PINN WITH ANTISYMMETRIC RNN FOR SOLVING NONLINEAR PARTIAL DIFFERENTIAL EQUATIONS

Pavodi Maniamfu[*1], **U. A. Md. Ehsan Ali**[1], **and Keisuke Kameyama**[2]
[1]Degree programs in Systems and Information Engineering, Graduate School of Science and Technology , University of Tsukuba, Tsukuba, 305-8577, Japan
[2]Institute of Systems and Information Engineering, University of Tsukuba, Tsukuba, 305-8577, Japan

ABSTRACT

Physics-informed neural network (PINN) is a new paradigm for solving partial differential equations (PDEs). It employs multilayer perceptrons (MLPs) with numerical methods, e.g. Runge-Kutta (RK) to model discrete-time PDEs. However, the present networks' structure exhibits constraints inherent in conventional numerical discretization approaches, particularly in handling iterative time-stepping processes. In this work, we propose physics-informed antisymmetric recurrent neural network (PIARNN), which encodes RK intermediate stages in the hidden state of the recurrent neural networks, thus enhancing the capabilities of the architecture of the PINN. We conducted the experiments on the Allen-Cahn equation, where the PIARNN achieved superior prediction performance across various RK stages compared to the standard PINN.

Keywords : PINN, Recurrent Neural Network, Partial Differential Equation, Runge-Kutta method.

1 Introduction

Multilayered neural networks approximate almost any function. They are a class of universal approximators[1]. This property together with their nonlinearity allows them to tackle complex problems beyond the linear mappings in fields such as computer vision [2] and natural language processing[3].

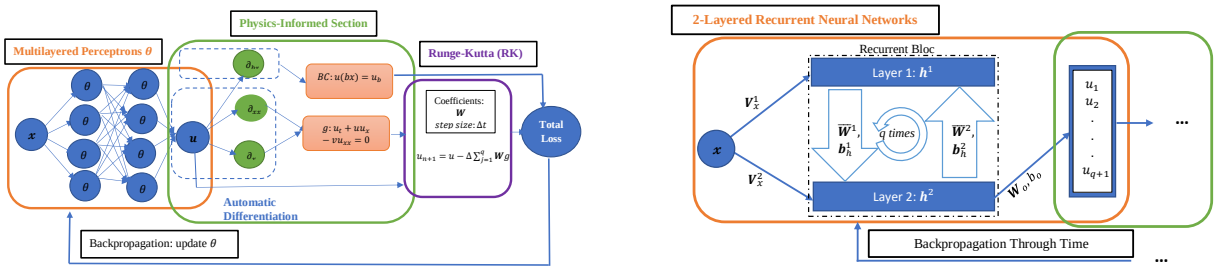
Recently, multilayer perceptrons (MLPs) are combined with numerical approaches to solve partial differential equations (PDEs)[4]. While neural network networks demanded large labeled data [2], which can be expensive to acquire in scientific settings. The existence of prior knowledge helps offset this data scarcity through regularization.

Physics-informed neural network (PINN) [4] is a novel approach, which utilizes a few samples of data to approximate the solution of the PDE via regularization that satisfies the PDE conditions. Despite its success in various applications such as gas dynamics, chemical kinetics, optimal control [5–7], it faces some challenges such as the the complexity of the loss landscape's geometry and an expensive training cost.

In this paper, we propose to utilize a robust and stable Recurrent Neural Network (RNN) architecture for the PINN framework. This architecture extends from the antisymmetric RNN proposed in [8]. The proposed approach employs a strictly upper triangular connection matrix to maintain the required antisymmetry. This makes the PIARNN more efficient in terms of the number of parameters. The main contributions of this work are as follows:

- We integrate RNN with the RK time-stepping methods in PINN's context. This approach allows to enhance the stability and expressivity of the PINN.
- The antisymmetric property allows to stabilize the learning process of the PINN framework and reducing the number of parameters while maintaining better performance.

[*]maniamfu@adapt.cs.tsukuba.ac.jp



(a) (b)

Figure 1: Comparative Architectures. (a) PINN. (b) RNN used in PIARNN.

2 Physics-Informed Neural Networks

Physics-informed neural networks (PINNs) are a three-stage framework for solving continuous- and discrete-time PDEs. They combine a layered neural network to learn the solution, a physics-informed section to compute PDE residuals via automatic differentiation [9], and a discrete-time model using numerical approaches, such as Runge-Kutta methods, for approximating the PDE solution. Fig. 1a depicts the global overview of the PINN.

Suppose the PDE that governs a spatio-temporal dynamical system of $u(x,t)$ is known as follows,

$$f(x,t) := u_t + N[u] = 0, \quad x \in \Omega, \quad t \in [0,T]. \tag{1}$$

Here, $u_t := \frac{\partial u}{\partial t}$ denotes the partial derivative of u w.r.t time t. Here, Function N is a nonlinear differential operator. Set Ω is a subset of $\mathbb{R}^D$, and represents the spatial domain. Interval $[0,T]$ is the time interval of the solution. The PINN solves the Eq. (1) by incorporating the PDE residuals and boundary conditions into the loss function as a soft constraint regularization.

The neural network in Fig. 1a takes a spatial scalar x as an input at a fixed time t, which is transformed through the network to output a $q+1$-dimensional feature vector. These outputs correspond to the Runge-Kutta stages $u^{n+c_i}(x)$, for $i = 1, ..., q$. The network's predictions are used to compute the derivatives of the nonlinear operator $N[.]$ w.r.t the input coordinate.

The solution is learned using the sum of squared errors, which is defined as follows:

$$SSE = SSE_n + SSE_b \tag{2}$$

where SSE_n denotes the residual error as:

$$SSE_n = \sum_{j=1}^{q+1} \sum_{i=1}^{N_n} |u_j^n(x^{n,i}) - u^{n,i}|^2, \tag{3}$$

and SSE_b enforces the boundary conditions of the PDE of interest.

In discrete-time modeling, the PINN is used with Runge-Kutta (RK) time-stepping schemes. A general form of Runge-Kutta with q stages can be applied to Eq. (1) to obtain the following discrete scheme:

$$\begin{aligned} u_i^n &= u^{n+c_i} + \Delta t \sum_{j=1}^{q} a_{ij} N[u^{n+c_j}], \quad i = 1, \ldots, q, \\ u_{q+1}^n &= u^{n+1} + \Delta t \sum_{j=1}^{q} b_j N[u^{n+c_j}], \end{aligned} \tag{4}$$

where $u^{n+c_j}(x) = u(t^n + c_j \Delta t, x)$ for $j = 1, \ldots, q$ and the superscript $n + c_j$ denotes the intermediate stages used to calculate the solution at u^{n+1}.

The idea of the PINN is to constrain the solution to abide by the physical laws of the system. In this way, the solution of the PDE is learned by the neural network, often implemented by leveraging the power of automatic differentiation.

3 Related works

Several works focus on analyzing the appropriate architecture for PINNs. In [10], a novel approach known as conservative PINNs (cPINNs) extends PINNs capability by introducing a discrete-domain decomposition for conservation laws. Later, it is further developed to extended PINNs (XPINNs) [11] where a space-time decomposition domain for PINNs with multiple MLPs is employed in the smaller subdomains of the system. Similarly, [12] proposed a parareal physics-informed neural network (PPINN) to accelerate the convergence of the PINN. Physics-informed attention-based neural networks (PIANNs) was introduced in [13]. The latter combines recurrent neural networks and attention mechanisms, specifically for solving the Buckley-Leverett problem. In [14], the authors introduced a backward-compatible version of physics-informed neural networks (bc-PINNs). This approach adopts a sequential learning strategy, where the PINN is trained across successive temporal segments.

4 Physics-Informed Antisymmetric Recurrent Neural Networks

In the physics-informed standard structure, we replace the MLP with the RNN. The latter relies on the stability of an antisymmetric property, and RK methods to improve the stability and expressivity of the network's structure used to train physics-informed models.

4.1 Recurrent Neural Networks

A Recurrent neural network (RNN) is a particular type of neural network used to learn a sequence of input $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_S)$ to a sequence of output $\boldsymbol{Y} = (\boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_S)$. A standard RNN is defined as:

$$\begin{aligned} \boldsymbol{h}_s &= f(\boldsymbol{V}_x \boldsymbol{x}_s + \boldsymbol{W}_h \boldsymbol{h}_{s-1} + \boldsymbol{b}_{\boldsymbol{h}}) \\ \boldsymbol{y}_s &= \boldsymbol{W}_y \boldsymbol{h}_s + \boldsymbol{b}_{\boldsymbol{y}}, \end{aligned} \tag{5}$$

Where $\boldsymbol{h}_s$ is the hidden state at time s, and $\boldsymbol{V}_x$, $\boldsymbol{W}_h$, $\boldsymbol{W}_y$ are the parameters. $\boldsymbol{b}_{\boldsymbol{h}}$ and $\boldsymbol{b}_{\boldsymbol{y}}$ are the bias terms. Here, f is an activation function.

4.2 Antisymmetric RNN

The RNN mentioned above has an unstructured representation of the hidden state. This limitation manifests as significant instability, which is further amplified by the recursive nature of the network architecture.

Given the following RNN hidden state:

$$\boldsymbol{h}_s = \boldsymbol{h}_{s-1} + \epsilon f\left(\boldsymbol{V}_x \boldsymbol{x}_s + (\boldsymbol{W}_h - \boldsymbol{W}_h^{\mathsf{T}} - \gamma \boldsymbol{I})\boldsymbol{h}_{s-1}\right) \tag{6}$$

where $\boldsymbol{W}_h \in \mathbb{R}^{n \times n}$ is the recurrent parameter, $\boldsymbol{V}_x \in \mathbb{R}^{n \times m}$ is the input parameter and $\boldsymbol{b}_{\boldsymbol{h}} \in \mathbb{R}^n$ is the bias term. $\boldsymbol{I}$ is the identity matrix. Scalars ϵ and γ are hyperameters. This approach combines the antisymmetric RNN proposed in [8] with the residual RNN in [15] to stabilize the learning of the PINN framework. The antisymmetric property is maintained by the recurrent parameter, which maintains the recurrent block stable.

Upper triangular antisymmetric RNN. In this work, we parameterize the recurrent weight $\boldsymbol{W}_h$ with a strictly upper triangular matrix. This results in a more efficient antisymmetric RNN with $\frac{n(n-1)}{2}$ degree of freedom.

4.3 RNN with Runge-Kutta methods

Given the recursive nature of the RK methods and RNN in Eq. (6), we proceed to implement a 2-layered RNN as:

$$\begin{aligned} \boldsymbol{h}_s^{(1)} &= \boldsymbol{h}_{s-1}^{(2)} + \epsilon f\left(\boldsymbol{V}_x^{(1)} \boldsymbol{x}_s + \overline{\boldsymbol{W}}^{(1)} \boldsymbol{h}_{s-1}^{(2)} + \boldsymbol{b}_h^{(1)}\right) \\ \boldsymbol{h}_s^{(2)} &= \boldsymbol{h}_s^{(1)} + \epsilon f\left(\boldsymbol{V}_x^{(2)} \boldsymbol{x}_s + \overline{\boldsymbol{W}}^{(2)} \boldsymbol{h}_s^{(1)} + \boldsymbol{b}_h^{(2)}\right), \end{aligned} \tag{7}$$

where $\boldsymbol{h}_s^{(1)}$ represents the hidden state in the first layer, and $\boldsymbol{h}_s^{(2)}$ the final hidden state in the second layer. The underscript s ranges from 0 to q, determining the number of iterations in the recurrent block. In Eq. (7) above, $\overline{\boldsymbol{W}}^{(1)}$ and $\overline{\boldsymbol{W}}^{(2)}$ are defined as $\boldsymbol{W}_h^{(1)} - \boldsymbol{W}_h^{(1)^{\mathsf{T}}} - \gamma \boldsymbol{I}$ and $\boldsymbol{W}_h^{(2)} - \boldsymbol{W}_h^{(2)^{\mathsf{T}}} - \gamma \boldsymbol{I}$, respectively. The structure of the standard PINN is represented in Fig. 1a, where the RNN in Fig. 1b replaces the MLPs, while maintaining the rest of the structure. The goal of the RNN is to process recursively the intermediate stages of the RK methods, a feature which is not found in feedforward MLP(s).

Table 1: Allen-Cahn - Single optimization: Adam

Runge-Kutta q	L_2 Norm Error		
	PIARNNs(LRS)	***PIARNNs***	***PINNs***
2	**2.4e-01**	3.5e-01	1.0e+00
4	**2.5e-01**	6.6e-01	7.3e-01
8	**2.5e-01**	**2.5e-01**	1.3e+00
16	**2.3e-01**	**2.3e-01**	9.2e-01
32	3.0e-01	**2.6e-01**	1.3e+00
64	**1.3e-01**	2.6e-01	2.8e-01
100	9.3e-01	**2.9e-01**	1.2e+00
Mean Error	3.3e-01	**3.2e-01**	1.0e+00

Table 2: Allen-Cahn - Dual optimization: Adam and L-BFGS

Runge-Kutta q	L_2 Norm Error		
	PIARNNs(LRS)	***PIARNNs***	***PINNs***
2	**2.2e-01**	3.9e-01	1.3e+00
4	6.0e-02	**1.0e-01**	1.6e-02
8	**8.2e-03**	1.9e-02	5.7e-02
16	2.2e-02	**1.4e-02**	5.6e-02
32	**1.1e-02**	6.0e-03	2.3e-01
64	1.6e-01	6.4e-03	**5.2e-03**
100	4.2e-02	**4.7e-03**	6.99e-03
Mean Error	**7.4e-02**	7.7e-02	2.3e-01

5 Experiments

5.1 Allen-Cahn Equation

Implementation. The Allen-Cahn equation is a fundamental equation used in various fields, particularly in physics, to describe phase separation processes. Given the following nonlinear Allen-Cahn equation along with periodic boundary conditions:

$$\begin{aligned} u_t - 0.0001u_{xx} + 5u^3 - 5u, \quad x \in [-1,1], \quad t \in [0,1] \\ u(0,x) = x^2\cos(\pi x), \quad u(t,-1) = u(t,1), \quad u_x(t,-1) = u_x(t,1) \end{aligned} \tag{8}$$

where u_t denotes the partial derivative of u w.r.t time t, indicating how u evolves over time. The second term involves the second partial derivative of u w.r.t the space x, scaled by a small coefficient.

If we apply the classical Runge-Kutta methods with an arbitrary number of q stages to the nonlinear operator $N[.]$ in Eq. (8), we obtain the following discretized Allen-Cahn equation:

$$N[u^{n+c_j}] = -0.0001u_{xx}^{n+c_j} + 5(u^{n+c_j})^3 - 5u^{n+c_j}. \tag{9}$$

Evaluation Measures. The loss function of the PIARNN has a similar structure as the PINN's loss function described in Eq. 2, with the following specific boundary loss:

$$\begin{aligned} SSE_b = \sum_{i=1}^{q} \left[u^{n+c_i}(-1) - u^{n+c_i}(1)\right]^2 + \left[u^{n+1}(-1) - u^{n+1}(1)\right]^2 \\ + \sum_{i=1}^{q} \left[u_x^{n+c_i}(-1) - u_x^{n+c_i}(1)\right]^2 + \left[u_x^{n+1}(-1) - u_x^{n+1}(1)\right]^2. \end{aligned} \tag{10}$$

Here, SSE_b enforces the boundary conditions in the loss function.

Results. In Table 1, we present a comparison of the performance between the PIARNN and the PINN, each utilizing a single optimizer, Adam. Remarkably, the PIARNN model trained either with or without learning rate scheduling (LRS) outperforms the standard PINN across various RK stages. Starting from $q = 2$ to $q = 100$ RK time-stepping stages, it improves incrementally the performance as the number of q increases. The mean performance of the PIARNN from $q = 2$ to $q = 100$ measured in L_2 norm error is 3.2e-01 compared to the mean performance of the PINN, which is 1.0e+00. It increased slightly to 4.1e-01 when the hidden size is 32 units, which significantly reduced the number of parameters required to search the solution.

As for the dual-optimization technique, where Adam and L-BFGS are used, the results are reported in Table 2. From $q = 2$ to $q = 100$, PIARNNs achieve state-of-the-art performance with a mean error of 7.4e-02 compared to 2.3e-01 achieved by PINNs. The proposed framework requires only a few parameters to attain optimal results.

6 Conclusion

In this work, a novel approach described as the PIARNN for solving PDEs is introduced. It combines RNN and Runge-Kutta methods. An upper triangular antisymmetric property is enforced in the recurrent connections of the RNN to stabilize the PINN's architecture. The PIARNN achieved superior prediction performance on the Allen-Cahn equation than the standard PINN, while maintaining fewer learnable parameters.

Acknowledgments

This work was supported by JSPS Kakenhi grant number JP23H03697.

References

[1] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[3] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems 27*. NIPS, 2014.

[4] 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," *Journal of Computational Physics*, vol. 378, pp. 686–707, 2019.

[5] M. De Florio, E. Schiassi, B. Ganapol, and R. Furfaro, "Physics-informed neural networks for rarefied-gas dynamics: Thermal creep flow in the Bhatnagar-Gross-Krook approximation," *Physics of Fluids*, vol. 33, no. 4, 2021.

[6] W. Ji, W. Qiu, Z. Shi, S. Pan, and S. Deng, "Stiff-PINN: Physics-informed neural network for stiff chemical kinetics," *Journal of Physical Chemistry A*, Sep 2021.

[7] M. De Florio, E. Schiassi, and R. Furfaro, "Physics-informed extreme theory of functional connections applied to optimal orbit transfer," *Chaos*, vol. 32, no. 6, p. 063107, 2022.

[8] B. Chang *et al.*, "AntisymmetricRNN: A dynamical system view on recurrent neural networks," in *7th International Conference on Learning Representations*, New Orleans, LA, USA, May 6-9 2019.

[9] A. G. Baydin *et al.*, "Automatic differentiation in machine learning: a survey," *Journal of Machine Learning Research*, vol. 18, no. 153, pp. 1–43, 2018.

[10] A. D. Jagtap, E. Kharazmi, and G. E. Karniadakis, "Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems," *Computer Methods in Applied Mechanics and Engineering*, vol. 365, p. 113028, 2020.

[11] A. D. Jagtap and G. E. Karniadakis, "Extended physics-informed neural networks (XPINNs): A generalized space-time domain decomposition-based deep learning framework for nonlinear partial differential equations," *Communications in Computational Physics*, vol. 28, no. 5, pp. 2002–2041, 2020.

[12] X. Meng, Z. Li, D. Zhang, and G. E. Karniadakis, "PPINN: Parareal physics-informed neural network for time-dependent PDEs," *Computer Methods in Applied Mechanics and Engineering*, vol. 370, p. 113250, 2020.

[13] R. Rodriguez-Torrado, P. Ruiz, L. Cueto-Felgueroso, M. C. Green, T. Friesen, S. Matringe, and J. Togelius, "Physics-informed attention-based neural network for hyperbolic partial differential equations: application to the Buckley-Leverett problem," *Scientific Reports*, vol. 12, no. 1, p. 7557, 2022.

[14] R. Mattey and S. Ghosh, "A novel sequential method to train physics-informed neural networks for Allen Cahn and Cahn Hilliard equations," *Computer Methods in Applied Mechanics and Engineering*, vol. 390, p. 114474, 2022.

[15] B. Yue, J. Fu, and J. Liang, "Residual recurrent neural networks for learning sequential representations," *Information*, vol. 9, no. 3, p. 56, 2018.