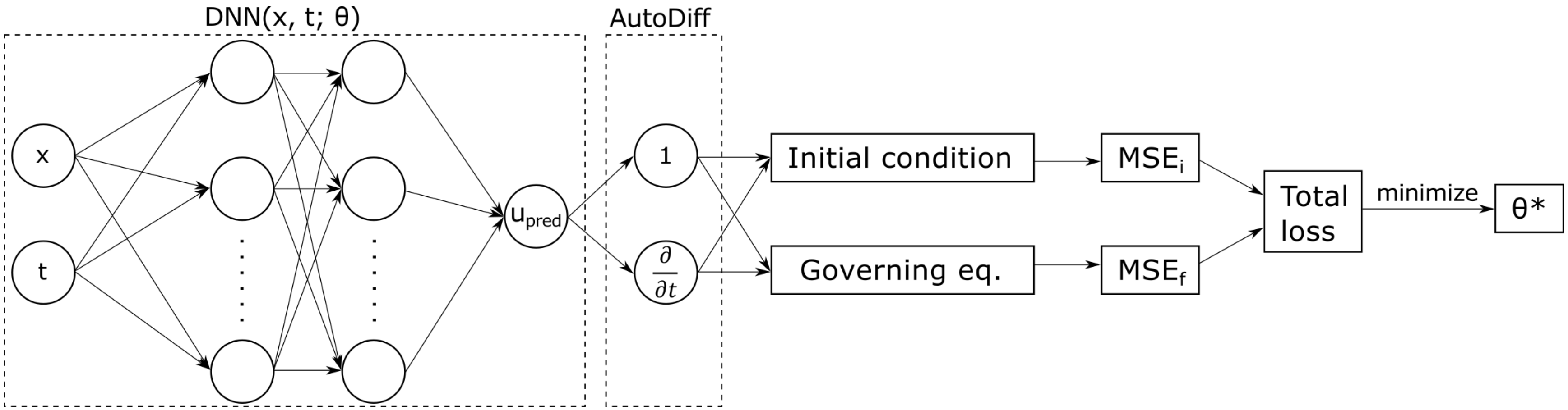
The oscillatory connectivity function w is given by

$$w_{osc}(x) = A_{osc} e^{-b|cx|}(b \sin|cx| + \cos(cx)). \tag{3}$$

PINNs and IDEs

- For the state-of-the-art PINN methods, **integral discretization** is a key prerequisite in order that IDEs can be transformed into ordinary differential equations (ODEs).
- Integral discretization inevitably introduces **discretization and truncation errors**.
- Possible solution: PINNs with auxiliary outputs (A-PINNs) which approximate the integrals in the governing equation.
- The integral term in the Amari equation is a spatial convolution and as such can be efficiently computed using **FFTs**.

PINN for Amari Equation



- Implemented in PyTorch.
- A fully connected feed-forward NN with two hidden layers and 40 neurons in each layer.
- tanh as the activation function because it satisfies the smoothness requirements for PINNs.
- L-BFGS optimizer with 0.01 learning rate.

The training data set

The inputs (x, t) to the neural network are the coordinates of the training points:

- 500 initial points $(x_i^{ini}, 0)$ uniformly sampled at $t = 0$.
- 10000 collocation points (x_t^f, t_t^f), uniformly sampled in the equation domain.

The loss function

- Learning takes place by **minimizing a loss function** which depends on the governing equation and the initial conditions:

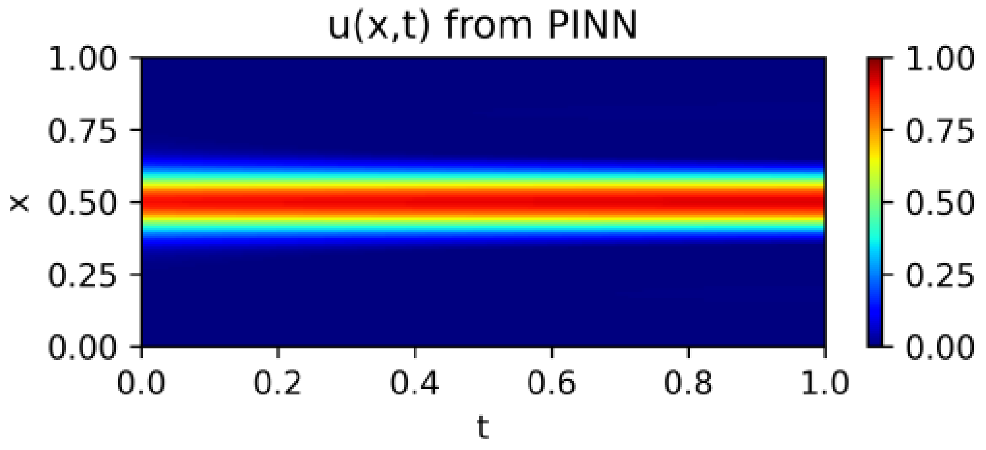
$$MSE_{total} = MSE_i + MSE_f$$

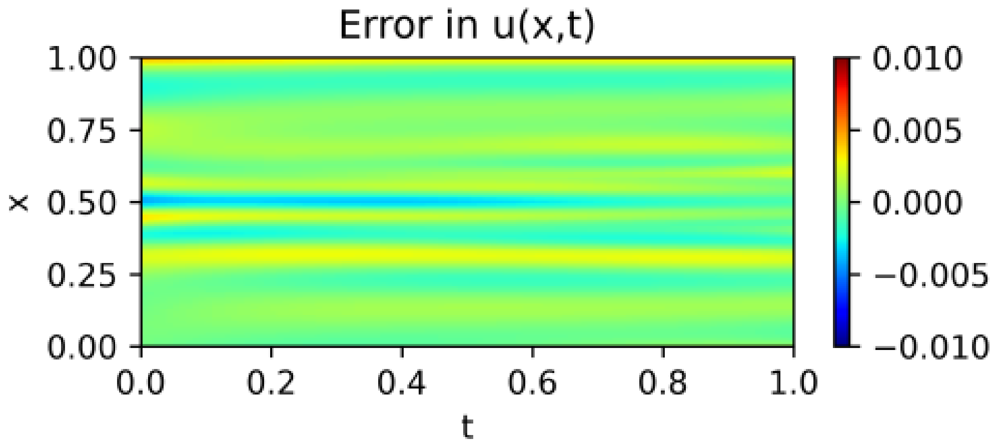
$$MSE_i = \frac{1}{N_i}\sum_{i=1}^{N_i} |u_{pred}(x_i^{ini}, 0; \theta) - u_0(x_i^{ini}, 0; \theta)|^2$$

$$MSE_f = \frac{1}{N_f}\sum_{i=1}^{N_f} |\frac{\partial u_{pred}(x_i^f, t_i^f; \theta)}{\partial t} + u_{pred}(x_i^f, t_i^f; \theta) - w \otimes H(u_{pred} - \kappa)(x_i^f, t_i^f; \theta)|^2$$

- Use of **FFTs** provides the **periodic boundary conditions** in x, so no the need to include them in the loss function.

Results

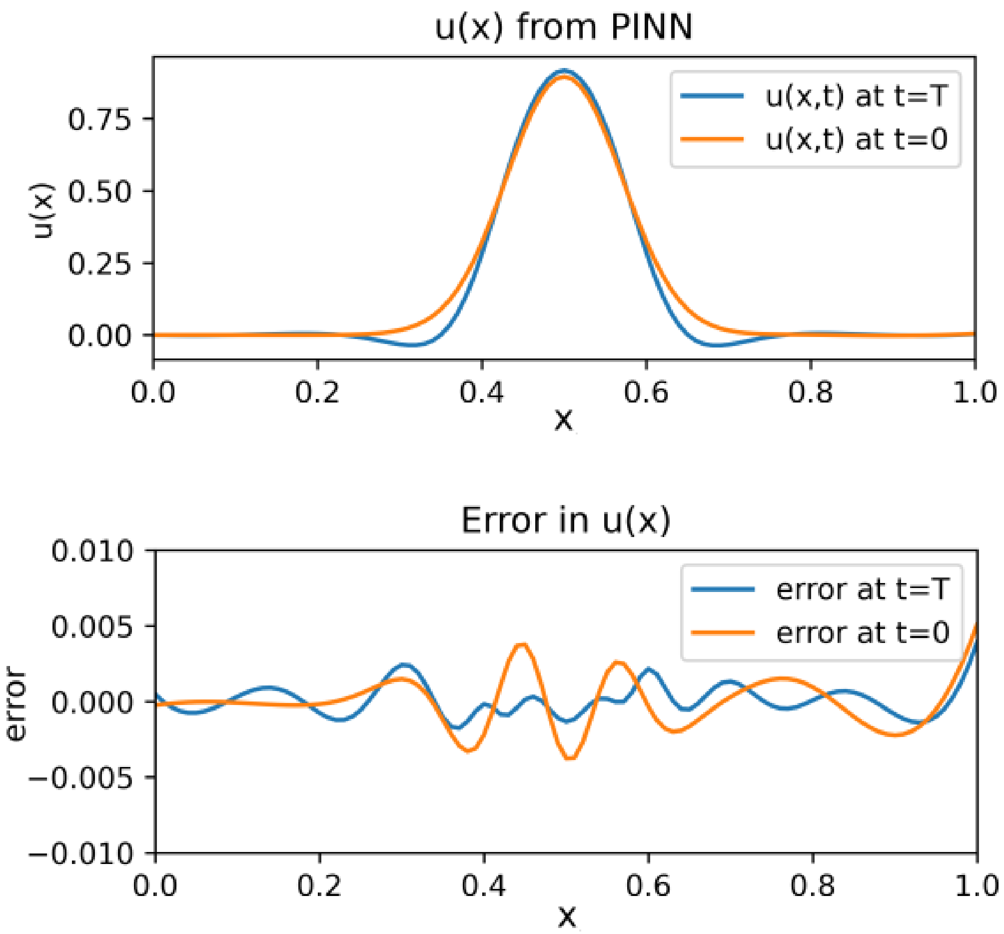




Left: Bump solution after 500 iterations.

The approximation error is the difference between the PINN solution and the one obtained by the forward Euler method.

Results



The relative L2 error between the PINN solution and the approximation obtained by the forward Euler method is 0.37%.

Future work

- Mitigate the restriction imposed by the tanh function (e.g., add a normalization layer).
- Adjust hyperparameters (e.g., number of layers, neurons, learning rate) for better accuracy.
- **NFEs with inputs (!)**, leveraging PINNs' transfer learning.
- **Inverse problem**: inferring NFE parameters from potentially noisy data.
- **NFEs in two or more spatial dimensions** or on surfaces with complex geometries.

References

- Amari, S. I. (1977). **Dynamics of pattern formation in lateral-inhibition type neural fields**. Biological Cybernetics, 27(2), 77-87.
- Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019). **Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations**. Journal of Computational Physics, 378, 686-707.
- Cuomo, S., Di Cola, V. S., Giampaolo, F., Rozza, G., Raissi, M., & Piccialli, F. (2022). **Scientific machine learning through physics–informed neural networks: Where we are and what's next.** Journal of Scientific Computing, 92(3), 88.
- Markidis, S. (2021). **The old and the new: Can physics-informed deep-learning replace traditional linear solvers?** Frontiers in Big Data, 4, 669097.