Structure Preserving PINN for Solving Reaction-Diffusion PDEs with Periodic Boundary

Baoli Hao

April 20, 2024

Abstract

In this paper, a structure preserving physics informed neural network (PINN) for solving various of reaction-diffusion type PDEs with periodic boundary is presented. The periodic boundary condition as the natural output of any deep neural net can be incorporated by this approach, hence significantly improving the training accuracy of baseline PINN. Together with mini-batching, the stiff PDEs for modeling reaction and diffusion processes can even be handled. The effectiveness of this PINNs on various PDEs from Allen Cahn to Gracy Scott are demonstrated.

Keywords: Physics-Informed Neural Network; Reaction-Diffusion PDE; Convection-Diffusion PDE.

1 Introduction

Reaction-Diffusion type PDEs play an important role in many science and engineering fields (Cherniha & Davydovych, 2017). Together with periodic boundary, these PDEs can model phenomena on infinite physical domain. However solving these types of PDEs can be challenging (Goussis, Valorani, Creta, & Najm, 2005): They often involve numerous parameters that govern the system's behavior, which makes them very sensitive to the parameters. Meanwhile, reaction-diffusion type PDEs may exhibit stiff behavior. Therefore, achieving convergence to accurate solutions of those PDES can be challenging, especially for high-dimensional or complex reaction-diffusion systems. In recent years, machine Learning based methodology has been steadily producing ground breaking scientific discoveries, from crystal identification, protein folding, to faster matrix multiplication (Fawzi et al., 2022; Noé, De Fabritiis, & Clementi, 2020; Spellings & Glotzer, 2018). These traditional data-driven machine learning methods often depend on large amounts of data for training. While effective in many cases, these approaches can sometimes generate models that lack interpretability for humans. This is because they focus solely on statistical patterns in the data without incorporating the underlying physical principles governing the problem domain. Furthermore, in many cases, it usually demands labor-intensive to collect or obtain data. Physics-informed machine learning is an emerging area that addresses both of these problems by introducing loss function that penalizes not satisfying known physical laws, allowing prediction of future performance of complex systems using sparse data (Cai, Wang, Lu, Zaki, & Karniadakis, 2021; Chen, Hosseini, Owhadi, & Stuart, 2021; Jin, Cai, Li, & Karniadakis, 2021; Liu & Wang, 2019, 2021; Rad, Viardin, Schmitz, & Apel, 2020; Raissi & Karniadakis, 2018; Raissi, Perdikaris, & Karniadakis, 2019; Yang, Barajas-Solano, Tartakovsky, & Tartakovsky, 2019; Zhu, Liu, & Yan, 2021). However, it is not easy to train physics-informed machine learning models successfully (Coutinho et al., 2023; McClenny & Braga-Neto, 2020; Wang, Teng, & Perdikaris, 2020; Wang, Yu, & Perdikaris, 2020). One of the basic issue is that it is difficult to integrate physical constraints. Balancing the incorporation of domain knowledge with the flexibility of neural networks is non-trivial and may vary depending on the specific application. Besides, the choice of loss function in PINNs is critical for effectively enforcing the physics constraints encoded in the PDE. Designing a loss function that balances data fidelity, regularization, and physics constraints can be non-trivial and may require domain-specific knowledge.

1.1 Related Work

While the standard Physics-Informed Neural Network (PINN) algorithm has proven effective across numerous applications, it may encounter difficulties when applied to certain types of partial differential equations (PDEs) known as "stiff" equations. There is a growing body of evidence that this occurs due to an imbalance between the data-fitting and residual components of the PINN loss function (Shin, Darbon, & Karniadakis, 2020; Wang, Teng, & Perdikaris, 2020; Wang, Yu, & Perdikaris, 2020; Wight & Zhao, 2020). Gradient descent will drive the residual loss to zero faster than the data-fitting component, which prevents convergence to the correct solution. In forward problems, this problem may be especially acute, since all the data are confined to initial and boundary conditions, and there may not be any data inside the domain of the PDE. For example, in time-evolution problems, it may

be difficult for neural network training to advance the information in the initial condition to later times, as has been observed by several authors, e.g. Haitsiukevich and Ilin (2022); Krishnapriyan, Gholami, Zhe, Kirby, and Mahoney (2021); McClenny and Braga-Neto (2020); Wang, Sankaran, and Perdikaris (2022); Wight and Zhao (2020). In Wight and Zhao (2020), an approach to propagate the information forward in time was proposed, whereby the time axis is divided into several smaller intervals, and PINNs are trained separately on them, sequentially, beginning with the interval closes to the initial condition; this approach is time-consuming due to the need to train multiple PINNs. It was pointed out in both Wight and Zhao (2020) and McClenny and Braga-Neto (2020) that it is necessary to weight the initial condition data and residual points near $t = 0$ in order to propagate the information forward in a well-known nonlinear Allen-Cahn PDE benchmark. In the case of the self-adaptive PINNs in McClenny and Braga-Neto (2020), this fact is automatically discovered during training, and again in a Wave equation benchmark with a complex initial condition. The same theme appears in Krishnapriyan et al. (2021), where a "sequence-to-sequence" approach is proposed, a time-marching scheme where the PINN learns each time step sequentially. In Wang et al. (2022), the authors attribute the problem to a lack of "causality" in standard PINN training, and propose to correct this by a weighting scheme that progressively weights the data from early to later times, using the information from the PDE residue itself, in a manner that is reminiscent of the self-adaptive weighting in McClenny and Braga-Neto (2020). Finally, a pseudo-label approach for training PINNs, which is essentially identical to our PINN self-training approach, has been proposed recently, and independently, in Haitsiukevich and Ilin (2022).

1.2 Contributions of this Work

- In this paper, we develop structure preserving PINN algorithms, and investigate their stability, and prediction accuracy. The structure preserving PINN is similar to the method proposed independently in (Dong & Ni, 2021). However in that paper, PDE is built into the structure; whereas ours built the IC/BC into the NN.
- We investigate how structure preserving ameliorates the issue of propagating information forward in time, which is a common failure mode of PINNs.

2 Methodology

In this section, a brief review of the standard physics informed neural network will be introduced (Raissi, Perdikaris, & Karniadakis, 2017). We build IC/BC into the NN and embed mini-batching strategy into this approach.

2.1 Standard PINN Algorithm

We consider the viscous Burger's equation (convection-diffusion Equation) as an example to elucidate the idea behind standard PINN:

$$\begin{aligned} &u_t + uu_x - (0.01/\pi)u_{xx} = 0, \quad x \in [-1, 1], \quad t \in [0, 1], \\ &u(0, x) = -\sin(\pi x), \qquad u(t, -1) = u(t, 1) = 0. \end{aligned} \tag{1}$$

To solve this equation, Raissi et al. (2017) first introduced the neural network that maps the domain coordinates to state values:

$$\mathcal{U} : (x, t) \to \mathcal{U}(x, t),$$

and then the PDE residual was specified:

$$\mathcal{F} : (x, t) \to \mathcal{U}_t(x, t) + \mathcal{U}(x, t)\mathcal{U}_x(x, t) - (0.01/\pi)\mathcal{U}_{xx}(x, t).$$

Given the initial data points $\{(0, x_u^i, u_i)\}_{i=1}^{N_u}$ and random samples from boundary $\{(t_b^i, -1), (t_b^i, 1)\}_{i=1}^{N_b}$ and bulk $\{(t_f^i, x_f^i), (t_f^i, x_f^i)\}_{i=1}^{N_f}$, the loss function can therefore be specified, i.e.

$$\mathcal{L} = MSE_{IC} + MSE_{BC} + MSE_R,$$

where

$$MSE_{IC} = \frac{1}{N_u}\sum_{i=1}^{N_u} |\mathcal{U}(0, x_u^i) - u^i|^2,$$

$$MSE_{BC} = \frac{1}{N_b}\sum_{i=1}^{N_b} [|\mathcal{U}(t_u^i, 1)|^2 + |\mathcal{U}(t_u^i, -1)|^2],$$

$$MSE_R = \frac{1}{N_r}\sum_{i=1}^{N_r} |\mathcal{F}(t_f^i, x_f^i)|^2,$$

where N_u, N_b, N_f are the number of initial data, boundary data and bulk data respectively. Once the loss function is specified, we can simply proceed with training the neural network to find the best parameters $\boldsymbol{\theta}^*$ by minimizing the loss function.

2.2 Enforcing Periodic Conditions with DNN

One of the main difficulties of the training PINN (i.e. finding the minimizer) is that PINNs tend to get stuck at one of the local minimizers for the multi-objective optimization problem.

To solve this issue, we compose a DNN-presented function, with a set of independent known periodic functions. We first consider the sinusoidal functions:

$$p(x) = A\cos(\omega x + \phi) + C, \text{with } \omega = \frac{2\pi}{L}.$$

Let $\sigma(\cdot)$ denote a nonlinear activation function:

$$v(x) = \sigma(p(x)) = \sigma(A\cos(\omega x + \phi) + C).$$

we then consider an arbitrary function $u(x)$ represented by a DNN, and define

$$\mathcal{U}_{NN} : (x,t) \to v(x,t) \to \mathcal{U}_{NN}(v(x,t)).$$

Finally, we consider this transformation:

$$\mathcal{U}_{NN} : (x,t) \to \mathcal{U}_{NN}(v(x,t)) \tag{2}$$

$$\mathcal{U} : \mathcal{U} = \mathcal{U}_0(x) + \phi(t)\,\mathcal{U}_{NN}(v(x,t)), \tag{3}$$

where $\phi(t)$ is a smooth function of t and $\phi(0) = 0$. Hence, since the IC and BC are built in the new neural network, the loss function can be redefined as:

$$\mathcal{L} = MSE_R. \tag{4}$$

2.3 Mini-batching

Mini-batching is a method commonly employed in deep learning to enhance performance. Instead of computing the gradient direction using the entire dataset, mini-batching involves using a smaller subset of the data, called a batch or mini-batch, to estimate the direction. Kleinberg, Li, and Yuan (2018) has demonstrated that mini-batching can be beneficial in steering clear of suboptimal local minima more effectively compared to using the entire dataset in full-batch gradient descent.

In this paper, we explore the mini-batching technique and find that it does indeed help the trained neural network converge more effectively when approximating reaction-diffusion equations.

3 Numerical Experiments

In this section, we will evaluate our proposed PINN scheme in the previous section on multiple prototypical reaction-diffusion PDEs with periodic boundary.

First of all, we use a Latin hypercube sampling (LHS) strategy to sample the collocation points in the whole domain including the boundary. Through the construction of the transformed u-network and the specification of appropriate loss function, we train our neural network using the generated points. Next, we utilize the trained model to approximate the solution to the partial differential equations (PDEs).

To ascertain the accuracy of our results, we use Chebfun, a spectral-style system implemented in MATLAB for handling functions in an object-oriented manner, allowing us to obtain solutions with high accuracy against which we can compare the performance of our scheme rigorously. Given the data points $\{x_i, t_i\}_{i=1}^N$ with N the total number of points, we take relative L_2 norm of the "exact" value $u(x_i, t_i)$ at those points and the network output $\mathcal{U}(x_i, t_i)$ at those points to evaluate the accuracy of our trained model:

$$\mathcal{E} = \frac{\sqrt{\sum_{i=1}^N |\mathcal{U}(x_i,t_i) - u(x_i,t_i)|^2}}{\sqrt{\sum_{i=1}^N |u(x_i,t_i)|^2}}.$$

3.1 Solving the Allen-Cahn Equation

Allen-Cahn type PDE is a classical phase-field model, has been widely used to investigate interfacial dynamic problems. We first tested the one-dimensional Allen-Cahn Equation with periodic boundary conditions, as follows:

$$\begin{aligned}
&u_t - \gamma_1 u_{xx} + \gamma_2 u^3 - \gamma_2 u = 0, \quad (t,x) \in (0,T) \times (a,b),\\
&u(0,x) = u_0(x), \quad x \in [a,b],\\
&u(t,a) = u(t,b), \quad t \in [0,T],\\
&u_x(t,a) = u_x(t,b), \quad t \in [0,T],
\end{aligned} \tag{5}$$

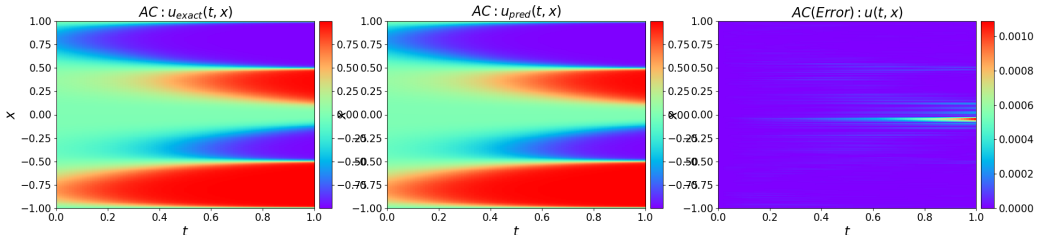

Figure 1: Exact solution of the 1D Allen-Cahn with the corresponding network prediction and the absolute error difference.

where $\gamma_1, \gamma_2 > 0, T > 0, a < b$ are prescribed constants. As γ_2 increases, the transition interface of the solutions is sharper, which makes it harder to solve the AC equation numerically. Therefore, we demonstrate the effectiveness of our scheme by testing on AC PDE with a large γ_2:

$$u_0(x) = x^2 \cos(\pi x), \quad T = 1, \quad a = -1, \quad b = 1, \quad \gamma_1 = 0.001, \quad \gamma_2 = 5.$$

We train a PINN solution using the following parameters in Table 1.

# Co. Points	# Layers	# Neurons
$16,384$	7	32

Table 1: PINN Params

It trains with $50k$ Adam steps with learning rate at $5 \cdot 10^{-3}$ first, and then uses the L-BFGS-B optimizer to fine-tune the neural network. The relative L_2 error is $1e-3$. Figure 2 and Figure 1 show the results:

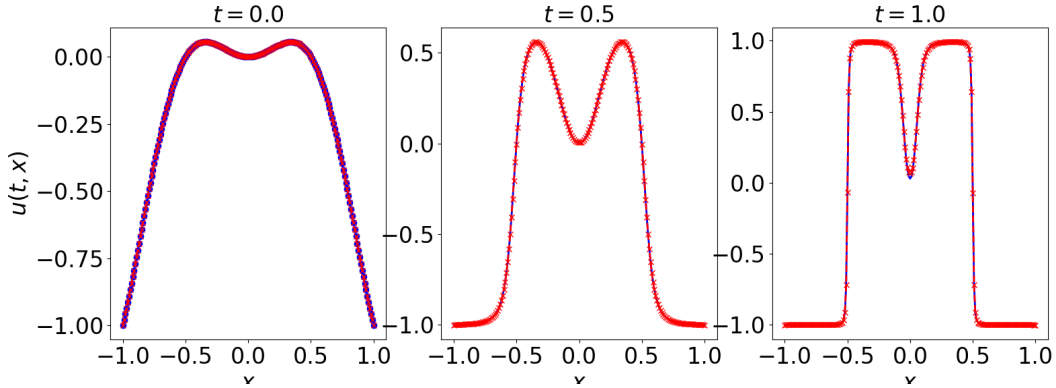

Figure 2: Solutions of the AC equation $\gamma_2 = 5$.

Table 2 compares the errors among the three approaches: standard PINN, re-sampling technique prposed by Wight and Zhao (2020) and our approach. Unfortunately, using the standard PINN approach alone, we were not able to learn the accurate solution for the Allen-Cahn equation. The dynamics of equation can not be captured. The relative error is almost equal to 1. Compared with the technique proposed by Wight and Zhao (2020), our error is much less and the accuracy is improved.

Allen-Cahn	Standard PINN	Re-sampling (Wight and Zhao (2020))	Our Approach
Relative L_2	9.90e-1	2.33e-2	9.16e-4
Relative L_1	9.90e-1	6.20e-3	4.84e-4
L_∞ norm	9.96e-1	2.64e-1	1.43e-1

Table 2: Comparison of errors in the learned solutions of the Allen-Cahn equation using various PINN approaches

3.2 Solving the Kuramoto-Sivashinsky equation

Kuramoto–Sivashinsky equation is a fourth-order nonlinear partial differential equation, known for its chaotic behavior, as follows:

$$\begin{aligned} &u_t = -u_{xx} - u_{xxxx} - uu_x, \quad (t,x) \in (0,T] \times (a,b), \\ &u(0,x) =, \quad x \in [a,b], \\ &u(t,a) = u(t,b), \quad t \in [0,T], \end{aligned} \tag{6}$$

where $T = 20, a = 0, b = 32\pi$, and $u_0(x) = \cos\frac{x}{16}(1 + \sin\frac{x-1}{16})$. The parameters are the same with Table 1. The results are shown in Figure 3 and 4:

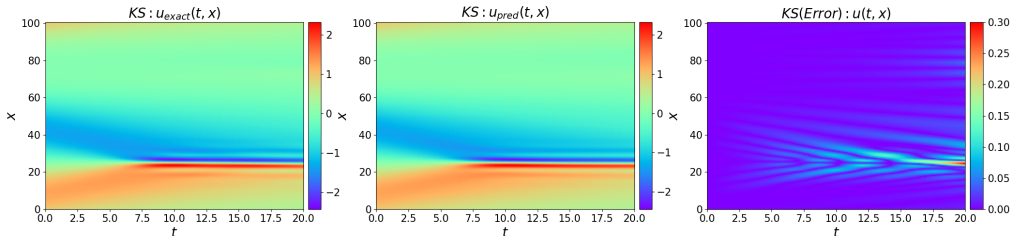

Figure 3: Exact solution of the 1D KS equation with the corresponding network prediction and the absolute error difference.

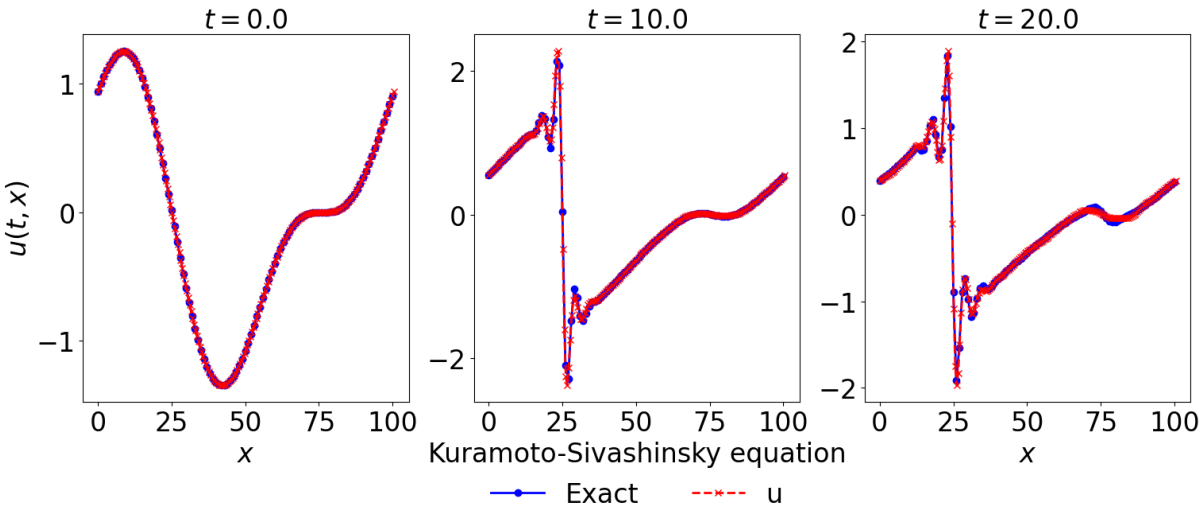

Figure 4: Solutions of the KS equation.

3.3 Solving the Gray-Scott Equation

Reaction and diffusion of chemical species can produce a variety of patterns, reminiscent of those often seen in nature. The Gray-Scott equations serve as a mathematical model for such chemical reactions:

$$\begin{aligned} U + 2V &\longrightarrow 3V, \\ V &\longrightarrow P. \end{aligned} \tag{7}$$

This system is defined by two equations that describe the dynamics of two reacting substances:

$$\begin{aligned} &u_t = \epsilon_1 u_{xx} + b(1-u) - uv^2, \quad (t,x) \in (0,T] \times (-L,L), \\ &v_t = \epsilon_2 v_{xx} - (b+k)v + uv^2, \\ &u(0,x) = u_0(x), \quad v(0,x) = v_0(x), \quad \forall x \in [-L,L], \\ &u(t,-L) = u(t,L), \quad v(t,-L) = v(t,L), \quad \forall t \in [0,T], j \end{aligned} \tag{8}$$

where $T = 20, L = 50, \epsilon_1 = 1, \epsilon_2 = 0.01$ are diffusion rates, $b = 0.02$ is the "feeding rate" that adds U, $k = 0.0562$ is the "killing rate" that removes V. We set our initial conditions as:

$$u_0(x) = 1 - \frac{\sin(\pi(x-50)/100)^4}{2}, \quad v_0(x) = \frac{\sin(\pi(x-50)/100)^4}{4}.$$

The parameters are the same with before. Figures 5, 6 and 7 show the results:

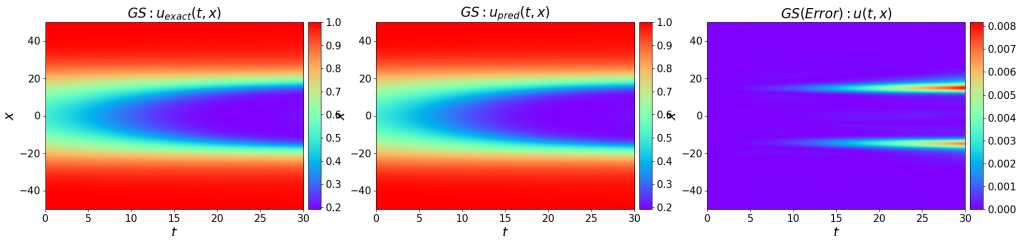

Figure 5: Exact solution of the 1D GS equation (u specie) with the corresponding network prediction and the absolute error difference.

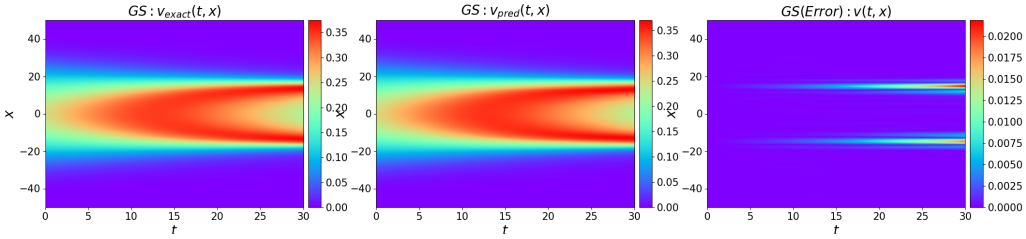

Figure 6: Exact solution of the 1D GS equation (v specie) with the corresponding network prediction and the absolute error difference.

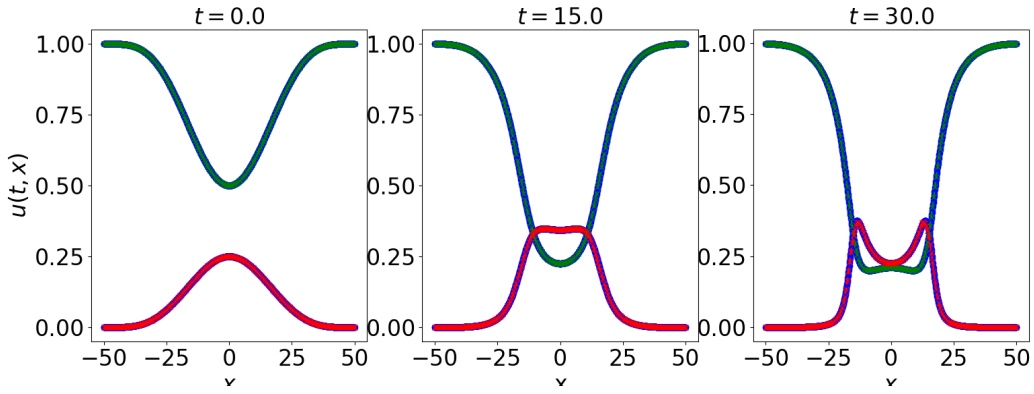

Figure 7: Solutions of the GS equation.

3.4 Solving the Belousov-Zhabotinsky equation

A Belousov–Zhabotinsky reaction is one of a class of reactions that serve as a classical example of non-equilibrium thermodynamics. It typically involves the oxidation of organic compounds by bromine in an acidic medium. The

system is as follows:

$$
\begin{aligned}
&u_t = \epsilon_1 u_{xx} + u + v - uv - u^2, \\
&v_t = \epsilon_2 v_{xx} + w - v - uv, \\
&w_t = \epsilon_1 w_{xx} + u - w, \quad (t,x) \in (0,T] \times [-L,L] \\
&u(0,x) = u_0(x), \quad v(0,x) = v_0(x), \quad w(0,x) = w_0(x), \quad \forall x \in [-L,L] \\
&u(t,-L) = u(t,L), \quad v(t,-L) = v(t,L), \quad w(t,-L) = w(t,L), \quad \forall t \in [0,T],
\end{aligned}
\tag{9}
$$

where $T = 3, L = 1$, $\epsilon_1 = 10^{-5}, \epsilon_2 = 2 \times 10^{-5}$ are diffusion rates. We let the initial conditions be:

$$u_0(x) = \exp(-100(x+0.5)^2), \quad v_0(x) = \exp(-100x^2), \quad w_0(x) = \exp(-100(x-0.5)^2).$$

The results are:

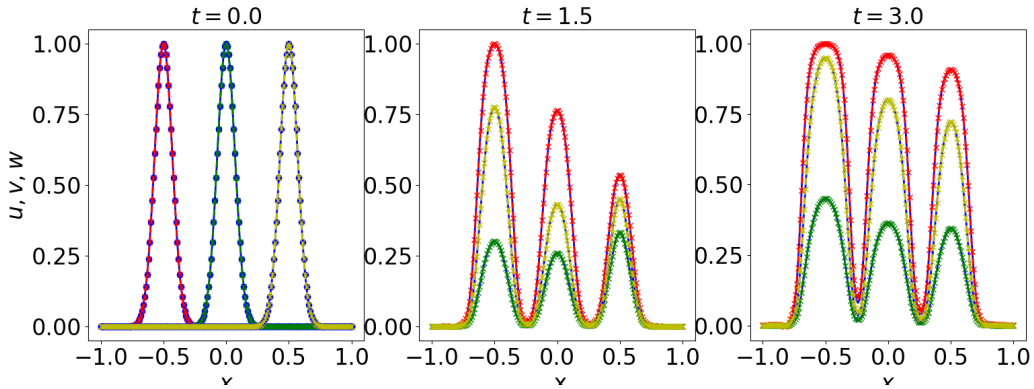

Figure 8: Solutions of the BZ equation.

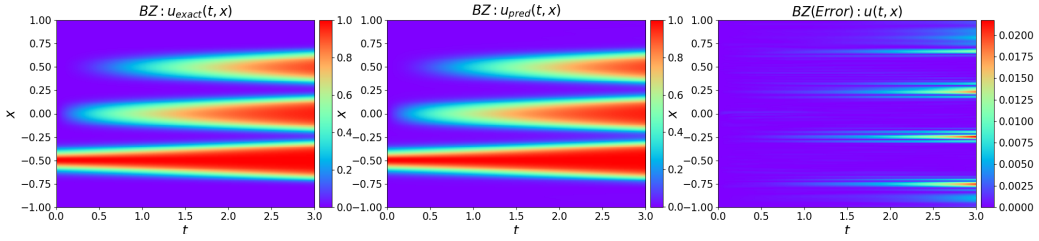

Figure 9: Exact solution of the 1D BZ equation (u specie) with the corresponding network prediction and the absolute error difference.

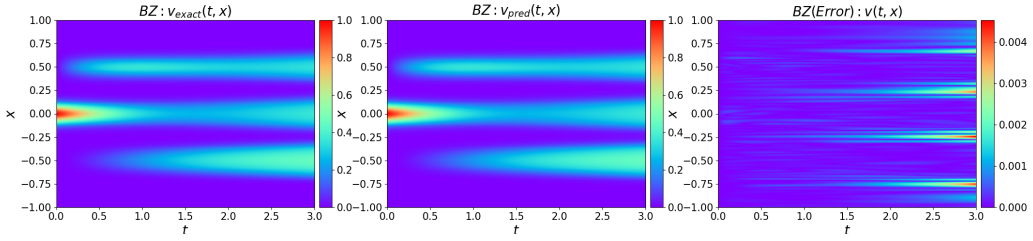

Figure 10: Exact solution of the 1D BZ equation (v specie) with the corresponding network prediction and the absolute error difference.

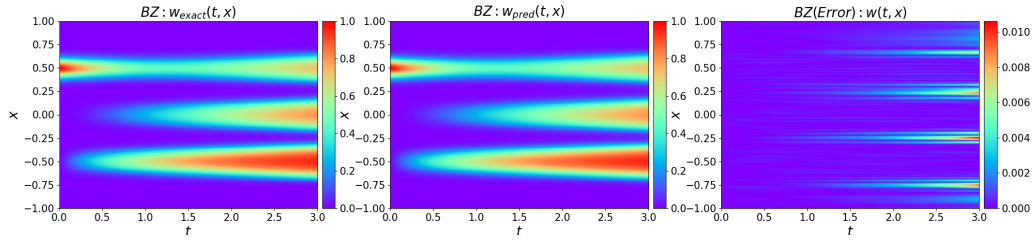

Figure 11: Exact solution of the 1D BZ equation (w specie) with the corresponding network prediction and the absolute error difference.

4 Discussion

Although we have made more accurate predictions of PDE solutions by our approach, there are still problems. Unstable results may happen for some PDEs with certain parameters due to optimization not being able to find the minimizer. Meanwhile, another big issue is that we can not train well for long-time behavior.

In the future, along with tackling the above issues, we will also explore 2D and 3D cases. Besides, we will investigate solution of more complicated PDEs, i.e. Keller-Segel Equation, Bertozzi's PDE, etc. Finally, we want to assume other transformations that embedded IC and BC into the new neural networ in the future, i.e., optimal choice of $\phi(t)$ and other type of periodic functions.

5 Github Link

A github link for this paper containing all of the codes is provided: https://github.com/baolihao/MMAE500_Final_Project.

References

Cai, S., Wang, Z., Lu, L., Zaki, T. A., & Karniadakis, G. E. (2021). DeepM&Mnet: Inferring the electroconvection multiphysics fields based on operator approximation by neural networks. *Journal of Computational Physics*, *436*, 110296.

Chen, Y., Hosseini, B., Owhadi, H., & Stuart, A. M. (2021). Solving and learning nonlinear PDEs with Gaussian processes. *Journal of Computational Physics*, *447*, 110668.

Cherniha, R., & Davydovych, V. (2017). Nonlinear reaction-diffusion systems. *Springer Lecture Notes in Mathematics LNM*, *2196*.

Coutinho, E. J. R., Dall'Aqua, M., McClenny, L., Zhong, M., Braga-Neto, U., & Gildin, E. (2023). Physics-informed neural networks with adaptive localized artificial viscosity. *Journal of Computational Physics*, *489*, 112265. Retrieved from https://www.sciencedirect.com/science/article/pii/S0021999123003601 DOI: https://doi.org/10.1016/j.jcp.2023.112265

Dong, S., & Ni, N. (2021). A method for representing periodic functions and enforcing exactly periodic boundary conditions with deep neural networks. *Journal of Computational Physics*, *435*, 110242.

Fawzi, A., Balog, M., Huang, A., Hubert, T., Romera-Paredes, B., Barekatain, M., ... others (2022). Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, *610*(7930), 47–53.

Goussis, D. A., Valorani, M., Creta, F., & Najm, H. N. (2005). Reactive and reactive-diffusive time scales in stiff reaction-diffusion systems. *Progress in Computational Fluid Dynamics, an International Journal*, *5*(6), 316–326.

Haitsiukevich, K., & Ilin, A. (2022). Improved training of physics-informed neural networks with model ensembles. *arXiv preprint arXiv:2204.05108*.

Jin, X., Cai, S., Li, H., & Karniadakis, G. E. (2021). NSFnets (Navier-Stokes flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations. *Journal of Computational Physics*, *426*, 109951.

Kleinberg, B., Li, Y., & Yuan, Y. (2018). An alternative view: When does sgd escape local minima? In *International conference on machine learning* (pp. 2698–2707).

Krishnapriyan, A., Gholami, A., Zhe, S., Kirby, R., & Mahoney, M. W. (2021). Characterizing possible failure modes in physics-informed neural networks. *Advances in Neural Information Processing Systems*, *34*, 26548–26560.

Liu, D., & Wang, Y. (2019). Multi-fidelity physics-constrained neural network and its application in materials modeling. *Journal of Mechanical Design*, *141*(12).

Liu, D., & Wang, Y. (2021). A dual-dimer method for training physics-constrained neural networks with minimax architecture. *Neural Networks*, *136*, 112–125.

McClenny, L., & Braga-Neto, U. (2020). Self-adaptive physics-informed neural networks using a soft attention mechanism. *arXiv preprint arXiv:2009.04544*.

Noé, F., De Fabritiis, G., & Clementi, C. (2020). Machine learning for protein folding and dynamics. *Current opinion in structural biology*, *60*, 77–84.

Rad, M. T., Viardin, A., Schmitz, G., & Apel, M. (2020). Theory-training deep neural networks for an alloy solidification benchmark problem. *Computational Materials Science*, *180*, 109687.

Raissi, M., & Karniadakis, G. E. (2018). Hidden physics models: Machine learning of nonlinear partial differential equations. *Journal of Computational Physics*, *357*, 125–141.

Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2017). Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561*.

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.

Shin, Y., Darbon, J., & Karniadakis, G. E. (2020). On the convergence and generalization of physics informed neural networks. *arXiv preprint arXiv:2004.01806*.

Spellings, M., & Glotzer, S. C. (2018). Machine learning for crystal identification and discovery. *AIChE Journal*, *64*(6), 2198–2206.

Wang, S., Sankaran, S., & Perdikaris, P. (2022). Respecting causality is all you need for training physics-informed neural networks. *arXiv preprint arXiv:2203.07404*.

Wang, S., Teng, Y., & Perdikaris, P. (2020). Understanding and mitigating gradient pathologies in physics-informed neural networks. *arXiv preprint arXiv:2001.04536*.

Wang, S., Yu, X., & Perdikaris, P. (2020). When and why PINNs fail to train: A neural tangent kernel perspective. *arXiv preprint arXiv:2007.14527*.

Wight, C. L., & Zhao, J. (2020). Solving Allen-Cahn and Cahn-Hilliard equations using the adaptive physics informed neural networks. *arXiv preprint arXiv:2007.04542*.

Yang, X., Barajas-Solano, D., Tartakovsky, G., & Tartakovsky, A. M. (2019). Physics-informed cokriging: A Gaussian-process-regression-based multifidelity method for data-model convergence. *Journal of Computational Physics*, *395*, 410–431.

Zhu, Q., Liu, Z., & Yan, J. (2021). Machine learning for metal additive manufacturing: predicting temperature and melt pool fluid dynamics using physics-informed neural networks. *Computational Mechanics*, *67*(2), 619–635.