Particle Swarm Training and Uncertainty quantification of Physics Informed Neural Networks

Sharaj Panwar
Department of Electrical and Computer Engineering
Texas A&M University
College Station, Texas
sharaj@tamu.edu

Allison Kaye Arabelo
Department of Materials Science and Engineering
Texas A&M University
College Station, Texas
aiarabelo@tamu.edu

Abstract

Physics-informed neural networks (PINNs) have recently gained popularity as a new machine Learning direction to solve partial differential equations (PDEs) based scientific and engineering problems. Despite the recently surceases in solving PDEs, the training efficiency and converges remain the active area of research and need further exploration. PINNs training by gradient descent have shown to displays pathologies and stiffness in the gradient flow dynamics causing poor approximation to the latent solution. In this report, we explore the use of a hybrid particle swarm optimization to train PINNs which has been recently proposed in PSO-PINNs paper. Several Particles' combinations which different hypermeters are explored. The resulting PSO-PINN algorithm successfully alleviate the pathologies and the undesired behaviors of PINNs trained with standard gradient descent and consistently outperforms a baseline PINN trained with Adam gradient descent. Furthermore, the PSO-PINNs provide an ensemble approach to PINN that naturally provide the quantified uncertainty of predictions.

1 Introduction

Physics-informed neural networks (PINNs) have recently emerged as an alternative way of solving partial differential equations (PDEs) without the need of building elaborate grids, instead, using a straightforward implementation proving to be useful in wide range of PDEs based engineering and scientific problems.(1) The residual of PDE solution is combined with the mismatch in the given data of the solution to formulate the loss function. PINNs use the PDE as a regularization term in the loss function.(2)(3) However, the effect of this regularization is not fully understood and PINNs often have difficulties in constructing an accurate approximation to the exact latent solution.(4) These pathologies can arise even in the simplest possible setting corresponding to solving classical linear elliptic equations(5)(6). This mode of failure is related to stiffness in the gradient flow dynamics. Which leads to an unstable imbalance in the magnitude of the back-propagated gradients during model training using gradient descent. The minimizing the PDE residual dominates the total PINNs' training process, consequently making our trained model heavily biased towards returning a solution that leads to a small error in the PDE residual, while a large error is attained in fitting the boundary conditions. With the erroneous boundary and initial conditions, a PDE solution may have infinitely

ECEN 689: Scientific Machine Learning Class Project, Spring 2022, TAMU, College Station.

many solutions. Hence, it is natural to apply more weight on the Initial and Boundary conditions than the residual to direct the training toward the unique solution.

One important aspect of designing scientific and engineering systems is analyzing their performance in accordance with underlying uncertainty. The PINNs framework is effective but lacks uncertainty quantification of the solution presented due to inherent randomness in the data and/or due to the approximation limitations of neural networks.(7) The uncertainty quantification in Neural Network based models is complicated because in addition to Aleatoric uncertainty associated with noisy data, there is also uncertainty due to limited data, NN hyperparameters, over parameterization, optimization and sampling errors as well as model miss specification.(8) There is a great interest in quantifying the total uncertainty of Physics Informed Neural Networks. There are two main categories of uncertainty quantification in scientific machine learning, one such category is the Bayesian based methods such as variational method, flow based methods, and arbitrary polynomial chaos method, and second category is neural architecture based methods such as Drop out method and ensemble method (8), (9).

The objective is to review use of a hybrid particle swarm optimization and gradient descent approach to train PINNs proposed in (10) (11). The resulting PSO-PINN algorithm mitigates the undesired behaviors of PINNs trained with standard gradient descent. It also presents a natural ensemble approach incorporated within PINNs proposed framework for uncertainty quantification. The swarms are a well establish algorithm, they also provide an ensemble of solutions. This category of learning combines different models such that the final one has improved generalization performance. This approach also leads naturally to uncertainty quantification, as the co-variance matrix can be considered as a measure of the uncertainty of the ensemble. Furthermore, different training aspects of this work are explored.

2 Method

2.1 Physics Informed Neural Networks

Consider a physical system governed by PDEs of the general form:

$$\mathcal{N}[u(s)] = f(s), \qquad s \in S \tag{1}$$

$$\beta[u(s)] = g(s), \qquad s \in \Gamma \tag{2}$$

where $\mathcal{N}$ is a general differential operator, $u(s)$ are the field variables of interest, $f(s)$ is the source field. B is the operator for boundary conditions defined on the boundary of the domain S.
Solving the above system of PDEs is to estimate the quantity of interest (QOI) i.e. $u(s)$. The solution $u(s)$ can be approximated by employing a neural network $f_\theta(s)$ by minimizing the residual loss:

$$\mathcal{L}_r = \frac{1}{n_r}\sum_{i=1}^{n_r}[\mathcal{N}[f_\theta(s_i^r)] - f(s_i^r)]^2, \tag{3}$$

where $\{s_i^r\}_{i=1}^{n_r} \subset S$ are collocation points, and $N[f_\theta(s)]$ is computed using automatic differentiation.The neural network is fitted to the boundary condition $g(s)$ using a traditional data-fitting loss function:

$$\mathcal{L}_b = \frac{1}{n_b}\sum_{i=1}^{n_b}[f_\theta(s_i^b)] - g(s_i^b)]^2, \tag{4}$$

where $\{s_i^b\}_{i=1}^{n_b} \subset \Gamma$ are boundary points. The total loss function to be minimized is the sum of the losses:

$$\mathcal{L} = \mathcal{L}_r + \mathcal{L}_b. \tag{5}$$

This generic framework can be extended to more complex boundary conditions.

2.2 Particle Swarm Optimization

The Particle Swarm Optimization (PSO) algorithm(12)(13) is a population-based stochastic optimization algorithm that emulates the swarm behavior of particles distributed in a n-dimensional search space(14). Each individual in this swarm represents a candidate solution. At each iteration, the particles in the swarm exchange information and use it to update their positions. Particle $\theta(t)$

at iteration t is guided by a velocity determined by three factors: its own velocity inertia $\beta V(t)$, its best-known position p_{best} in the search-space, as well as the entire swarm's best-known position g_{best}:

$$V(t+1) = \beta V(t) + c_1 r_1(p_{best} - \theta(t)) + c_2 r_2(g_{best} - \theta(t)), \tag{6}$$

where c_1 and c_2 are the cognitive and social coefficients, respectively, and r_1 and r_2 are uniformly distributed random numbers in range [0; 1). Then the particle position is updated as

$$\theta(t+1) = \theta(t) + V(t+1). \tag{7}$$

Many variations of the PSO algorithm were proposed over time, including a hybrid PSO and gradient based algorithm known as PSO-BP[28], which adds a gradient descent component to the swarm framework. This strategy came particularly handy when applying the method to train neural networks since all partial gradients can be computed efficiently by backpropagation. In the PSO-BP algorithm, the particle's velocity is updated via:

$$V(t+1) = \beta V(t) + c_1 r_1(p_{best} - \theta(t)) + c_2 r_2(g_{best} - \theta(t)) - \alpha \nabla \mathcal{L}(\theta(t)), \tag{8}$$

where α is the learning rate and $\nabla \mathcal{L}(\theta(t))$ is the loss gradient. Therefore, the gradient participates in the velocity magnitude and direction, by an amount that is specified by the learning rate.

2.3 Ensemble

The simulation of scientific systems inhabit intrinsic and extrinsic uncertainties coming from data, parameters and approximations etc. Bayesian modelling as a popular method for epistemic uncertainty quantification approximate the posterior distribution $p(\theta|S)$. The maximum of this posterior is selected as a NN approximation to predict the quantity of interest along with a credible interval to quantify the epistemic uncertainty. Ensembles, instead of approximating the posterior aims to obtain multiple modes of $p(\theta|S)$ by obtaining multiple minima of the NN parameter loss landscape. Apart from being a well-establish optimization method, the PSO provide a natural ensemble of solutions which as a set of parameters $\{\hat{\theta}_j\}_{j=1}^{M}$ can be used exactly as the posterior samples obtained with Bayesian techniques. Thus, we can take advantage of the various properties provided by ensembles. This category of learning combines different models such that the final one has improved generalization performance(15). Instead of simply choose the most fitted model, we will summarize the ensemble's prediction by model averaging.

$$\hat{f}_\theta(s) = \frac{1}{|\Theta|} \sum_{\theta_i \in \Theta} f_{\theta_i}(s), \tag{9}$$

The uncertainty quantification is performed by considering the covariance matrix as a measure of the uncertainty of the ensemble:

$$\Sigma_{f_\theta(s)} = \frac{1}{|\Theta| - 1} \sum_{\theta_i \in \Theta} (f_{\theta_i}(s) - \hat{f}_\theta(s))(f_{\theta_i}(s) - \hat{f}_\theta(s))^T, \tag{10}$$

3 Experiments

The experiments are performed to get the optimal ensembles of particle swarms. We tune the hyperparameters such as number of particles, collocation points, and whether or not a starting velocity is used. The best fit models are selected and a comparative study is performed in order to determine the effect of varying hyperparametrs in this section. For the sake of understanding and to truly evaluate the potential of the reviewed method, we select the classic Poisson equation, Burgers' equation, and the advection equation. We enlist our findings in the subsections below.

3.1 Poisson equation

The PSO algorithm was tested on the classic Poisson equation, which is an elliptic ordinary differential equation. It is defined in Eq. 1:

$$\begin{aligned} u_{xx} &= g(x), x \in [0,1], \\ u(0) &= a, \\ u(1) &= b. \end{aligned} \tag{11}$$

In this project, $g(x) = u(x) = \sin(2\pi x)$, and the boundaries are chosen to be $u(0) = u(1) = 0$ for simplicity.

The particle swarm optimizer is selected to have a particle inertia of 0.9, a p_{best} coefficient of 0.8, a g_{best} coefficient of 0.5, with weights generated from -1 to 1. The learning rate was selected to be 1×10^{-5}. For the purpose of benchmarking using the Poisson equation, the number of particles in the swarm was varied, using a particle population of 50, 100, and 250. The swarm was selected to have a cold start, and thus the initial velocity was kept 0.

Figure 1 shows the effect of varying the number of particles. It can be seen that when the number of particles is increased, the approximation produced by the neural network (the mean) improves. Moreover, the uncertainty is reduced, as demonstrated by the narrowing uncertainty band. These results suggest that a higher number of particles in the swarm would improve the accuracy of the training, given that other parameters are held constant.

It should be noted however that an increased particle population would lead to more calculations being performed, since each particle is effectively attempting to find the minima. This therefore would increase the computational cost of the simulation. It is thus very apparent that tuning the particle size to balance computational cost and accuracy is necessary when using PSO.

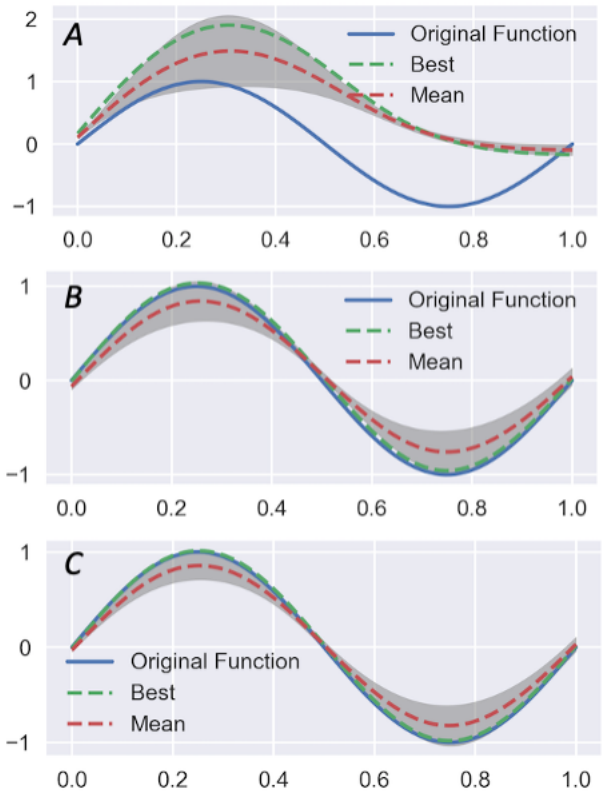

Figure 1: The training results after 90 iterations when the number of particles is varied from ***A:*** 50, ***B:*** 100, and ***C:*** 250.

When the algorithm is initialized with a non-zero starting velocity, instability in the training of neural network seems to be induced. It can be seen in Figure 2 that when the swarm is initialized with a cold start, the magnitude of the loss is very low in comparison to the loss when it is initialized with a velocity. Moreover, it converges much more steadily rather than erratically, in contrast to Fig. 2B. Even though it seems that convergence is reached slower when using a cold start, it should be noted that the uncertainty is much more significant in doing so. In contrast, the PSO algorithm with a cold start has more reliable results, as shown by the tighter error band.

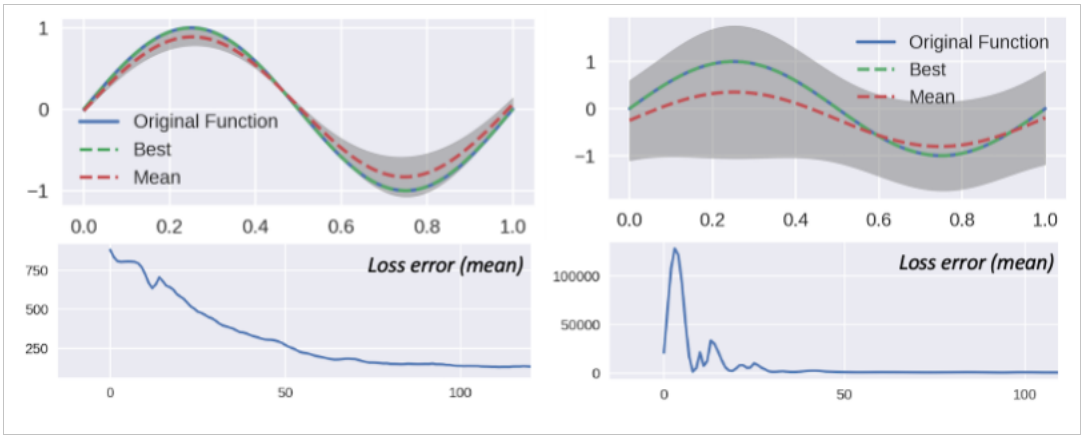

Figure 2: Training results after 300 iterations with 150 particles in the swarm ***A:*** when initialized with a cold start, and ***B:*** when initialized with a non-zero velocity.

Table 1: Comparison of the average L_2 errors using ADAM optimizer and PSO-PINN optimizer after 3,000 iterations.

Optimizer	L_2 error
ADAM *(best)*	4.331×10^{-3}
PSO-PINN *(best)*	5.894×10^{-4}
PSO-PINN *(mean)*	7.324×10^{-3}

The Poisson equation was then approximated using both the ADAM optimizer and the PSO-PINN optimizer. As can be seen in Table 1, the best result for PSO-PINN had a lower L_2 error than the best result of the ADAM optimized approximation. However, it must be noted that the average L_2 error of the PSO-PINN optimized approximation is higher.

3.2 Advection equation

Next, the advection equation was used to study how different parameters affect how the PSO-PINN approximates the equation. This equation is of particular interest because of its ubiquitous presence in fluid dynamics for modeling bulk fluid motion.

In this project, it is assumed that the advection equation is a linear univariate such that $q_t + uq_x = 0$. Here, the speed is u, which is a constant. The Riemman initial condition is stated in Eq. 12.

$$q(x,0) = \begin{cases} q_l, & 0 \leq x < x_0, \\ q_r, & x_0 < x \leq L \end{cases} \tag{12}$$

When $0 < (L - x_0)/u$, this has a solution of:

$$q(x,t) = \begin{cases} q_l, & 0 \leq x < x_0 + ut, \\ q_r, & x_0 + ut < x \leq L \end{cases} \tag{13}$$

The particle swarm optimizer parameters were kept the same as what was used in the Poisson's equation tests. The advection model was chosen to have a solution grid of $(x,t) = (1024, 256)$. The default channel length and advection speed was 2 and 1, respectively. The learning rate was chosen to be 1×10^{-2}.

First, the effect of advection speed was tested. From Fig. 3, it seems that when all other factors are kept constant, the neural network does not do as well in approximating the function. This could be improved by either increasing the number of particles in the swarm or increasing the number of

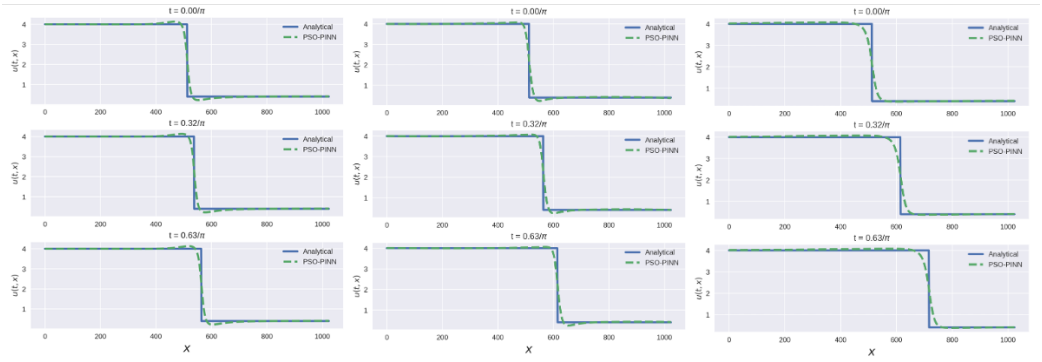

Figure 3: Training results at different times for different initial advection speeds: ***left:*** 0.5, ***middle:*** 1.0, and ***right:*** 2.0.

collocation points. For a system with a longer channel length (Fig. 4), the approximation is further improved given that all other factors are kept constant, likely due to a more stable learning that allows the particles to optimize more easily since there are less changes.

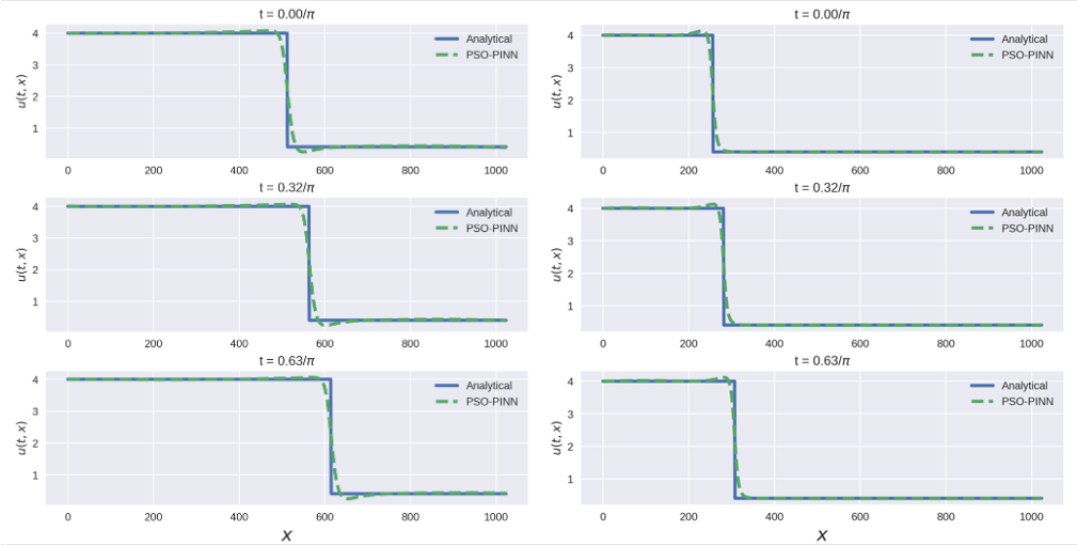

Figure 4: Approximation of the advection equation using PSO at different channel lengths: ***left:*** $L = 2$, and ***right:*** $L = 4$.

Finally, the number of particles was varied, as seen in Fig. 5. Similar to the results in the Poisson's equation tests, the model is improved when the number of particles is increased.

Table 2: Comparison of the average L_2 errors using ADAM optimizer and PSO-PINN optimizer after 3,000 iterations.

Optimizer	L_2 error
ADAM *(best)*	4.749×10^{-1}
PSO-PINN *(best)*	2.891×10^{-1}

Approximating the advection equation was then performed using the ADAM and PSO-PINN optimizer. A summary of the best results for both (same system parameters) are shown in Table 2. It was found that PSO-PINN as the optimizer had better performance than the classic ADAM optimizer.

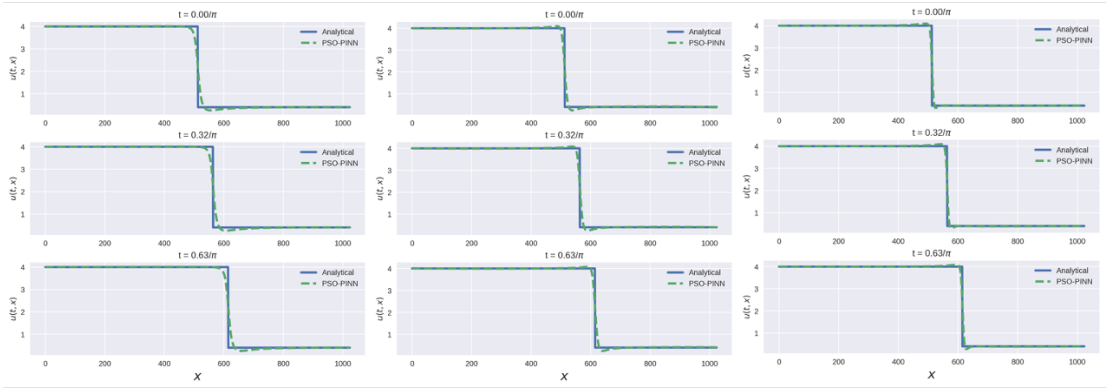

Figure 5: Approximation of the advection equation using PSO at different channel lengths: ***left:*** 50, ***middle:*** 100, and ***right:*** 200.

3.3 Burgers' equation

The viscous Burgers' equation is given below in Eq. 14.

$$\begin{aligned} u_t + uu_x - (0.01/\pi)u_{xx} = 0, x \in [-1, 1], t > 0 \\ u(0, x) = -\sin(\pi x), \\ u(t, -1) = u(t, 1) = 0 \end{aligned} \tag{14}$$

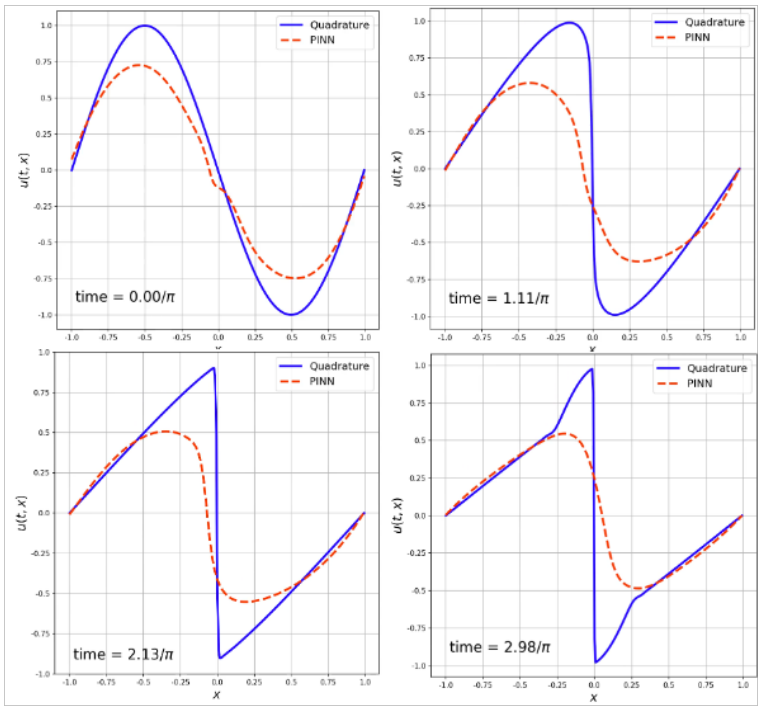

Figure 6: Approximation of the Burgers' equation when 50 particles are used.

When the number of particles is insufficient (Fig. 6), it can be seen that the neural network cannot approximate the function sufficiently. To improve this, the number of particles should be increased. It can be seen below in Fig. 7, which has a particle population of 100, that the approximation is much better and is reached faster, since optimization is more efficient with more particles.

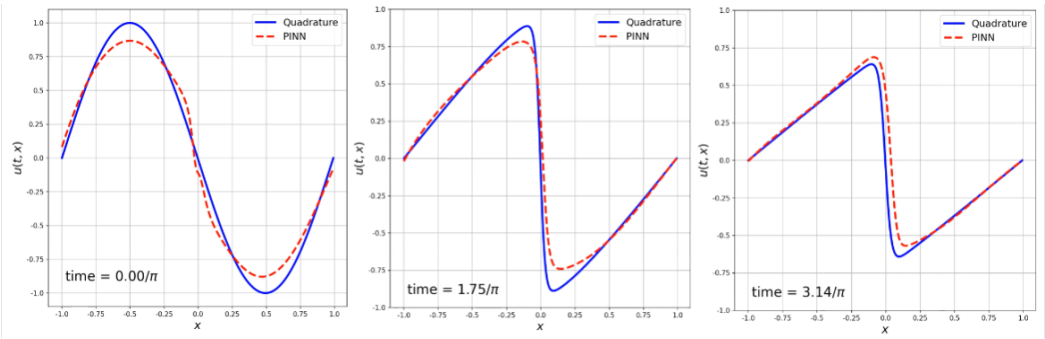

Figure 7: Approximation of the Burgers' equation when 100 particles are used.

Overall, it was observed in this project that there are a myriad of parameters that must be considered when using the PSO-PINN algorithm. It is very apparent that when optimizing the hyperparameters with accuracy in mind, the system of the function that is desired to be optimized must be considered. Bigger systems require a larger number of collocation points or particles, etc. These all translate to better approximations of the partial differential equation of the systems considered by reducing the residual. However, one must still consider using only the required number of collocation points and particles in order to reduce computational cost, since the run time exponentially increases with the number of collocation points.

4 Conclusion and Future Work

In this report, particle swarm optimization method for physics informed neural networks called PSO-PINN is explored. This is based on a particle-swarm optimation methodology for training physics informed neural networks. PSO-PINN is based on PSO-BP, a hybrid particle swarm optimization-gradient descent back-propagation algorithm for training neural networks. The PSO-PINN creates the ensemble of particles to approximate a PINNs solutions. The ensembles has the advantage of variance estimation for quantifying the uncertainty in the prediction. A better convergence and approximation of PINNs render a smaller variance and a higher degree of confidence in the predicted values. We explore the PSO-PINNs results and performance with linear and non-linear partial differential equations and shows that PSO-PINN is a promising approach to deal with failure modes of PINNs resulting from gradient pathologies and that PSO-PINN consistently outperformed the baseline PINN trained with Adam gradient descent. A larger ensemble size yields better approximation but of course at the higher computation cost. In future work, We intend to explore the improvements in PSO-PINNs methodology and tackle more complicated problems including inverse mapping of parameters, explore stochasticity. In regard with the uncertainty quantification, the drop out method for uncertainty quantification is also an intriguing method of quantifying epistemic uncertainty.

References

[1] 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.

[2] M. Raissi, A. Yazdani, and G. E. Karniadakis, "Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations," *Science*, vol. 367, no. 6481, pp. 1026–1030, 2020.

[3] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, "Physics-informed machine learning," *Nature Reviews Physics*, vol. 3, no. 6, pp. 422–440, 2021.

[4] S. Wang, Y. Teng, and P. Perdikaris, "Understanding and mitigating gradient flow pathologies in physics-informed neural networks," *SIAM Journal on Scientific Computing*, vol. 43, no. 5, pp. A3055–A3081, 2021.

[5] L. McClenny and U. Braga-Neto, "Self-adaptive physics-informed neural networks using a soft attention mechanism," *arXiv preprint arXiv:2009.04544*, 2020.

[6] R. van der Meer, C. W. Oosterlee, and A. Borovykh, "Optimally weighted loss functions for solving pdes with neural networks," *Journal of Computational and Applied Mathematics*, vol. 405, p. 113887, 2022.

[7] D. Zhang, L. Lu, L. Guo, and G. E. Karniadakis, "Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems," *Journal of Computational Physics*, vol. 397, p. 108850, 2019.

[8] A. F. Psaros, X. Meng, Z. Zou, L. Guo, and G. E. Karniadakis, "Uncertainty quantification in scientific machine learning: Methods, metrics, and comparisons," *arXiv preprint arXiv:2201.07766*, 2022.

[9] S. Wang, X. Yu, and P. Perdikaris, "When and why pinns fail to train: A neural tangent kernel perspective," *Journal of Computational Physics*, vol. 449, p. 110768, 2022.

[10] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95-international conference on neural networks*, vol. 4, pp. 1942–1948, IEEE, 1995.

[11] C. Davi and U. Braga-Neto, "Pso-pinn: Physics-informed neural networks trained with particle swarm optimization," *arXiv preprint arXiv:2202.01943*, 2022.

[12] R. Eberhart and J. Kennedy, "Particle swarm optimization," in *Proceedings of the IEEE international conference on neural networks*, vol. 4, pp. 1942–1948, Citeseer, 1995.

[13] Y. Shi and R. C. Eberhart, "Empirical study of particle swarm optimization," in *Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406)*, vol. 3, pp. 1945–1950, IEEE, 1999.

[14] D. Wang, D. Tan, and L. Liu, "Particle swarm optimization algorithm: an overview," *Soft Computing*, vol. 22, no. 2, pp. 387–408, 2018.

[15] M. Ganaie, M. Hu, *et al.*, "Ensemble deep learning: A review," *arXiv preprint arXiv:2104.02395*, 2021.