2023 IEEE 19th International Conference on Automation Science and Engineering (CASE) | 979-8-3503-2069-5/23/$31.00 ©2023 IEEE | DOI: 10.1109/CASE56687.2023.10260639

A Physics-Informed Neural Network Modeling Approach to Direct Ink Writing 3D Printing Process

Vaibhav Sharma, Rong Pan and Giulia Pedrielli

School of Computing and Augmented Intelligence, Arizona State University, Tempe, AZ 85281, U.S.A.

***Abstract*— Direct ink writing is a 3D printing process and the quality of this process depends on the steady-state flow of materials at the tip of nozzle. In this paper, we investigate a data-driven approach, the physics informed neural network, for predicting flows, and compare different neural network architectures and their performance.**

I. Introduction

Artificial neural network (ANN) can be viewed as a universal function approximator. The solutions of ordinary differential equations (ODEs) or partial differential equations (PDEs) are often very complicated functions while closed-form solutions are rarely available. In recent years, using ANN to approximate ODE/PDE solution has been actively researched. The ANNs constructed for this purpose are called physics-informed neural networks (PINNs). These PINN models are both physics-based and data-driven, because the loss functions used in these models incorporate the prediction deviations from PDE and boundary condition functions, as well as the losses from data prediction errors as in conventional neural networks. PINNs provide a faster alternative to other numerical solutions of ODE/PDE for complex systems. In addition, it is able to investigate the inverse problem, where the parameters existed in differential equations can be estimated by using an almost identical PINN architecture. This data-driven approach to understanding the underlying physics of a dynamic system is highly desired by a broad range of applications in science and engineering.

The direct ink writing (DIW) manufacturing process is a versatile, cost-effective, 3D printing technique. It uses semi-fluid mixtures as inks, extrudes and layouts inks through a nozzle on the substrate line-by-line. The quality of DIW products are determined by the ink printability, which is controlled by printing process parameters. It is desirable to predict the velocity field of extrusion and the nozzle pressure so as to better control the manufacturing process. The method of computational fluid dynamic (CFD) is a powerful tool to elucidate the physical phenomena during the shear-thinning extrusion process [7]. However, this computational method is mesh-based and often too slow to be used for online process control. Furthermore, it requires pre-defined material parameters which are often unknown apriori. Therefore, in recent years, PINNs have been trained as alternatives to CFD models and their predictions have been shown to have comparable accuracy to CFD but with much faster computation [1], [5], [9], [18].

In this paper, we first introduce the generic architecture of PINN and demonstrate the connection of data-based and physics-based approaches to system dynamics. Next, we present several modified ANN architectures that have been proposed for improving PINN performance. We apply them on the material extrusion process of DIW and show that our PINN models can predict the nozzle pressure and the fluid flow velocity well, thus they have the potential for real-time process control, becoming building blocks for the cyber coordinated additive manufacturing system [17].

II. Previous Work

A. PINNs

Physical and engineering problems, such as fluid flow, heat transfer, or electric circuit design are often modeled with ODEs and PDEs as the physical phenomenon's governing equations. Solving these differential equations is not easy. Conventional numerical methods include finite element method (FEM) and finite difference method (FDM), which are mesh-based and computationally intensive. Unlike FEM or FDM, the PINN approach is mostly data-driven and mesh-free, thus suitable for the

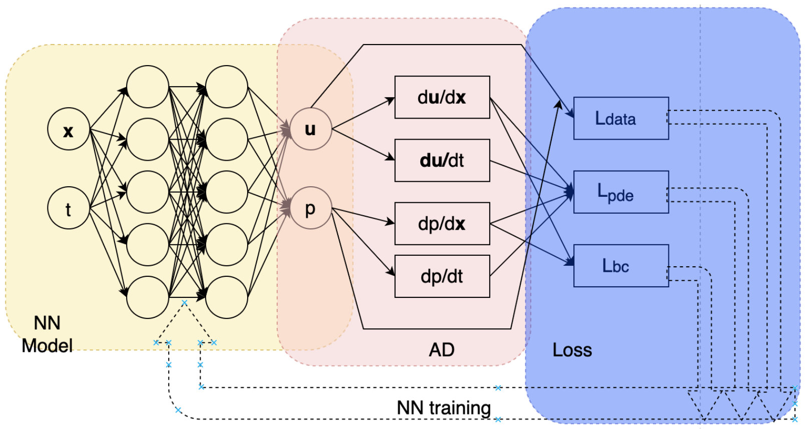


Fig. 1. Illustration of a generic PINN model. There are three components – neural network, automatic differentiation, and loss functions for data, PDEs and boundary conditions.

case where scattered partial spatio-temporal data are available and the initial condition or boundary condition may be unknown.

A generic PINN model is a fully-connected feed-forward ANN, which is composed of multiple hidden layers, is used to approximated the solution of the PDE by take the space and time coordinates as inputs. It takes advantage of automatic differentiation (AD) of computational neural network and uses this function to construct the PDE loss that is a component of the total loss function for neural network training.

Taking the Navier-Strokes equation of fluid dynamics as an example, the parameterized PDE system can be expressed as

$$f(x, t, \hat{u}, \partial_x \hat{u}, \partial_t \hat{u}, \theta) = 0,\ x \in \Omega,\ t \in [0, T],$$

with $\hat{u}(x, t_0) = g_0(x)$ for $x \in \Omega$, and $\hat{u}(x, t) = g_B(t)$ for $x \in \partial\Omega$. All parameters in the system are represented by θ. Here, function f denotes the residual of the PDE, containing the differential operators and PDE parameters, $\hat{u}(x, t)$ is the solution of PDE with the initial condition $g_0(x)$ and the boundary condition $g_B(t)$. Its corresponding PINN model is illustrated in Figure 1. One can see that the overall loss function consists of three components – the loss for model prediction errors, the loss that penalizes the residual of the governing equation, and the loss for unsatisfying the boundary condition and/or initial condition.

B. *Relevant literature*

The origin of PINN can be traced to Lagaris et al. [12], in which the solution to differential equations was framed as a minimization problem using the sum of two sub-functions, while one of these functions was chosen such that it satisfies the boundary conditions and the other one was approximated by using a neural network having the number of input layers equal to the number of independent variables, one hidden layer, and one output layer having a number of units equal to the number of dependent variables. Raissi et al. [14], [15], [16] took this idea and coined the model, physics-informed neural network or PINN, as a deep learning framework to solve nonlinear partial differential equations. They proposed that, by regularizing the loss function by the PDEs governing the physics of the system, better neural network models can be developed for engineering and biological systems with limited data. It was shown that PINNs can solve two classes of problems – firstly, solving the PDEs governing the system given the limited data within the domain of the system; and secondly, estimating the parameters involved in the PDEs. Comparing with a finite elements solver, they observed the accuracy of the PINN solution to be within 99%, even with 1% Gaussian uncorrelated noise, in the first class of problem and the accuracy was around 95% for parameter estimation in the second class of problem.

The PINN approach has also been used in reinforcement learning, where active control of flow dynamics is made possible [5]. For simple linear dynamic systems, a rapid version of PINNs has been developed by taking advantage of stationary and time-dependent linear partial differential equations [6]. A special neural network architecture called NSFnets was proposed for solving

Navier-Stokes equations for simulating incompressible laminar and turbulent flows [9]. Using PINNs to solve inverse problems for metamaterial design was discussed in Chen et al. [2]. Cuomo et al. [4] and Karniadakis et al. [10] provide the latest summary of the literature on PINNs, including a discussion of their advantages and disadvantages, and variants of PINNs adapted for different applications. The paper by Cai et al. [1] provides a comprehensive review of physics-informed neural networks (PINNs) as a solver for fluid dynamics problems, with a particular focus on their efficacy in simulating dynamic fluid flows.

While this approach is a good way to combine the data and the physics governing the system, we can improve its convergence rate by selecting a suitable combination of initialization, architecture, and optimization method. One improvement of the vanilla PINN architecture comes from the use of ResNet. To counter the degradation problem from deeper neural networks, He et al. [8] introduced the deep residual neural network, or ResNet. In this architecture, they introduced the residual or skip connections or identity mapping in between groups of layers, this allows the network to choose either the identity mapping or the combination weights. Cheng and Zhang [3] solve the Navier-Stokes equation by using PINN paired with a Resnet block. The ResNet block is used to improve the stability of the neural network.

Rahaman et al. [13] demonstrated that the neural network prioritizes learning smaller frequencies over local fluctuations and called this phenomenon the spectral bias of the neural network. They demonstrated that lower frequency noise affects the system more than that of higher frequency; therefore, they argued that a given signal defined on the manifold is easier to fit if the coordinate system that corresponds to the manifold can be expressed in terms of higher frequency components. This idea of spectral adjusted neural network has not been attempted on PINNs yet.

In PINNs there are at least two loss functions being formulated - one function evaluates the NN solutions using training data and the other function evaluates the gradient of NN against the analytical differential function. Yu [19] and Kumar et al. [11] described that multitask learning is much more efficient than learning each task independently, but there are several challenges that prevent significant efficiency gains. The main challenges are: (a) conflicting gradient of individual tasks because parameter improvements with respect to one task may degrade it with respect to other tasks, (b) large differences in the magnitudes of gradients of individual tasks leading to some task gradients dominate over the other in the gradient of overall loss, and (c) high curvature in the multitask optimization landscape. They proposed a model independent algorithm, Projecting Conflicting Gradients (PCGrad), to alleviate these issues. In this algorithm, they adopted a simple procedure to reduce the conflict between the gradients during optimization by projecting the gradient of each task onto the normal plane of the gradient of other tasks if the gradients of two tasks are in conflict,i.e., if their cosine similarity is negative. A substantial improvement in the performance had been observed, including over 30% improvement in multi-task reinforcement learning problems compared to other approaches. Again, the idea of PCGrad has not been applied on PINNs yet.

III. Architectures of PINNs for DIW

A. *DIW fluid dynamics simulation*

Guo et al. [7] conducted a comprehensive simulation study on the flow of non-Newtonian fluid ink in the DIW process. Their study indicated that a nozzle designed with a cylindrical shape exhibits stable fluid velocity and shear stress profiles in the nozzle region. However, this shape can also generate very high stresses in the nozzle, causing severe material degradation. Thus, they recommended a design with conical body and optimally sized cylindrical nozzle tip for DIW. Their recommended nozzle shape is adopted in our study.

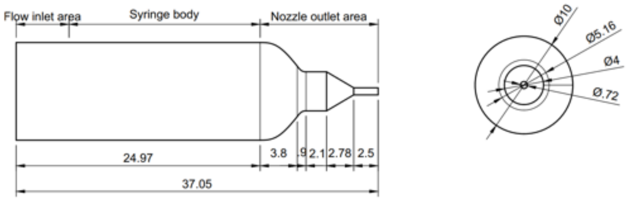


Fig. 2. Nozzle shape for DIW process and geometry specifications.

Utilizing the symmetry of the nozzle, a 2D axis-symmetric planar surface has been considered for the nozzle meshing. ANSYS Fluent is used to generate structured meshing, with the inclusion of five inflation layers for capturing the boundary layer in proximity to solid surfaces. In this study, the flow process of non-Newtonian DIW ink has been modeled as a steady-state viscous laminar fluid flow problem. To model the viscosity of the non-Newtonian shear-thinning fluid, the Herschel-Bulkley viscosity model has been incorporated into the software as a user-defined function (UDF). This approach enables the simulation of complex rheological behavior of the ink, and facilitates the calculation of relevant flow parameters.The Herschel-Bulkley model is expressed as follows:

$$\tau = \tau_{yield} + K\dot{\gamma}^n \tag{1}$$

where τ is the shear stress, τ_{yield} is the yield stress, K is the consistency index, $\dot{\gamma}$ is the shear rate, and n is the flow behavior index. From the above equation, the dynamic viscosity (μ) can be calculated as:

$$\mu = \frac{\tau}{\dot{\gamma}} = \frac{\tau_{yield}}{\dot{\gamma}} + K\dot{\gamma}^{n-1} \tag{2}$$

where $\dot{\gamma}$ is the shear rate magnitude.

The governing equations for a viscous laminar fluid flow problem are the continuity equation and the Navier-Stokes equation. The continuity equation represents the conservation of mass in the fluid flow, requiring that the rate of mass entering a control volume is equal to the rate of mass leaving it, and any net accumulation of mass inside the control volume is equal to the difference between the inflow and outflow rates. Mathematically, it is expressed as

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \tag{3}$$

where ρ is the fluid density, t is time, $\mathbf{u}$ is the velocity vector, and $\nabla \cdot (\rho \mathbf{u})$ represents the divergence of the mass flux.

For a 2D steady-state viscous laminar fluid flow, this continuity equation can be simplified to

$$\frac{\partial(\rho u)}{\partial x} + \frac{\partial(\rho v)}{\partial y} = 0 \tag{4}$$

where u and v are the velocities in the x and y directions, respectively. It serves as a crucial constraint for the solution of the Navier-Stokes equations and it plays a significant role in the calculation of important flow parameters such as pressure, velocity, and turbulence. The continuity equation can be automatically satisfied by representing velocities in terms of a scalar flow field, known as the stream function, which is constant along streamlines. Mathematically, this can be expressed as

$$u = \frac{\partial \psi}{\partial y}, \qquad v = -\frac{\partial \psi}{\partial x} \tag{5}$$

where ψ is the stream function.

The Navier-Stokes equation is a fundamental equation governing the motion of viscous fluids. It expresses the conservation of momentum and provides a mathematical framework for modeling and simulating fluid flow. Mathematically, this can be expressed as

$$\rho \frac{\partial \mathbf{u}}{\partial t} + \rho(\mathbf{u} \cdot \nabla)\mathbf{u} = -\nabla p + \mu \nabla^2 \mathbf{u} \tag{6}$$

where μ is the dynamic viscosity of the fluid.

The Navier-Stokes equation for 2D steady-state case can be simplified as

$$\rho u \frac{\partial u}{\partial x} + \rho v \frac{\partial u}{\partial y} = -\frac{\partial p}{\partial x} + \mu \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) \tag{7}$$

along x axis. And,

$$\rho u \frac{\partial v}{\partial x} + \rho v \frac{\partial v}{\partial y} = -\frac{\partial p}{\partial y} + \mu \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right) \tag{8}$$

along y axis.

The training data for this study is generated by ANSYS Fluent solver after providing it with the parameters and boundary conditions listed in Tables I and II

TABLE I. Fluid Properties

Property	Value
Density (ρ)	1240 kg/m^3
Consistency coefficient (k)	568.6 Pa-s
Flow index (n)	0.335
Yield stress (τ_{yield})	764.01 Pa

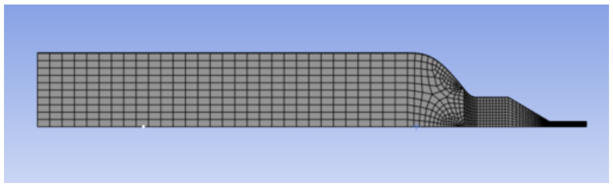
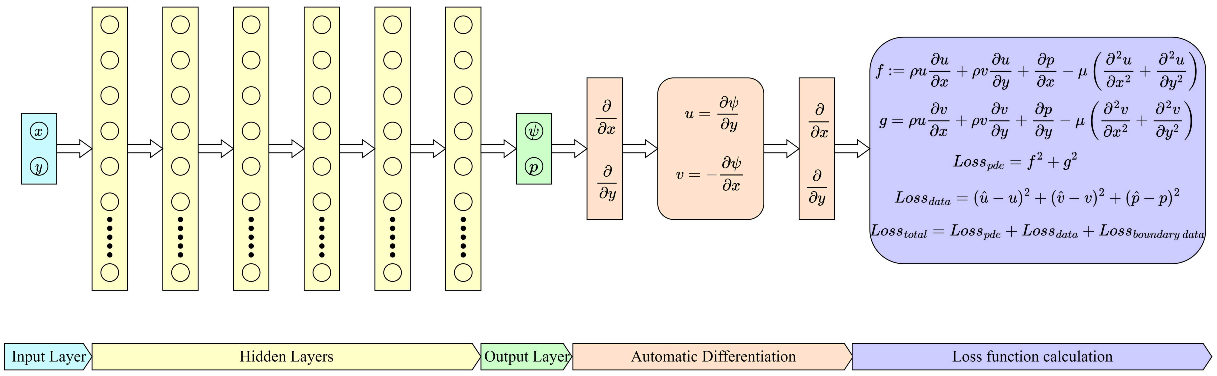


Fig. 3. Ansys Fluent mesh model for the nozzle.

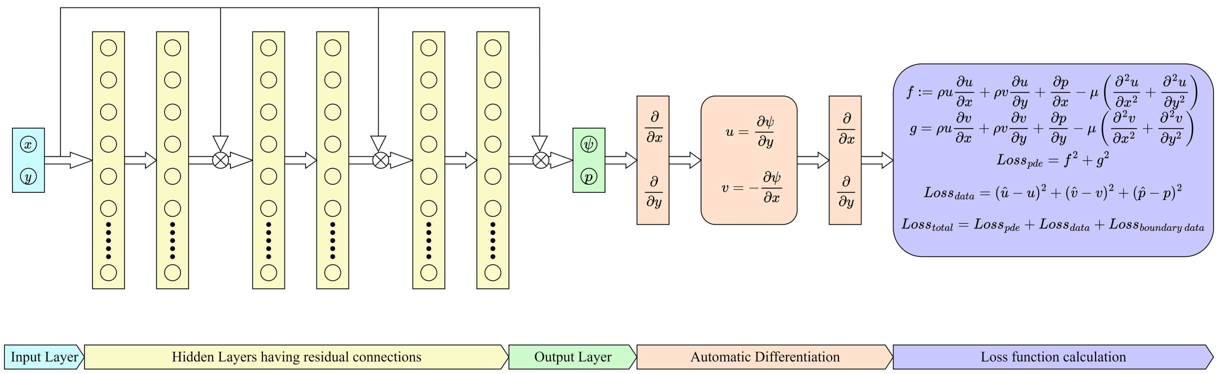
B. *PINN models*

The initial PINN architecture used in our experiment is a sequential feed-forward neural network, where randomly chosen mesh location coordinates are used as



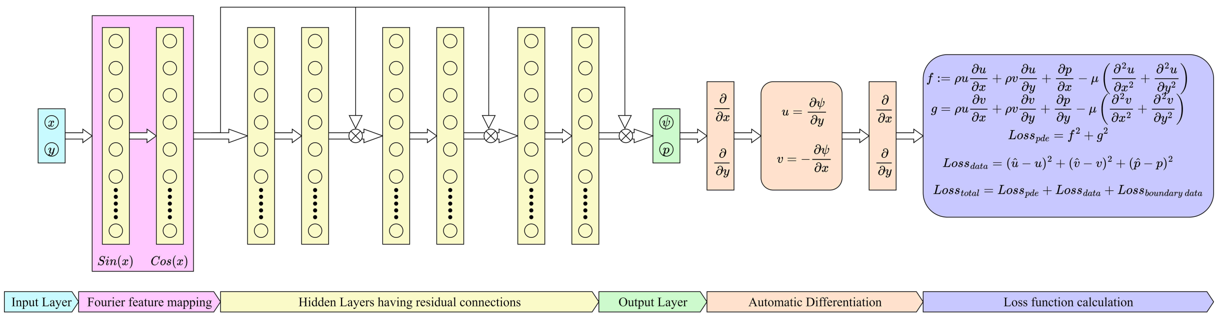
x
y
ψ
p
$\frac{\partial}{\partial x}$
$\frac{\partial}{\partial y}$
$u = \frac{\partial \psi}{\partial y}$
$v = -\frac{\partial \psi}{\partial x}$
$f := \rho u \frac{\partial u}{\partial x} + \rho v \frac{\partial u}{\partial y} + \frac{\partial p}{\partial x} - \mu \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right)$
$g = \rho u \frac{\partial v}{\partial x} + \rho v \frac{\partial v}{\partial y} + \frac{\partial p}{\partial y} - \mu \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right)$
$Loss_{pde} = f^2 + g^2$
$Loss_{data} = (\hat{u} - u)^2 + (\hat{v} - v)^2 + (\hat{p} - p)^2$
$Loss_{total} = Loss_{pde} + Loss_{data} + Loss_{boundary\ data}$
Input Layer
Hidden Layers
Output Layer
Automatic Differentiation
Loss function calculation

Fig. 4. Sequential Neural Network Architecture.



x
y
ψ
p
$\frac{\partial}{\partial x}$
$\frac{\partial}{\partial y}$
$u = \frac{\partial \psi}{\partial y}$
$v = -\frac{\partial \psi}{\partial x}$
$f := \rho u \frac{\partial u}{\partial x} + \rho v \frac{\partial u}{\partial y} + \frac{\partial p}{\partial x} - \mu \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right)$
$g = \rho u \frac{\partial v}{\partial x} + \rho v \frac{\partial v}{\partial y} + \frac{\partial p}{\partial y} - \mu \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right)$
$Loss_{pde} = f^2 + g^2$
$Loss_{data} = (\hat{u} - u)^2 + (\hat{v} - v)^2 + (\hat{p} - p)^2$
$Loss_{total} = Loss_{pde} + Loss_{data} + Loss_{boundary\ data}$
Input Layer
Hidden Layers having residual connections
Output Layer
Automatic Differentiation
Loss function calculation

Fig. 5. Residuals Neural Network Architecture.



x
y
$Sin(x)$
$Cos(x)$
ψ
p
$\frac{\partial}{\partial x}$
$\frac{\partial}{\partial y}$
$u = \frac{\partial \psi}{\partial y}$
$v = -\frac{\partial \psi}{\partial x}$
$f := \rho u \frac{\partial u}{\partial x} + \rho v \frac{\partial u}{\partial y} + \frac{\partial p}{\partial x} - \mu \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right)$
$g = \rho u \frac{\partial v}{\partial x} + \rho v \frac{\partial v}{\partial y} + \frac{\partial p}{\partial y} - \mu \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right)$
$Loss_{pde} = f^2 + g^2$
$Loss_{data} = (\hat{u} - u)^2 + (\hat{v} - v)^2 + (\hat{p} - p)^2$
$Loss_{total} = Loss_{pde} + Loss_{data} + Loss_{boundary\ data}$
Input Layer
Fourier feature mapping
Hidden Layers having residual connections
Output Layer
Automatic Differentiation
Loss function calculation

Fig. 6. Fourier Residuals Neural Network Architecture.

TABLE II. Boundary Conditions

BC	Value
Inlet	Pressure = 68750 Pa
Outlet	Vent (Pressure multiplier = 1)
Wall	Stationary wall, no slip

the inputs and 2-D velocities and pressures are the outputs. The training data are sampled from ANSYS simulations. The automatic differentiation functions of neural network are utilized to build the Navier-Stokes equations on the x and y axes, which leads to the pde loss function. The data loss function is constructed by combining the squared errors from velocity and pressure predictions. Figure 4 illustrates this architecture.

In addition to the sequential neural network, we also experimented with the ResNet [8], ResNet with Fourier feature mapping [13], and ResNet with PCgrad [19] architectures. Figures 5 and 6 illustrate the first two of these architectures, and the last one is not shown here because it is the same architecture with of ResNet with Fourier features but with adjustments of the gradient functions for the two loss functions during the neural network training process.

To simplify the training, we focused on the fluid flow in the rectangular nozzle outlet region, where the solutions of Navier-Stokes equations were approximated by PINNs. We compared the solution convergence from four architectures of both the forward and inverse analyses. The forward analysis is about making predictions of velocities along x and y axes and predictions of pressure at any location, while the inverse analysis provides the estimations of parameters used in the Navier-Strokes PDEs.

IV. Results and Discussions

Figures 7 and 8 depict the training processes of 4 neural networks in terms of reducing the errors in velocity prediction, and Figure 9 for the errors in pressure prediction. One can see that the sequential neural network is not stabilized for velocity prediction even after a long training epochs. ResNet with Fourier features and ResNet with PCgrad provides superior performance in terms of faster convergence and lower error values they can achieve. In particular, the PCgrad approach can better quickly stabilize the prediction error, no matter for velocity or pressure.

For the inverse analysis, the advantage of the PCgrad approach becomes even more prominent. Figures 10 and 11 show that the sequential neural network and ResNet architectures cannot reach the convergence of parameter estimation after a long training period, while with Fourier features and PCgrad the estimation converges. However, a bias in parameter estimation is also observed. The reason of this bias needs to be further investigated.

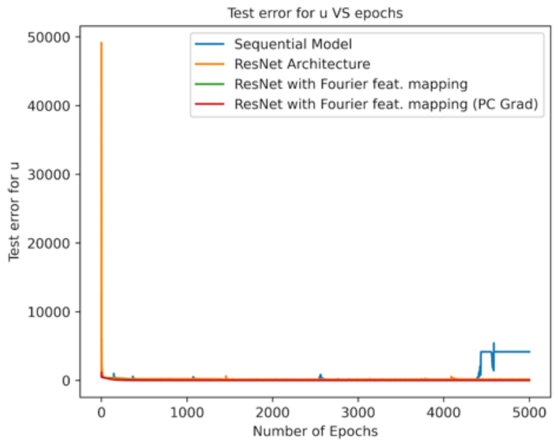


Fig. 7. The convergence of test error of velocity u from forward analysis.

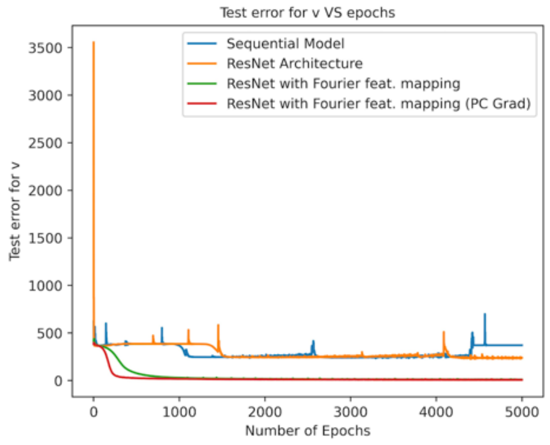


Fig. 8. The convergence of test error of velocity u from forward analysis.

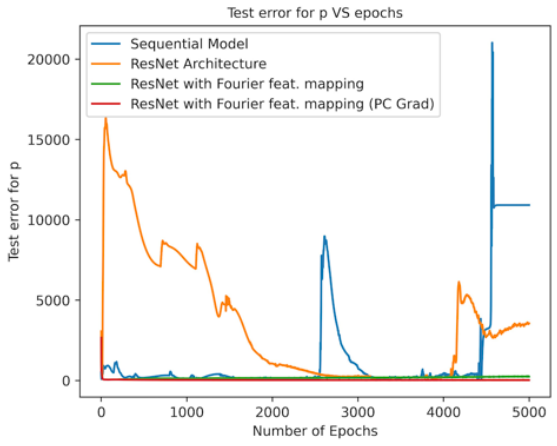


Fig. 9. The convergence of test error of velocity u from forward analysis.

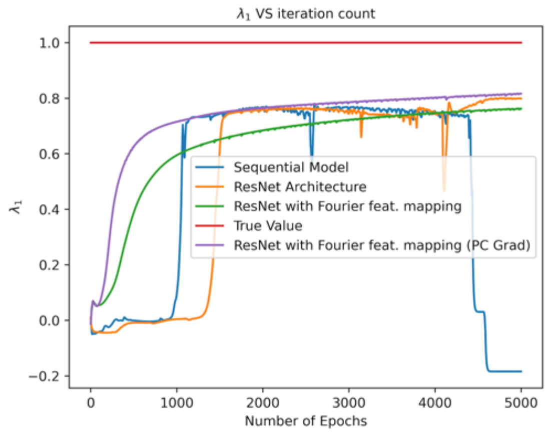


Fig. 10. Inverse Analysis of PDE parameter 1.

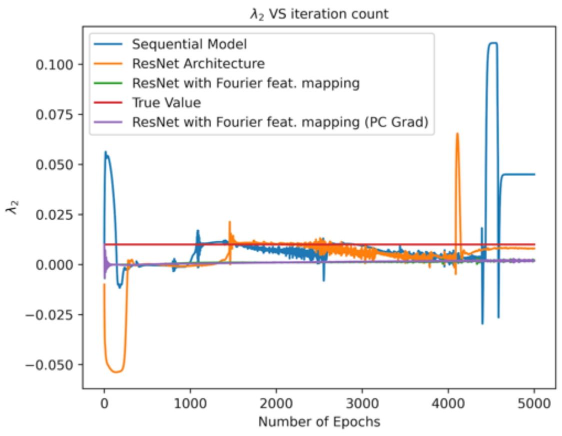


Fig. 11. Inverse Analysis of PDE parameter 2.

V. Conclusions

In this paper we investigated four PINN architectures for approximating the solutions of DIW fluid flow in the nozzle region, which is defined by the Navier-Stokes equations. Our results showed that adding Fourier features and PCgrad approach to resolve the gradient search conflict in PINNs is an effective approach. This approach has not been explored much in literature. Our results also demonstrate the feasibility of PINNs for solving the PDEs that govern the DIW process, thus paving the way for building a real-time digital twin for predicting DIW process quality. It is noted that a specific application of DIW may have many other process variables that need monitoring, diagnosis and prognosis, beside of velocity and pressure. Adapting these PINN models to a broader range of DIW processes requires a future research.

Acknowledgement

This work was partially supported by National Science Foundation (Grant No. 2134409).

References

[1] S. Cai, Z. Mao, Z. Wang, M. Yin, and G. E. Karniadakis. Physics-informed neural networks (pinns) for fluid mechanics: A review. *Acta Mechanica Sinica*, 37(12):1727–1738, 2021.

[2] Y. Chen, L. Lu, G. E. Karniadakis, and L. Dal Negro. Physics-informed neural networks for inverse problems in nano-optics and metamaterials. *Optics express*, 28(8):11618–11633, 2020.

[3] C. Cheng and G.-T. Zhang. Deep learning method based on physics informed neural network with resnet block for solving fluid flow problems. *Water*, 13(4):423, 2021.

[4] S. Cuomo, V. S. Di Cola, F. Giampaolo, G. Rozza, M. Raissi, and F. Piccialli. Scientific machine learning through physics–informed neural networks: where we are and what's next. *Journal of Scientific Computing*, 92(3):88, 2022.

[5] Z. Dang and M. Ishii. A physics-informed reinforcement learning approach for the interfacial area transport in two-phase flow. *arXiv preprint arXiv:1908.02750*, 2019.

[6] V. Dwivedi and B. Srinivasan. Physics informed extreme learning machine (pielm)–a rapid method for the numerical solution of partial differential equations. *Neurocomputing*, 391:96–118, 2020.

[7] Z. Guo, F. Fei, X. Song, and C. Zhou. Analytical study of shear-thinning fluid flow in direct ink writing process. In *International Manufacturing Science and Engineering Conference*, volume 85802, page V001T01A034. American Society of Mechanical Engineers, 2022.

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[9] X. Jin, S. Cai, H. Li, and G. E. Karniadakis. Nsfnets (navier-stokes flow nets): Physics-informed neural networks for the incompressible navier-stokes equations. *Journal of Computational Physics*, 426:109951, 2021.

[10] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

[11] A. Kumar and H. Daume III. Learning task grouping and overlap in multi-task learning. *arXiv preprint arXiv:1206.6417*, 2012.

[12] I. E. Lagaris, A. Likas, and D. I. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE transactions on neural networks*, 9(5):987–1000, 1998.

[13] N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. Hamprecht, Y. Bengio, and A. Courville. On the spectral bias of neural networks. In *International Conference on Machine Learning*, pages 5301–5310. PMLR, 2019.

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

[15] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics informed deep learning (part ii): Data-driven discovery of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10566*, 2017.

[16] 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*, 378:686–707, 2019.

[17] H. Sun, G. Pedrielli, G. Zhao, C. Zhou, W. Xu, and R. Pan. Cyber coordinated simulation for distributed multi-stage additive manufacturing systems. *Journal of manufacturing systems*, 57:61–71, 2020.

[18] S. Wang, H. Wang, and P. Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed deeponets. *Science advances*, 7(40):eabi8605, 2021.

[19] T. Yu, S. Kumar, A. Gupta, S. Levine, K. Hausman, and C. Finn. Gradient surgery for multi-task learning. *Advances in Neural Information Processing Systems*, 33:5824–5836, 2020.