Accredited by National Journal Accreditation (ARJUNA) Managed by
Ministry of Education, Culture, Research and Technology with Second Grade (Sinta 2)
since year 2021 to 2026 according to the decree No. 158/E/KPT/2021

Published online on the journal's webpage: **http://jurnal.iaii.or.id**

JURNAL RESTI
(Rekayasa Sistem dan Teknologi Informasi)
Vol. 5 No. 6 (2021) 1106 - 1112 ISSN Media Elektronik: 2580-0760

On the Neural Network Solution of One-Dimensional Wave Problem

Aditya Firman Ihsan[1]
[1]Informatics, School of Computing, Telkom University
[1]adityaihsan@telkomuniversity.ac.id

Abstract

Artificial neural network has become an emerging popular method to handle various problems, especially in case where it has deep multiple neural layers. In this study, we use a deep artificial neural network model to solve one-dimensional wave equation, without any external datasets. Different type of boundary conditions, i.e., Dirichlet, Neumann, and Robin, are used. We analyze the model learning capabilities in a set of settings, such as data setup and the model width and depth. We also present some discussions of advantages and disadvantages of the model in comparison with other matured existing techniques to solve wave equation.

Keywords: neural network, wave problem, one-dimensional wave.

1. Introduction

Artificial neural network (ANN) has revolutionized many computational tasks, with broad applications and implementations. It is well known in its capability in handling data-based learning tasks, such as classification, object detection, forecasting, captioning, and many others [1]. These tasks are handled with some probability aspects in its result, which makes ANN is not popular yet in areas with demand of high accuracy such as scientific or mathematical problems. Recently, some studies began shifting the use of ANN to scientific problem, especially dynamical systems [2] and partial differential equations (PDE).

The application of ANN in science and engineering consists of two major tasks. The first is data-driven system identification, which usually use some forms of sparse regression [3]–[5]. The second is prediction of physical behavior by observed data or already identified governing system [5]. For the latter, there are at least three methods has been developed to solve PDE using neural networks [6], namely Physics-Informed Neural Network (PINN) [5], Feynman-Kac formula based method [7], and backward stochastic differential equation (BSDE) [8]. While last two of these methods are based on stochastic process and aimed specifically to solve PDE posed on high-dimensional domains, PINN has more flexibility and more applicable to wide variety of PDEs [6].

PINN overcomes the lack of information from data by utilizing the underlying physical laws of the system. Instead learning purely by supervision of data, PINN is supervised also by mathematical equations that governs the behavior of corresponding physical phenomena. It has been applied to various problems since proposed, hyperbolic or parabolic, including Navier-Stokes [9], 2D wave acoustic [10], seismic wave [11], cardiac activation mapping [12], and power systems [13].

Although PINN has been applied in complex system, an analysis of its performance in simpler system is needed. In this paper, we apply PINN in a simpler form of hyperbolic PDE, which is one-dimensional wave problem with finite domain. We analyze its performance and applicability to various model settings and condition.

Using a homogeneous Dirichlet-type boundary condition on one side, we apply three different basic types of boundary condition on the other boundary, i.e. Dirichlet, Neumann, and Robin. We use standard numerical solution by finite difference schema as comparison for the result obtained by PINN. This result obtained in this study can be a consideration for future usage of this method, especially the advantage and disadvantage of this method for a simple physics systems such as one-dimensional wave, as studied in this paper.

Accepted by Editor: 19-10-2021 | Final Revision: 30-11-2021 | Online Publication: 30-12-2021

2. Research Method

2.1. Problem Formulation

For current study, we consider following one dimensional wave equation with finite domain $[0, L]$.

$$u_{tt} = c u_{xx}, \quad 0 < x < L, \quad t > 0 \tag{1}$$

As it involves second derivatives of time, we need following two initial conditions, i.e. initial displacement and initial velocity of the wave.

$$u(x, 0) = f(x) \tag{2}$$

$$u_t(x, 0) = g(x) \tag{3}$$

We also need two boundary conditions, which we set homogeneous in this study, i.e.

$$u(x, L) = 0 \tag{4}$$

$$B[u](t) = 0 \tag{5}$$

Equation (5) is boundary condition at $x = 0$, where operator B depends on the type of the boundary condition. Here, we use three different types, i.e.

(a) Dirichlet: $B[u] = u(0, t)$ (6)

(b) Neumann: $B[u] = u_x(0, t)$ (7)

(c) Robin: $B[u] = u(0, t) - u_x(0, t)$ (8)

For the simulation, we choose $L = 1$ and $T = 2$ as our domain boundary. We set initial conditions that represents small displacement which released without initial velocity: $f(x) = 2\sin(\pi x)$ and $g(x) = 0$.

2.2. Neural Network

Artificial neural network (ANN) is basically an approximation of a function with predefined input and output. ANN is equipped with the so-called gradient descent technique, which is basically an optimization method with some objective function to be minimized [1]. Usually, the objective function is a loss function between the approximate output and the true must-be output. ANN approximate a function by a combination of linear units called neuron activated by a nonlinear function. ANN updates its state iteratively by computing the gradient of the loss function. Typically, because ANN requires some data to be optimize itself, it is part of machine learning methods, which learn from data to optimize its capability to a given task. If the set of neurons are stacked to form some layers, ANN is called Deep Neural Network (DNN), which deep represents the depth of the neural layers.

DNN can be used to approximate a function given some set of data. In this matter, we use DNN to approximate wave displacement $u(x, t)$ with input location x and time t as its independent variable. DNN refines the approximation by iteratively update its state using gradient descent algorithm based on predefined objective function or metric. One of standard used metrics is mean squared error between output prediction and true data.

Neural network we use is a stack of linear layer with hyperbolic tangent activation. Each layer has the same number of units, except the output layer with only 1 neuron. This last neuron is not equipped with any activation function. We vary the width (number of neurons each layer) and the depth (number of layers) of the DNN. The illustration of the network is shown in Figure 1.

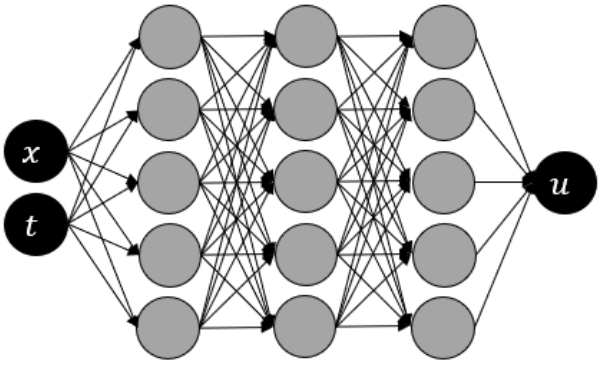


Figure 1. Neural network used for PINN. The input is randomly generated values of x and t. It can be seen as an approximation of function $u(x, t)$.

2.3. Physics-Informed Neural Network (PINN)

Physical problems rarely have large amount of clean data to be used. To overcome this issue, we can utilize other source of information as additional supervision for the DNN. In this case, PINN uses all known governing equations related to the function to be approximated as supervisor to the learning of the DNN.

PINN uses randomly generated data of independent variables (in this case x and t) to be inputted to DNN to Physical equations governing the problem are then used as information for DNN to update its state. In this case, the wave equations (1), initial conditions (2)-(3), and boundary conditions (4)-(8) are then used to form a loss function for the prediction, which then backpropagated to update the state of the DNN.

To reach our objective, for each equation we need some samples of x and t to be inputted to the model. In that case, we generate three different sets of data by random with values within domain of simulation, i.e., $x \in [0, L]$ and $t \in [0, T]$. These input datasets are (x^p, t^p), x^n, and t^b, each with size N_p, N_n, and N_b, respectively.

Next, we formulate the objective of the DNN as follows.

$$\min(L_p + L_{init} + L_{bound}) \tag{9}$$

where L_p, L_{init}, and L_{bound} represents how close the output fit the governing equation (1) inside given domain, fit the initial conditions (2) and (3), and fit the

DOI: https://doi.org/10.29207/resti.v5i6.3565

boundary conditions (4) and (5) respectively. To be exact, the formula for each loss function is following.

$$L_p = \frac{1}{N_p}\sum_{i=1}^{N_p}\left(u_{tt}(x_i^p, t_i^p) - cu_{xx}(x_i^p, t_i^p)\right)^2, \quad (10)$$

$$L_{init} = \frac{1}{N_n}\sum_{i=1}^{N_n}\left[\left(u(x_i^n, 0) - f(x_i^n)\right)^2 + \left(u_t(x_i^n, 0) - g(x_i^n)\right)^2\right], \quad (11)$$

$$L_{bound} = \frac{1}{N_b}\sum_{i=1}^{N_b}\left[\left(u(0, t_i^b)\right)^2 + \left(B[u](t_i^b)\right)^2\right], \quad (12)$$

By differential programming, we can compute the values of all derivatives of u from the output of the DNN. Using gradient descent or its variation, derivatives of three loss functions above will be used to update the parameters of the DNN. This learning mechanism of PINN is illustrated at Figure 2. Each loss is computed in different ways as stated in equation (10)-(12). Note first that u_x in the computation of L_{bound} only used in Neumann and Robin type condition.

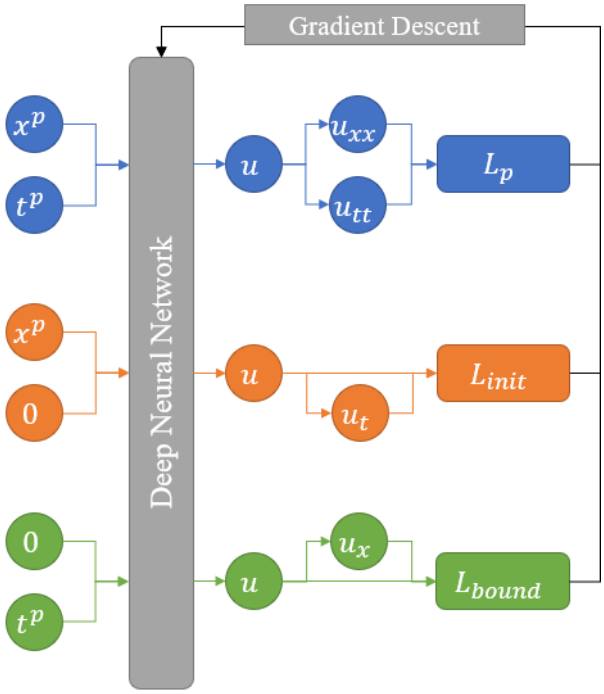


Figure 2. Illustration of how PINN we used in this study works.

3. Result and Discussion

We use different values of N_n and N_p where N_b is set to be $2N_n$ because of ratio of the domain. We train the model with maximum 100 epochs and terminate if the total loss value is not decreasing after some epochs. Adam (Adaptive Moment Estimation) optimization technique is used with 0.005 learning rate. This value of learning rate is actually a result of hyperparameter tuning done prior. Higher learning rate gives more unstable training process. As we will see later, the problem loss tends to give periodic spike.

The simulation result is then evaluated by computing the mean squared differences with the result from finite difference numerical result.

3.1. Variations of Architecture

In this initial simulation, we use $N_n = 10000$ and $N_p = 100000$ as the size of our input data. We also batched the boundary data with 200 as the batch size. We simulate all types of boundary condition while varying different depths and widths of the DNN which represents the variation of model architecture.

It will be a natural expectation that larger the model, better result we will get. It is also mentioned in the first paper of PINN, where adding more neurons or layers to the PINN decreases the relative error of the result [5]. However, as we run the simulation for some values of number of neurons and number of layers, surprisingly what we found is quite different. The result can be seen in Figure 3-4.

On one hand, how the number of neural units affects the performance is quite fit the expectation. Lower number of neurons gives worse result as less hidden features are captured by the model. However, the trend changes as 64 units of neuron gives worse result than model with 32 units. Interestingly, this pattern occurs in all types of boundary condition. Small anomaly happens in Robin-type where model with 4 neurons gives slightly better result than model with 8 neurons. Why 32 units model gives optimal result rather than model with lower or higher number of units is quite a mystery. As units represent additional hidden features extracted by the model, it is possible that too many features reduce model capability to fit all the governing equations simultaneously.

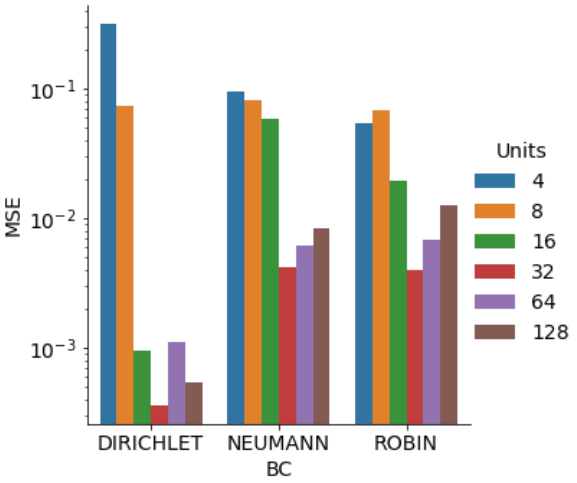


Figure 3. Mean squared error of the model with of layers is fixed to 2 while the number of neural units varies

DOI: https://doi.org/10.29207/resti.v5i6.3565

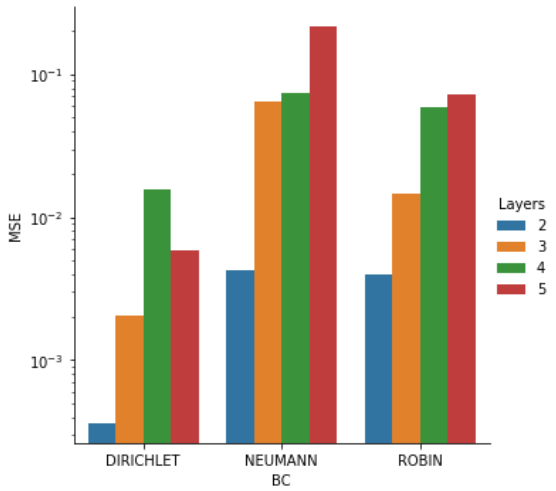


Figure 4. Mean squared error of the model with number of units per layer is fixed to 32 while the number of layers varies

What is quite out of expectation is how the performance behaves with different model depths. We can see at Figure 4, as the number of layers increases, the error also increases, except for the Dirichlet-type system where 5 layers model gives better result than 4 layers one. This means that high model complexity, which represented by the depth of the neural network, is not well-suited to the wave problem. Non-linearity the model needs to have to properly fit all the governing equations in this case is not much, where 2 stacks of hyperbolic tangent activated linear layer is enough to approximate the solution.

3.2. Variations of Data Size

In this section, we use a model with 2 layers of 32 neural units, as it gives the best result from previous section. We vary the number of input data, i.e., N_b and N_p. Despite the number of total input data, for each case, we set constant batch size for training. This is due to our findings that some values of batch size may cause instability to training process, which will be shown later. Other hyperparameters, such as learning rate, are tuned beforehand. Overall result can be seen in Table 1.

For the values of N_b, it is shown that increasing it will give better result. The values of N_b, together with N_n, are crucial for model to fit the boundary at the domain. On the other hand, N_p only affect how the model learn the solution behavior inside of the domain. The model is not fitting exact values inside the domain, it only fit the governing equation – how the values are changing inside the domain, which in wave problem represented by second derivatives. This is one of the possible reasons why N_p does not need to be too large. Because as long as the model learns enough the general dynamics inside the domain, it can generalize well to all other points. This conclusion is proven by the fact that the optimal value of N_p is not the largest one, as shown in Table 1.

Table 1. Mean squared errors of results from model trained with different values of input data

N_p	N_b		
	1000	**5000**	**10000**
10000	0.20554	0.00399	0.0008
50000	0.10472	0.00222	**0.0003**
100000	0.17894	0.00401	0.00036

Table 2. Mean squared errors of results from model trained with different number of batch size

B_p	B_b		
	50	**200**	**500**
1000	0.00012	0.00165	0.00169
10000	0.00025	0.00095	0.00152
100000	0.00012	0.00035	0.00099

It is shown in the table that the best evaluation result, i.e., less error, is obtained when the number of internal domain data N_p is 5000 not 100000. One important thing to be noted is that value of N_p are the number of data used to represent 2-dimensional internal domain, while N_b used to represent 1-dimensional domain boundary. Sampling distribution used may have effect. In this study, we use uniform distribution for each variable x and t.

3.3. Variations of Batch Size

Batch processing is one of the most useful techniques in neural network, as it optimizes the learning process. Computing completely a huge amount of data in a single iteration may cause memory overload, reducing efficiency of the process. Dividing the data in some set of mini batches leads to more efficient computation, even though the model will take more iterations to converge. To check how batching affect training process in PINN, we vary two different batch sizes. Input data for boundary is batched using batch size B_b, while input data for internal domain is batched using batch size B_p. The summary of the result can be seen in Table 2.

The two batch sizes give different effect on the model performance. Higher the value of B_b, higher the error, while the value of B_p gives opposite effect. Behavior of the model in variation of $B_b$is quite natural, because small batch size offers regularization effect. The difference of behavior is quite interesting. Highest number of B_p used in this case is actually the same as N_p used, which is 100000, which means in this case the input data for internal domain is not batched.

The strange behavior of B_p may be caused by high dimension internal data must represent, as discussed before. The model needs to learn the dynamics of the wave at different place and time simultaneously. This makes any partition of training data gives higher error as the model learn different dynamics separately. Thus,

DOI: https://doi.org/10.29207/resti.v5i6.3565

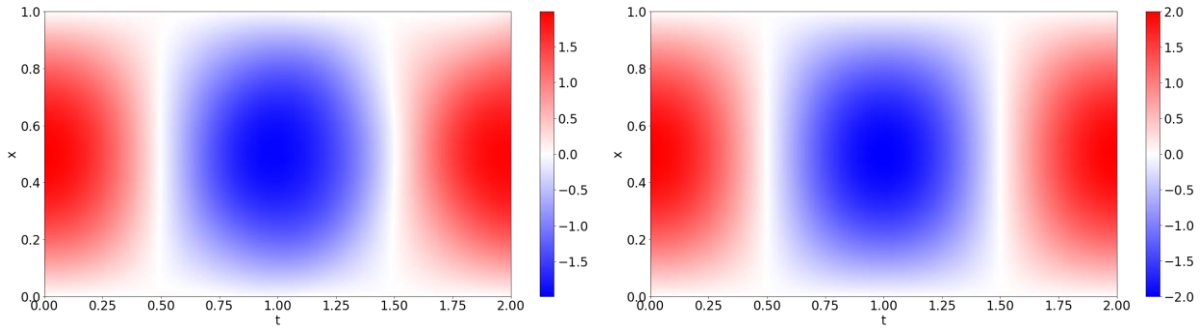
training process without batching gives the best performance.

3.3. Overall Analysis

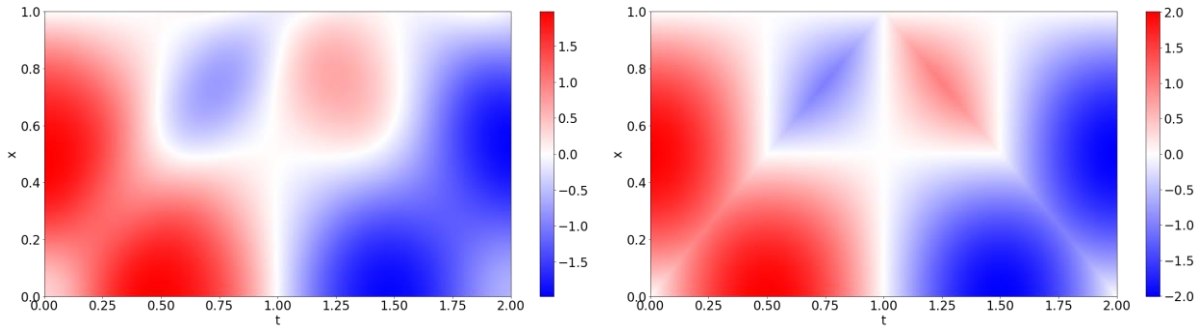
Previously, we have computed overall mean squared difference between the neural network solution and standard numerical solution. However, overall averaged difference may not represent how the solutions gives similar dynamics inside the domain. In that case, we plot the map of the solution within the domain of simulation. It can be seen in Figure 5.

We can see that in the map how the wave evolves over time with different boundary conditions. Dirichlet boundary represents a wave with attached ends, so it will form a perfect standing wave. Neumann boundary on the other hand gives one free end to the wave, makes additional dynamics at the boundary. The similar happens to Robin boundary, but the reflection is not perfect as the wave end is not entirely free, represents elastic attachment. This forms similar dynamics to the Neumann case but kind of distorted.

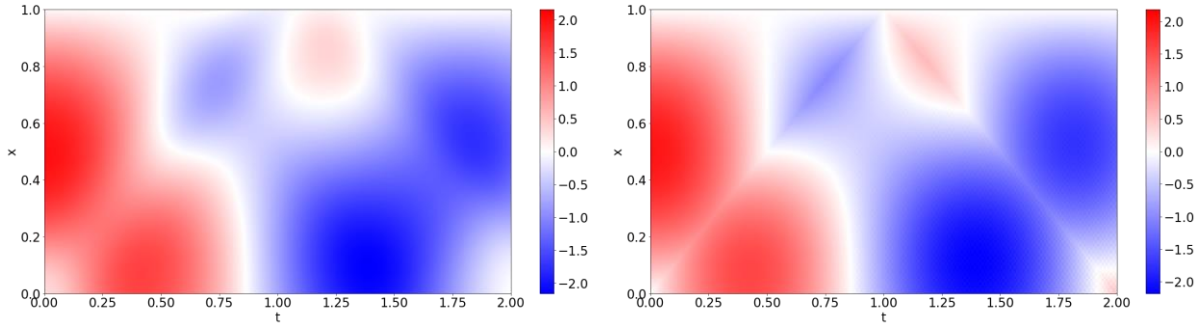
The neural network solutions in Figure 5 are the result of training with the best combination of parameters found in previous section. Briefly, we can see that solutions from both methods are quite similar, especially in case of Dirichlet-type boundary condition. Unfortunately, as we see closer at the results of Neumann-type and Robin-type, the neural network solutions tend to give smoother wave solution. Sharp wave fronts resulted from reflection at the boundary shown in numerical solution hardly appears in the neural network solution.



(a)



(b)



(c)

Figure 5. Map of the result in spatial and temporal domain of the problem with different boundary condition, i.e., (a) Dirichlet-type; (b) Neumann-type, (c) Robin-type

DOI: https://doi.org/10.29207/resti.v5i6.3565

As we have seen that the model architecture used in this section is already optimal, the source of this issue may lie at the amount of data used to train the model, especially the data generated at the boundary. The values of N_b and N_n are still can be increased further to increase the performance of the model. Unfortunately, at this point, the training time has already very long, especially compared to standard numerical computation. One of the major factors of the training time is the amount of data. Thus, increasing it further may cause unacceptable training time without any guarantee that the result may be better.

This issue has shown one of possible current disadvantages of neural network approach to solve wave problem, in addition to the very long computational time. However, many improvements to the methods may still open, such as the use of Lagrangian descent algorithm [14], Xavier initialization of the network parameters [14], BFGS optimization [15], and many others. We have to remember that PINN, as well as other neural network-based approach in solving differential equations, is still an emerging topic. At least we can see that because PINN is somehow like an approximation function, we can compute the solution for any value of input variables as long as they lie within simulation domain.

If we plot the graph of each loss over epochs, as shown in Figure 6-8, we can see that actually both initial conditions and boundary conditions fitted pretty quickly in few epochs with monotonic trends. However, the spikes in problem loss L_p indicates the difficulties faced by the model in fitting the governing equation inside the domain.

As stated before, sampling distribution of the data may have influence, because different than boundary or initial conditions, the model need to capture 2-dimensional information from the governing equation. The spikes appear in all types of boundary condition but more frequently in Robin type. It may be caused by the complexity of interaction of the wave from inside the domain with the boundary in Robin condition. Also, the spikes show that the model was not really learning inside the boundary except few first epochs. Further investigation and study may be needed to understand more this phenomenon.

4. Conclusion

A neural network has been implemented to the case of 1-D wave problem. The architecture used are Physics Inform Neural Network (PINN), which learns to approximate the solution using governing equation and initial/boundary conditions. We have studied how the size of the model architecture, namely depth and width, affects the performance significantly. Shallower yet wider networks are shown to give better result, as deeper network gives too much nonlinearity to the model.

We have seen also how amount of generated data influence the model performance. More data may give better result but will consume more computational time and memory. This becomes one of the disadvantages of PINN compared to existing numerical solver of PDEs. However, PINN have large potentials to become powerful method. Further improvements are still open.

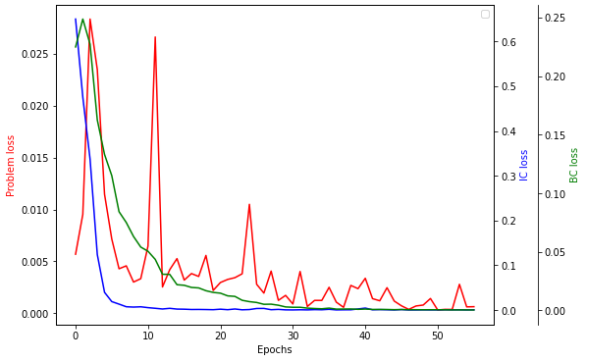


Figure 6. Loss curve of PINN training for Dirichlet problem

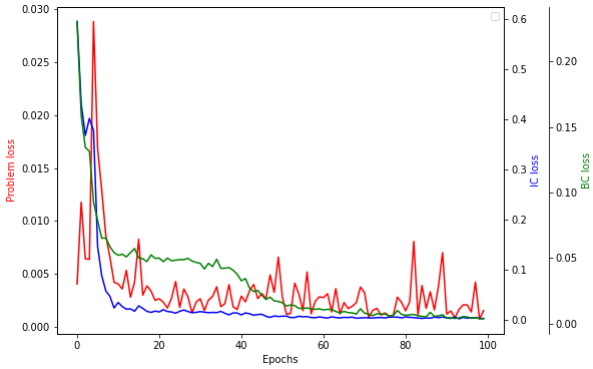


Figure 7. Loss curve of PINN training for Neumann problem

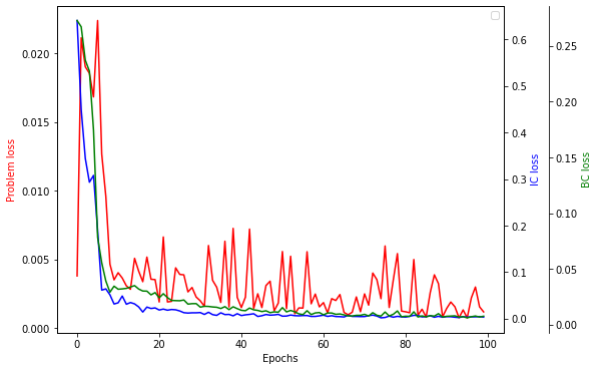


Figure 8. Loss curve of PINN training for Robin problem

References

[1] Y. Bengio, A. Courville, and I. J. Goodfellow, *Deep Learning*. 2016.

[2] J. Horrocks and C. T. Bauch, "Algorithmic discovery of dynamic models from infectious disease data," *Sci. Rep.*, vol. 10, no. 1, pp. 1–19, 2020, doi: 10.1038/s41598-020-63877-w.

DOI: https://doi.org/10.29207/resti.v5i6.3565

[3] N. M. Mangan, T. Askham, S. L. Brunton, J. N. Kutz, and J. L. Proctor, "Model selection for hybrid dynamical systems via sparse regression," *Proc. R. Soc. A Math. Phys. Eng. Sci.*, vol. 475, no. 2223, 2019, doi: 10.1098/rspa.2018.0534.

[4] N. M. Mangan, J. N. Kutz, S. L. Brunton, and J. L. Proctor, "Model selection for dynamical systems via sparse regression and information criteria," *Proc. R. Soc. A Math. Phys. Eng. Sci.*, vol. 473, no. 2204, pp. 1–14, 2017, doi: 10.1098/rspa.2017.0009.

[5] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *J. Comput. Phys.*, vol. 378, pp. 686–707, 2019, doi: 10.1016/j.jcp.2018.10.045.

[6] J. Blechschmidt and O. G. Ernst, "Three ways to solve partial differential equations with neural networks — A review," *GAMM Mitteilungen*, vol. 44, no. 2, pp. 1–29, 2021, doi: 10.1002/gamm.202100006.

[7] C. Beck, S. Becker, P. Grohs, N. Jaafari, and A. Jentzen, "Solving the Kolmogorov PDE by Means of Deep Learning," *J. Sci. Comput.*, vol. 88, no. 3, pp. 1–56, 2021, doi: 10.1007/s10915-021-01590-0.

[8] J. Han, A. Jentzen, and E. Weinan, "Solving high-dimensional partial differential equations using deep learning," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 115, no. 34, pp. 8505–8510, 2018, doi: 10.1073/pnas.1718942115.

[9] H. Eivazi, M. Tahani, P. Schlatter, and R. Vinuesa, "Physics-informed neural networks for solving Reynolds-averaged Navier-Stokes equations," 2021, [Online]. Available: http://arxiv.org/abs/2107.10711.

[10] B. Moseley, A. Markham, and T. Nissen-Meyer, "Solving the wave equation with physics-informed deep learning," 2020, [Online]. Available: http://arxiv.org/abs/2006.11894.

[11] S. Karimpouli and P. Tahmasebi, "Physics informed machine learning: Seismic wave equation," *Geosci. Front.*, vol. 11, no. 6, pp. 1993–2001, 2020, doi: 10.1016/j.gsf.2020.07.007.

[12] F. Sahli Costabal, Y. Yang, P. Perdikaris, D. E. Hurtado, and E. Kuhl, "Physics-Informed Neural Networks for Cardiac Activation Mapping," *Front. Phys.*, vol. 8, no. February, pp. 1–12, 2020, doi: 10.3389/fphy.2020.00042.

[13] G. S. Misyris, A. Venzke, and S. Chatzivasileiadis, "Physics-informed neural networks for power systems," *IEEE Power Energy Soc. Gen. Meet.*, vol. 2020-Augus, 2020, doi: 10.1109/PESGM41954.2020.9282004.

[14] T. Dockhorn, "A Discussion on Solving Partial Differential Equations using Neural Networks."

[15] S. Markidis, "The Old and the New: Can Physics-Informed Deep-Learning Replace Traditional Linear Solvers?," vol. 1, no. 1, pp. 1–20, 2021, [Online]. Available: http://arxiv.org/abs/2103.09655.

DOI: https://doi.org/10.29207/resti.v5i6.3565