Parameter estimation of partial differential equations via neural networks

Alexander Glushko, Dmitry I. Kabanov

Final project on the Stochastic Numerics course

Once upon a time in 2019

Outline

- Introduction to the problem of parameter estimation of PDEs
- Solution set up
- Description of neural networks

Introduction to the problem

The main goal fo our project is:

Given data $\boldsymbol{D} = \{t_i, x_i, u_i\},$‘ $i = 1, ..., N$, observed from the solution of PDE of the form

$$u_t + \mathcal{N}(u; \boldsymbol{\lambda}) = 0, \tag{1}$$

estimate $\boldsymbol{\lambda}$.

Here $u = u(x, t)$ is the solution of the equation, $\mathcal{N}(u; \boldsymbol{\lambda})$ a nonlinear algebraic-differential operator, $\boldsymbol{\lambda}$ the vector of unknown parameters.

By Bayes' rule, the optimal value of $\boldsymbol{\lambda}$ is found through maximization of the posterior distribution [6]

$$\rho(\boldsymbol{\lambda}|\boldsymbol{D}) \propto \rho(\boldsymbol{D}|\boldsymbol{\lambda}) \times \rho(\boldsymbol{\lambda}). \tag{2}$$

Furthermore, we assume

$$u_i = u(x_i, t_i; \boldsymbol{\lambda}) + \epsilon_i, \quad i = 1, \ldots, N, \tag{3}$$

where $\epsilon_i \sim N(0, \sigma^2)$, and assign uninformative prior for $\boldsymbol{\lambda}$

$$\rho(\boldsymbol{\lambda}) = \text{const} \quad \text{for all } \boldsymbol{\lambda}, \tag{4}$$

so that, the problem of finding $\boldsymbol{\lambda}$ is a nonlinear unconstrained optimization problem

$$\arg\min_{\boldsymbol{\lambda}} \quad \log \sum_{i=1}^{N} \left[u_i - u(x_i, t_i; \boldsymbol{\lambda}) \right]^2, \tag{5}$$

here noise variance σ^2 is a nuisance parameter [6, section 8.2]

Analytical solution of the optimization problem (5) can be expensive, so following [5] we replace $u(x_i, t_i; \boldsymbol{\lambda})$ with a feedforward neural network [3]. The description of the neural network will be presented in the slides below.

Furthermore, to ensure that $u_{\mathrm{NN}}(x, t; \boldsymbol{\theta})$ is close to the exact solution of Eq. (1), we formulate the problem of estimating of the unknown parameters $\boldsymbol{\lambda}$:

$$\underset{\boldsymbol{\lambda}, \boldsymbol{\theta}}{\arg\min} \quad \sum_{i=1}^{N} \left[u_i - u_{\mathrm{NN}}(x_i, t_i; \boldsymbol{\theta}) \right]^2 \tag{6a}$$

$$\text{subject to} \quad u_{\mathrm{NN},t} + \mathcal{N}(u_{\mathrm{NN}}; \boldsymbol{\lambda}) = 0. \tag{6b}$$

The equality constraint is difficult to satisfy exactly as u_{NN} is an approximation of the exact solution of Eq. (1). Therefore, we relax the optimization problem introducing a secondary NN

$$f_{\text{NN}}(x, t; \boldsymbol{\lambda}, \boldsymbol{\theta}) = u_{\text{NN},t} + \mathcal{N}(u_{\text{NN}}; \boldsymbol{\lambda}), \tag{7}$$

where we plug the neural network u_{NN} into Eq. (1) and require that $f_{\text{NN}} \approx 0$.

Then both networks are trained simultaneously:

$$\arg\min_{\boldsymbol{\lambda}, \boldsymbol{\theta}} \sum_{i=1}^{N} \left[u_i - u_{\text{NN}}(x_i, t_i; \boldsymbol{\theta}) \right]^2 + \gamma \sum_{i=1}^{N} \left[f_{\text{NN}}(x_i, t_i; \boldsymbol{\lambda}, \boldsymbol{\theta}) \right]^2, \tag{8}$$

where γ is an extra hyperparameter that controls the importance of the penalty term. This parameter γ is chosen via cross-validation and bayesian regularization.

Description of neural networks

Let us recall the neural network equation:

$$u_{\mathrm{NN}}(x,t;\boldsymbol{\theta}) = g_L \circ g_{L-1} \circ \cdots \circ g_1,$$

where

$$g_\ell(z;\boldsymbol{\theta}_\ell) = \sigma(W_\ell z + b_\ell), \quad \ell = 1,\ldots,L.$$

Here

- L - is the number of layers in the network,
- 0 layer and L layer are input and output layers, respectively,
- layers from 1 to $L-1$ - are hidden layers,
- $\sigma = \tanh(z)$ - is a nonlinear activation function applied componentwise.

The neural-network parameter $\boldsymbol{\theta}$ contains the components of matrices $W_\ell \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$ and bias vectors $b_\ell \in \mathbb{R}^{n_\ell}$, where n_ℓ (number of "neurons") denotes the width of the ℓ^{th} layer.

Below, one can see the scheme of the neural network that we use in this project:

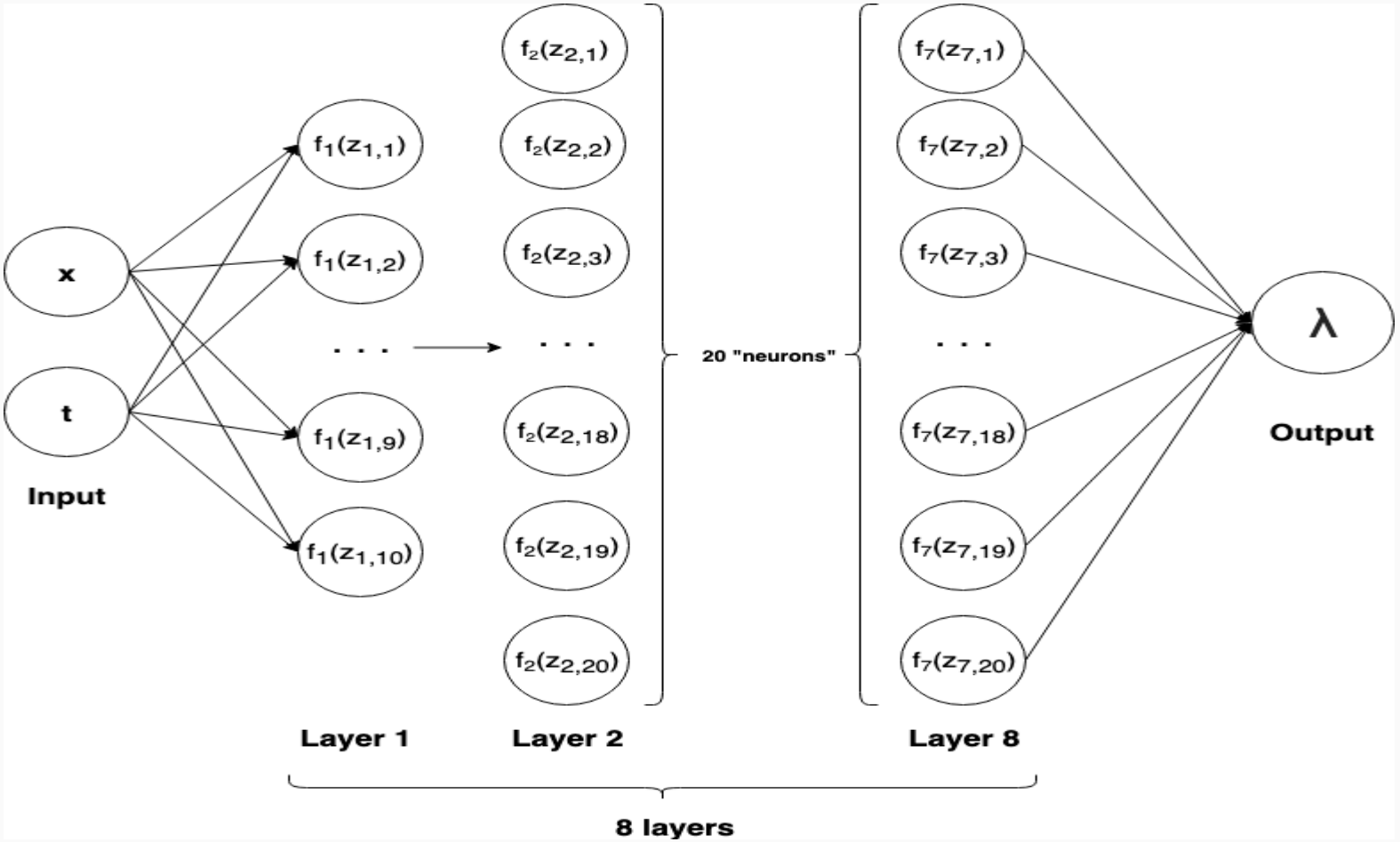


Figure 1: Figure 1. Physics informed neural network.

As one can see on the Figure 1., we use the first layer as input data driven form the equation (1). As an activation function, we use $\sigma = tanh(z)$. We use $tahn(z)$ as its derivative has a good behavior in the region $[-1,1]$ [4]. In total, the neural network has 10 layers. 6 of them have a width of 20 "neurons". Below, we will show the dependence of the resulting value λ on the neural network configuration. Especially, the affection of the number of hidden layers in the network.

Below, one can see the table of the output $\boldsymbol{\lambda}$ values depending on the network configuration for Burgers' equation. Here we have the relative error: $\frac{|\lambda_i - \lambda_{i\ True}|}{\lambda_{i\ True}},\ i = 1, 2$

Configuration	λ_1	Error λ_1	λ_2	Error λ_2
[2, 10, 1]	0.7347	26.527	0.00537	68.703
[2, 20, 1]	0.7398	26.0326	0.004882	53.3654
[2, 10, 20, 1]	0.951	4.89992	0.0044454	39.65582
[2, 10, 20x2, 1]	0.9975185	0.248152	0.003245	1.9329
[2, 10, 20x3, 1]	0.99794	0.20582	0.0032	0.614
[2, 10, 20x4, 1]	1.00022	0.02117	0.00321	0.2508
[2, 10, 20x5, 1]	0.99963	0.0374	0.003221	1.1881
[2, 10, 20x6, 1]	0.99985	0.01495	0.0031793	0.120

Table 1: Table 1. Dependence of the $\boldsymbol{\lambda}$ on the number of network layers.

As one can see in Table 1., the network output directly depends on the network length.

Example 1. Heat equation

Heat equation

We consider linear heat equation with homogeneous Dirichlet boundary conditions

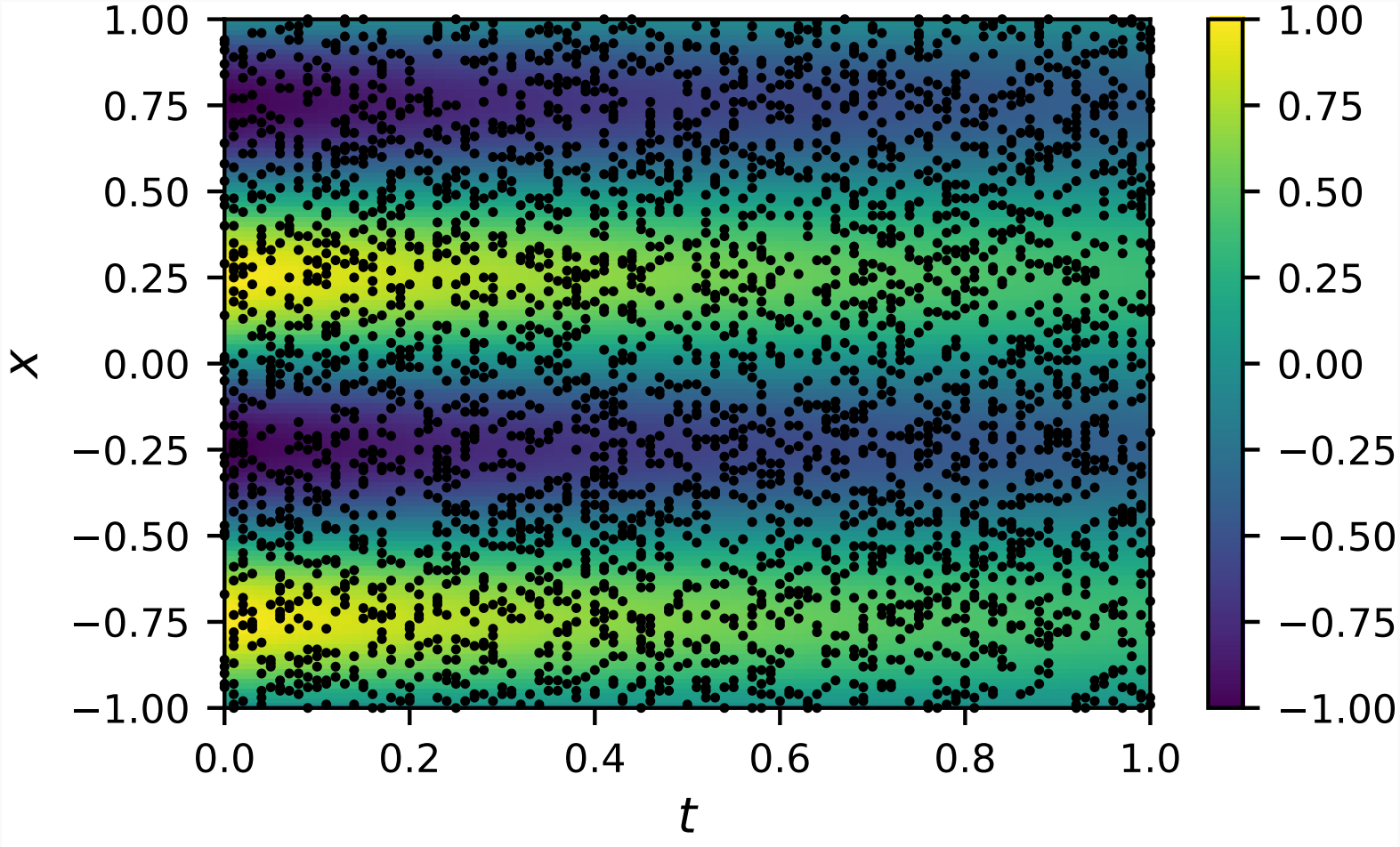
$$
\begin{aligned}
&u_t - \lambda u_x x - g(x,t) = 0, \quad x \in [-1;1], \quad t \in [0,1] \\
&u(x,0) = 0 \\
&u(0,x) = u(1,x) = 0
\end{aligned}
$$

with source term $g(x,t) = (4\pi^2 - 1)\sin(2\pi x)\exp(-t)$.

True value of λ is 1.

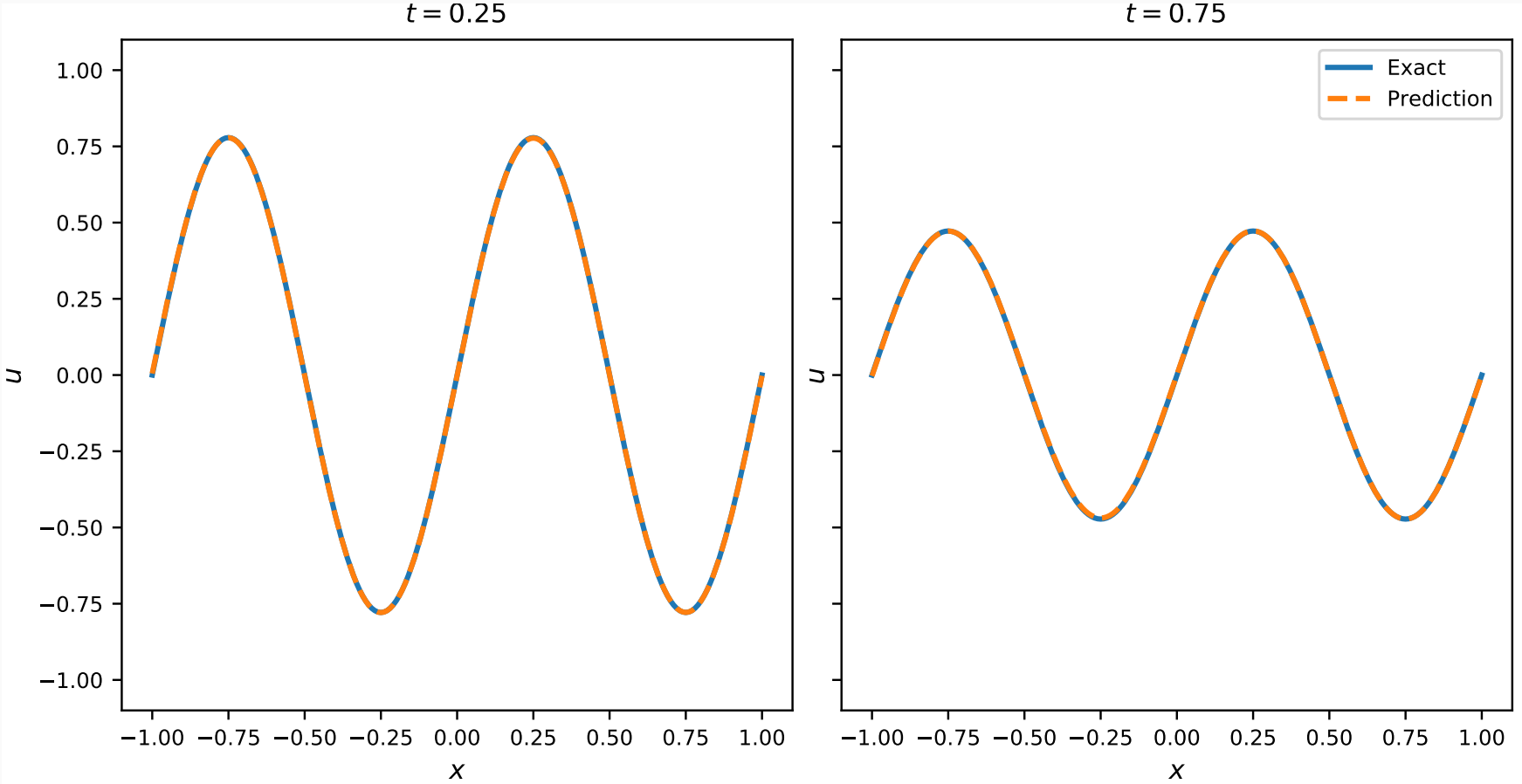
Observations

The exact solution $u = \sin(2\pi x)\exp(-t)$ is given on the uniform grid with 201 points in x and 101 points in t. We sample 3200 observations uniformly.



Preliminary results with $\gamma = 2$

Training the network with $\gamma = 2$ gives prediction $\widehat{\lambda} = 1.00175$ with error $\approx 0.18\%$.



Burgers' equation

Let us consider the Burgers' equation. This equation is nonlinear and serves as a prototype of the governing equations of fluid dynamics [1].

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu^2 \frac{\partial^2 u}{\partial x^2} \tag{9}$$

For small values of the viscosity parameter ν , Burgers' equation can lead to shock formation that is notoriously hard to resolve by classical numerical methods.

In this project we are going to consider the input data of the following form of the Burgers' equation:

$$u_t + uu_x - (0.01/\pi)u_{xx} = 0, x \in [-1, 1], t \in [0, T], \tag{10}$$

$$u(0, x) = -sin(\pi x), \tag{11}$$

$$u(t, -1) = u(t, 1) = 0. \tag{12}$$

As we are solving the inverse problem, our goal is to find values:

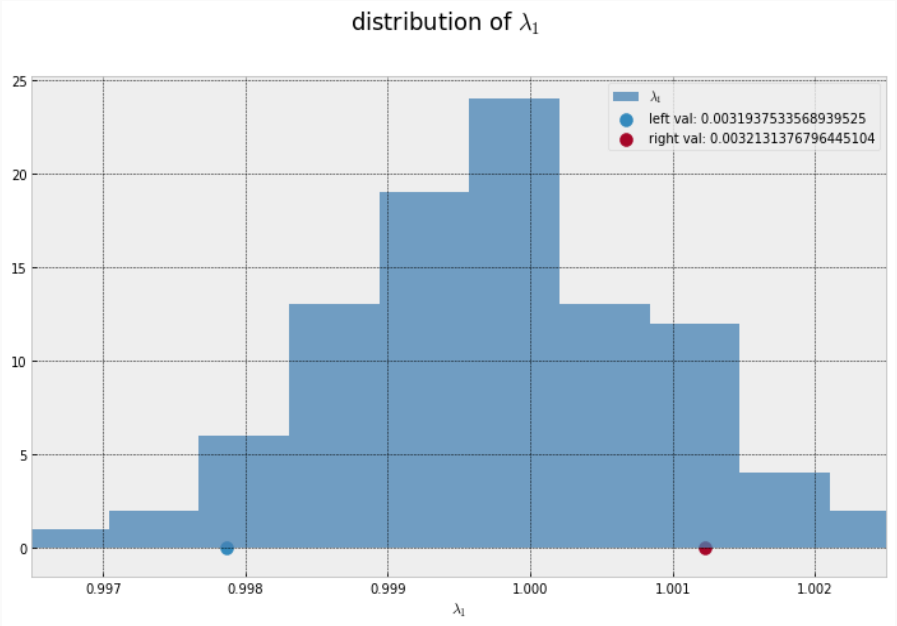
$$\lambda_1 = 1, \lambda_2 = (0.01/\pi) \approx 0.318$$

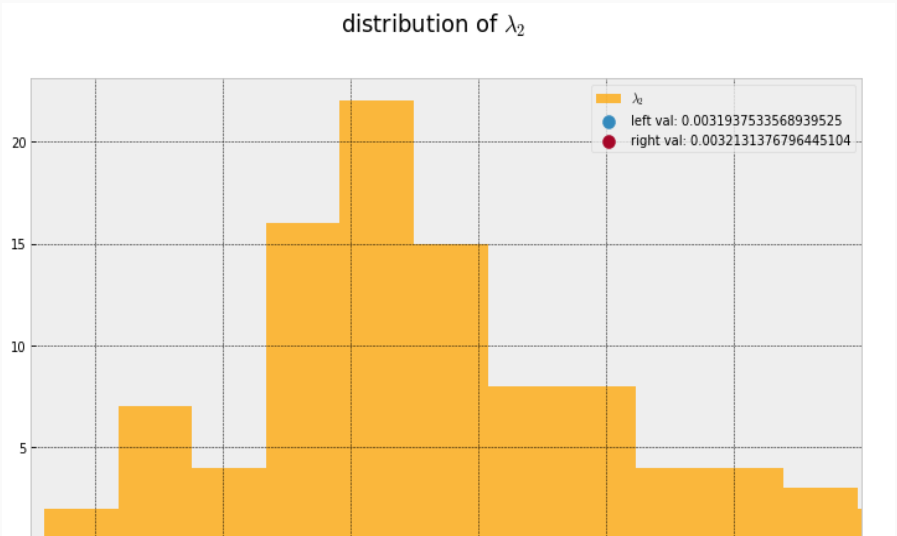
using the following input data matrices:

$$t \in [0, 1] \text{ of size } [100, 1], \tag{13}$$

$$x \in [-1, 1] \text{ of size } [256, 1], u \text{ of size } [256, 100] \tag{14}$$

Optimizing all loss functions using L-BFGS (a quasi-Newton, full-batch gradient-based optimization algorithm [2]), we have got the following bootstrapped results:





The panel of Figure 2 shows the predicted solution u(t,x) along with the locations of the initial and boundary training data. This prediction is obtained without any sort of discretization of the spatiotemporal domain.

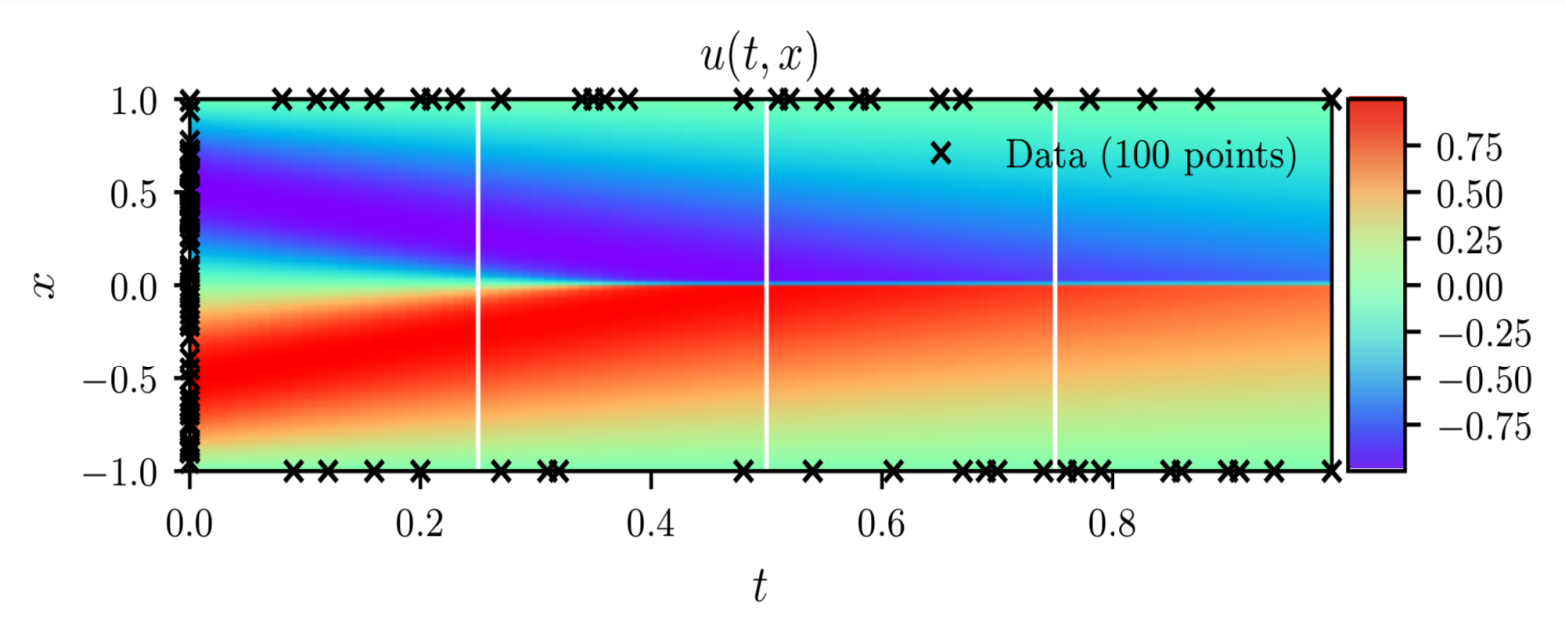


Figure 3: Figure 2. Predicted solution $u(t,x)$ along with the initial and boundary training data.

Below, we present a comparison between the exact and the predicted solutions at different time instants t = 0.25, 0.50, 0.75. The physics informed NN accurately captures the nonlinear behavior of the Burgers' equation that leads to the development of a sharp internal layer around t = 0.4.

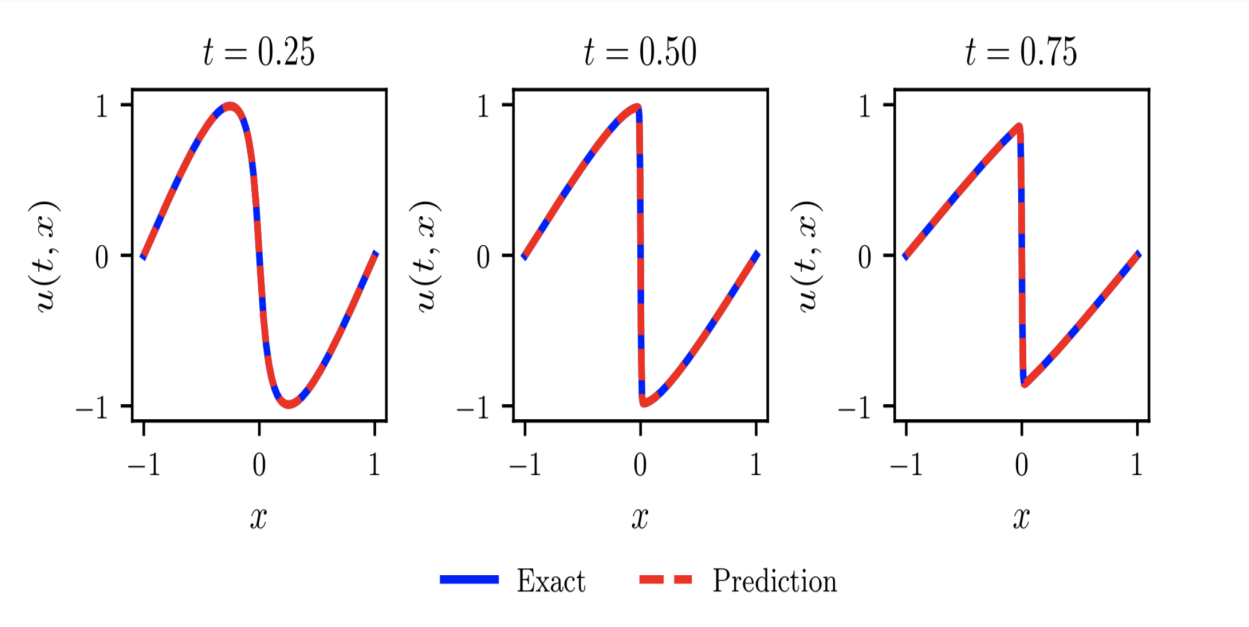


Figure 4: Figure 3. Comparison of the predicted and exact solutions corresponding to the three temporal snapshots depicted by the white vertical lines in the top panel.

Therefore, a key property of physics informed neural networks is that they can be effectively trained using small data sets; a setting often encountered in the study of physical systems for which the cost of data acquisition may be prohibitive.

Conclusions

M. D. C. Basdevant.
Spectral and finite difference solutions of the burgers equation.
Computers & fluids, 1986.

J. N. D. C. Liu.
On the limited memory bfgs method for large scale optimization.
Mathematical programming 45, 1989.

I. Goodfellow, Y. Bengio, and A. Courville.
***Deep learning*.**
MIT press, 2016.

H. B. D. Martin T. Hagan.
***Neural Networks Design*.**

M. Raissi, P. Perdikaris, and G. E. Karniadakis.

Physics informed deep learning (part ii): Data-driven discovery of nonlinear partial differential equations.
arXiv preprint arXiv:1711.10566, 2017.

D. Sivia and J. Skilling.
***Data analysis: a Bayesian tutorial*.**
OUP Oxford, 2006.