Korean J. Chem. Eng., 40(9), 2119-2127 (2023)
DOI: 10.1007/s11814-023-1420-4

pISSN: 0256-1115
eISSN: 1975-7220

Physics-informed neural networks for learning fluid flows with symmetry

Younghyeon Kim‡, Hyungyeol Kwak‡, and Jaewook Nam†

School of Chemical and Biological Engineering, Institute of Chemical Processes,
Seoul National University, Seoul 08826, Korea
(*Received 4 November 2022 • Revised 3 March 2023 • Accepted 11 March 2023*)

Abstract−We suggest symmetric variational physics-informed neural networks (symmetric VPINN) to learn the symmetric fluid flow and physical properties of fluids from a limited set of data. Symmetric VPINN is based on the VPINN framework and guarantees the symmetry of the solutions by modifying the network architecture. The effectiveness of the symmetric VPINN is demonstrated by predicting the velocity profiles and power-law fluid properties in the Poiseuille flow of a parallel channel. Symmetric VPINN models robustly and accurately learn power-law fluid flow in both forward and inverse problems. We demonstrate that the symmetric VPINN can be particularly useful when the power-law index is small and the data are extremely limited. The modified network architecture in the symmetric VPINN guides the neural network towards an exact solution by reinforcing symmetry. We show that symmetric VPINN is effective in obtaining unknown physical properties in practical experiments where data are scarce, suggesting the possibility of introducing known conditions of the system directly into the network structure to improve the accuracy of the network.

Keywords: Physics-informed Machine Learning, Network Architecture, Power-law Fluid, Pressure-driven Flow, Inverse Problem

INTRODUCTION

Machine learning based on neural networks has become a popular method that has been applied to various problems, such as solving differential equations [1-3], identifying and predicting dynamic systems [4-6], and predicting material properties [7-9]. Because neural networks are trained solely from the given data, they may violate the underlying physics. In this regard, many solvers have been suggested to involve the known physics in neural networks.

In particular, physics-informed machine learning using physics-informed neural networks (PINNs) has emerged as a popular method in scientific computing, which uses the physics of systems by including governing differential equations in the loss function of the neural network and using collocation methods. The methods using PINN have been used to solve a wide range of problems, such as the Navier-Stokes equation [10], Buckley-Leverett equation [11], and heat-transfer problems [12]. Several networks were developed based on the PINN framework. For example, parareal PINN (PPINN) [13], conservative PINN (cPINN) [14], and extended PINN (XPINN) [15] improve flexibility in handling problems by domain decomposition in time, space, and time-space, respectively. To enhance the predictivity of the model, variational physics-informed neural network (VPINN) based on the sub-domain Petrov-Galerkin method was suggested [16]. The VPINN introduces a variational residual derived from the weak form of differential equations. In the past, VPINN was used to solve the advection-diffusion equation [16] and the two-phase transport problem [17]. Additionally, several methods that force known conditions of the system to neural networks were proposed. For example, Mattheakis et al. [18] used a specialized network architecture to guarantee the symmetry of the neural network outputs.

Predicting unknown physical properties from experiments is an important issue in engineering because certain properties are difficult to obtain directly and require the solution of inverse problems. In the past, a series of studies had to be conducted to learn properties from given data using neural networks [8,9]. For example, Reyes et al. [7] applied the PINN algorithm to predict the viscosity profiles from computationally generated velocity data. However, according to Kim et al. [19], a sufficient amount of well-distributed velocity data is difficult to obtain in real-world experiments. In many experiments, data are scarce, limiting the applicability of data-driven discoveries in scientific studies.

Along this path, we propose an effective method for data-driven discovery by testing a combination of existing physics-embedded methodologies. In particular, we explore the efficacy of combining the VPINN with a specialized network architecture to solve a fluid flow problem with geometric symmetry. We apply them to the Poiseuille flow of a power-law fluid, which is a nonlinear system but can be analytically solved to evaluate the performance of various combinations.

In this study, we first formulated a given flow problem within the framework of the VPINN. We also demonstrated how the specialized network design suggested in a previous study is combined with VPINN to guarantee symmetry in the fluid flow. The proposed framework was applied to both forward and inverse flow problems to solve unknown velocity profiles or fluid properties. We also tested various combinations of PINN/VPINN and network archi-

†To whom correspondence should be addressed.
E-mail: jaewooknam@snu.ac.kr
‡These authors contributed equally to this work.

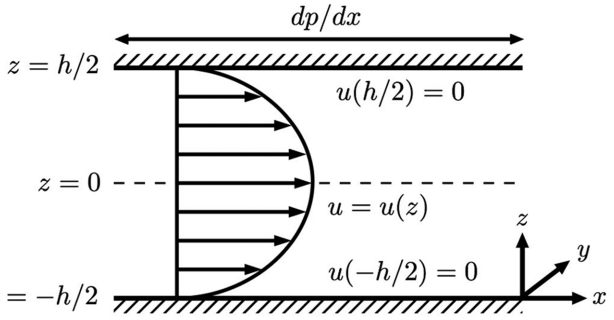


Fig. 1. Configuration of poiseuille flow in parallel channel considered in this study.

tectures to perform a comparative analysis on their performance. The reason for the superior performance of the proposed combination of VPINN and the specialized network architecture is also briefly discussed.

PINN AND VPINN FOR POISEUILLE FLOW OF POWER-LAW FLUID

To examine the performance of PINN and VPINN, we considered the Poiseuille flow of power-law fluid between two infinite parallel planes, as shown in Fig. 1. We assumed a one-dimensional steady flow of an incompressible fluid where x is the flow direction and z is the traverse direction normal to the flow. Under these assumptions, the fluid velocity u is only dependent on z and the system is governed by

$$\frac{\partial \tau}{\partial z}=\frac{\partial p}{\partial x}, \tag{1}$$

where τ is the shear stress, p is the pressure independent of z, and the pressure gradient dp/dx is a constant.

The constitutive equation for power-law fluid that follows the Ostwald-de Waele power-law model is as follows:

$$\tau=\eta\frac{du}{dz}, \tag{2}$$

where apparent viscosity η is given by

$$\eta=m\left|\frac{du}{dz}\right|^{n-1} \tag{3}$$

where m is the power-law consistency coefficient and n is the power-law index.

The governing differential equation is derived from Eqs. (1), (2), and (3) as follows:

$$\frac{d\tau}{dz}=mn\frac{d^2u}{dz^2}\left|\frac{du}{dz}\right|^{n-1}=\frac{dp}{dx}, \tag{4}$$

with non-slip boundary conditions:

$$u\left(z=\pm\frac{h}{2}\right)=0. \tag{5}$$

We obtain the analytical solution of Eq. (4) as

$$u=\frac{n}{n+1}\left(-\frac{dp}{dx}\frac{1}{m}\right)^{\frac{1}{n}}\left\{\left(\frac{h}{2}\right)^{\frac{n+1}{n}}-z^{\frac{n+1}{n}}\right\} \tag{6}$$

Despite its nonlinearity, the Poiseuille flow of a power-law fluid in a parallel channel has an analytical velocity solution, as mentioned above. We used this problem to observe the performance of PINN-based methods in solving complicated or nonlinear systems.

Below, we extend the PINN and VPINN frameworks to obtain the velocity profile (forward problem) or power-law consistency coefficient (m) and the power-law index (n) from the velocity data (inverse problem). We used a deep feedforward network as the network architecture of both PINN and VPINN, and derived their loss functions from the governing differential equation. We constructed a deep neural network with z as the input and velocity profile u as the output. Thus the proposed neural network is a mapping u_{NN}: $\Omega\rightarrow\mathbb{R}$, where Ω=[−h/2, h/2].

Based on the PINN and VPINN frameworks, we obtained the loss functions from residual r(z):

$$r(z)=\frac{d\tau_{NN}(z)}{dz}-\frac{dp}{dx}=mn\frac{d^2u_{NN}}{d(z_f^i)^2}\left|\frac{du_{NN}}{dz_f^i}\right|^{n-1}-\frac{dp}{dx}, \tag{7}$$

which is derived from Eq. (4). The loss function of PINN consists of boundary loss ($\mathcal{L}_b$) and residual loss ($\mathcal{L}_f$):

$$\mathcal{L}_{PINN}=\mathcal{L}_f+\tau_b\mathcal{L}_b, \tag{8}$$

$$\mathcal{L}_f=\frac{1}{N_f}\Sigma_{i=1}^{N_f}\left|r(z_f^i)\right|^2=\frac{1}{N_f}\Sigma_{i=1}^{N_f}\left|mn\frac{d^2u_{NN}}{d(z_f^i)^2}\left|\frac{du_{NN}}{dz_f^i}\right|^{n-1}-\frac{dp}{dx}\right|^2, \tag{9}$$

$$\mathcal{L}_b=\frac{1}{2}\left(\left|u_{NN}\left(-\frac{h}{2}\right)\right|^2+\left|u_{NN}\left(\frac{h}{2}\right)\right|^2\right). \tag{10}$$

We calculated $\mathcal{L}_f$ as the mean squared error of N_f collocation points. τ_b is the weight that adjusts the contributions of $\mathcal{L}_f$ and $\mathcal{L}_b$ and we used τ_b=1 in this study following the previous study [16].

However, variational loss ($\mathcal{L}_v$) derived from the weak form of differential equations is used in the loss function of VPINN instead of $\mathcal{L}_f$. The variational loss $\mathcal{L}_v$ may have several forms owing to integration by parts. In this system, we obtained two forms of variational residuals, resulting in two types of VPINN loss functions:

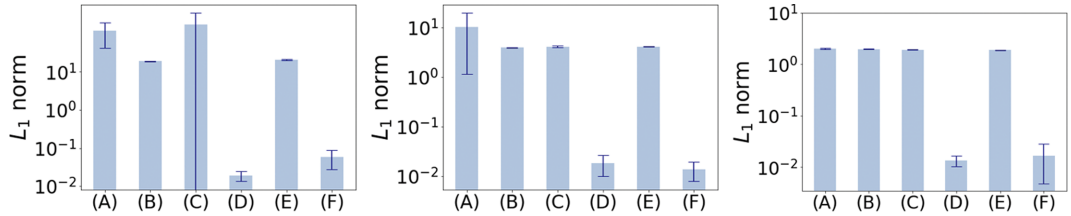
$$\mathcal{L}_{VPINN}^{(i)}=\mathcal{L}_v^{(i)}+\tau_b\mathcal{L}_b, \tag{11}$$

$$\mathcal{L}_v^{(i)}=\Sigma_{e=1}^{N_{el}}\frac{1}{K^{(e)}}\Sigma_{k=1}^{K^{(e)}}\left|{}^{(i)}\mathcal{R}_k^{(e)}\right|^2, \tag{12}$$

$${}^{(i)}\mathcal{R}_k^{(e)}=\int_{z_{e-1}}^{z_e}r(z)v_k^{(e)}(z)dz, \tag{13}$$

$$\begin{aligned}{}^{(1)}\mathcal{R}_k^{(e)}&=\int_{z_{e-1}}^{z_e}r(z)v_k^{(e)}(z)dz=\int_{z_{e-1}}^{z_e}\frac{d\tau_{NN}(z)}{dz}v_k^{(e)}(z)dz-\int_{z_{e-1}}^{z_e}\frac{dp}{dx}v_k^{(e)}(z)dz\\&=\int_{z_{e-1}}^{z_e}mn\left|\frac{du_{NN}(z)}{dz}\right|^{n-1}\frac{d^2u_{NN}(z)}{dz^2}v_k^{(e)}(z)dz-\mathcal{F}_k^{(e)},\end{aligned} \tag{14}$$

$$\begin{aligned}{}^{(2)}\mathcal{R}_k^{(e)}&=-\int_{z_{e-1}}^{z_e}\tau_{NN}(z)\frac{dv_k^{(e)}(z)}{dz}dz+\left[\tau_{NN}(z)v_k^{(e)}(z)\right]_{z_{e-1}}^{z_e}-\mathcal{F}_k^{(e)}\\&=-\int_{z_{e-1}}^{z_e}m\left|\frac{du_{NN}(z)}{dz}\right|^{n-1}\frac{du_{NN}(z)}{dz}\frac{dv_k^{(e)}(z)}{dz}dz+\left[\tau_{NN}(z)v_k^{(e)}(z)\right]_{z_{e-1}}^{z_e}-\mathcal{F}_k^{(e)}\\&=-\int_{z_{e-1}}^{z_e}m\left|\frac{du_{NN}(z)}{dz}\right|^{n-1}\frac{du_{NN}(z)}{dz}\frac{dv_k^{(e)}(z)}{dz}dz-\mathcal{F}_k^{(e)},\end{aligned} \tag{15}$$



(a) $n = 0.5$ (b) $n = 0.8$ (c) $n = 1.2$

Fig. 2. L_1 norm of forward PINN and VPINN solutions for poiseuille flow of power-law fluid with (a) n=0.5, (b) n=0.8, and (c) 1.2. Alphabets (A)-(F) in x axis indicate the types of models: (A) vanilla PINN, (B) symmetric PINN, and (C) vanilla VPINN with form 1 loss, (D) vanilla VPINN with form 2 loss, (E) symmetric VPINN with form 1 loss, and (F) symmetric VPINN with form 2 loss.

where

$$\mathcal{F}_k^{(e)} = \int_{z_{e-1}}^{z_e} \frac{dp}{dx} v_k^{(e)}(z)dz. \tag{16}$$

$\mathcal{L}_b$ has the same form as that in Eq. (10). N_{el} is the number of elements (subdomains); we applied N_{el}=1. $v_k(z)$ are the test functions that form the test function space onto which the residuals are projected, and $K^{(e)}$ is the number of test functions employed in element e. We took $K^{(e)}$=60 and set the test functions using Legendre polynomials: $v_k(z)=P_{k+1}(z)-P_{k-1}(z)$, where $P_k(z)$ is the Legendre polynomial of degree k, following the choice of the previous study [16]. This reduced the boundary terms initially obtained from integration by parts to zero, as shown in Eqs. (14) and (15), because $v_k(z)=0$ at boundaries according to Kharazmi et al. [16]. To calculate the integral terms, we used the Gauss quadrature rules with Q=80 quadrature points, which are the same as the number of collocation points in PINN.

Finally, based on a study by Mattheakis et al. [18], we applied the so-called hub layer to PINN and VPINN to force symmetry to the results of neural networks (u_{NN}). Initially, without the hub layer, the output of the neural network (u_{NN}) was calculated from the nodes of the last hidden layer in the neural network:

$$u_{NN} = \Sigma_{i=1}^{N} w_i h_i(z) + b, \tag{17}$$

where i, N, and w_i are the the index of the neuron, number of neurons, and weights of the last hidden layer, respectively; b is the bias of the output node, and $h_i(z)$ denotes the activation function of a neuron in the last hidden layer. However, when the hub layer is introduced after the output layer of the deep neural network, u_{NN} is obtained as follows to guarantee symmetry:

$$u_{NN} = \Sigma_{i=1}^{N} \frac{1}{2}(w_i h_i(z) + h_i(-z))) + b, \tag{18}$$

The hub layer can also be employed in an axisymmetric environment in cylindrical or spherical coordinates by substituting z for the axis of symmetry.

In this study, we discuss four types of PINN-based frameworks: PINN and VPINN models with or without the hub layer.

FORWARD PROBLEM-SOLVING VELOCITY PROFILE OF POWER-LAW FLUID IN POISEUILLE FLOW

First, we extended PINN and VPINN to calculate the velocity profile of Poiseuille flow of a power-law fluid. We built deep neural networks with four layers and 32, 16, 16, 32 neurons in each layer, with a swish activation function. Next, we used the Xavier initialization scheme to initialize the weights and biases of the neural networks. Finally, we ran the Adam optimizer with 100,000 iterations and a learning rate of 0.001. We used the loss function value at the end of learning to evaluate the optimization performance of the methods and expressed the deviation of the obtained solutions from the exact solution as L_1 norm of the error: $\Sigma_{i=1}^{N_{test}} |u_{NN}(z_i) - u_{exact}(z_i)|$, where N_{test} represents the number of test points. We also adopted the even metric, $S_+ = (1/N_{test}) \Sigma_{i=1}^{N_{test}} (u_{NN}(z_i) - u_{NN}(-z_i))^2$, from [18] to quantify the degree of symmetry in the solution. The test points were uniformly collocated throughout the domain with a step size of 0.001. We modified the code proposed in [16], which is built upon the TensorFlow library, to introduce the hub layer and impose symmetry in VPINN models.

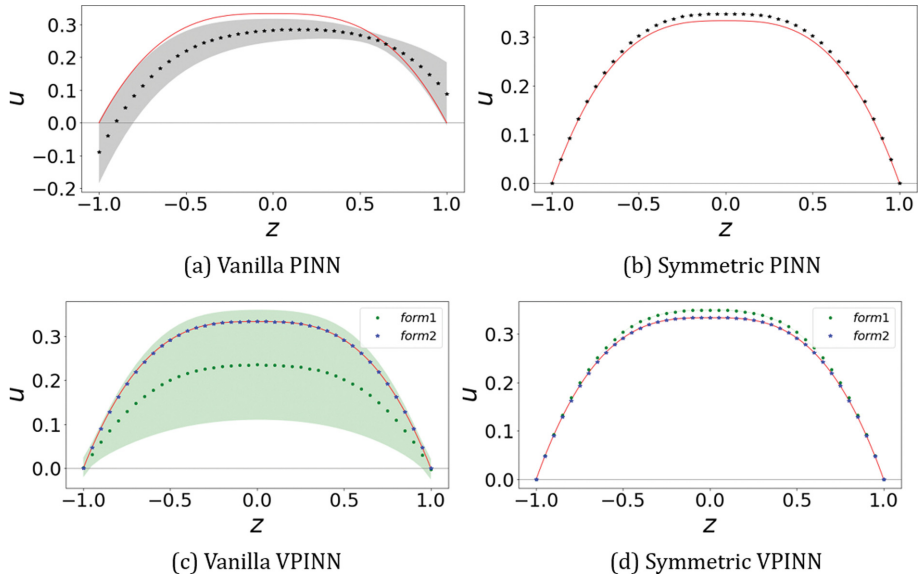
We solved the velocity profile with ten different initializations to generalize the accuracy of PINN-based methods. We considered the system of dp/dx=−1, h=2 and m=1. For n=0.5, 0.8, 1.2 averaged solutions of the four methods, that is, PINN and VPINN with or without the hub layer, are compared with the analytical solutions in Table 1, and their L_1 norm and the solutions of n=0.5 are visualized in Fig. 2 and Fig. 3, respectively. In short, we refer to frameworks with the hub layer as *symmetric PINN* or *symmetric VPINN*, and original frameworks without the hub layer as *vanilla PINN* or *vanilla VPINN*.

In general, the VPINN solutions that use variational loss form 2 agree well with the analytic velocity profile compared with the PINN models. For example, for n=0.5, VPINN with loss form 2 yields the L_1 norm of $O(10^{-2})$, which outperforms the L_1 norm of $O(10^2)$ when PINN is used. While PINN uses the pointwise error of the governing equations at collocation points, VPINN is based on the projection to the test function space and calculates the integral by quadrature rules. In this regard, VPINN can capture the dynamics of the solution over the entire domain more efficiently with fewer points than PINN can.

In the PINN and VPINN models with loss form 1, better symmetry generally makes the NN models more accurate. When loss form 2 is used, the symmetric VPINN solutions still show comparable performance to the vanilla VPINN models. Specifically, reinforcing symmetry in the network structure results in lower variation with comparable performance. Fig. 3 shows that the hub layer generally reduces the deviation of the solutions learned by the PINN

Table 1. L_1 norm, S_+, loss function value of forward PINN and VPINN solutions for Poiseuille flow of power-law fluid with n=0.5, 0.8, 1.2

Methods			L_1 norm		S_+		Loss	
		Loss form	Mean	Stdev	Mean	Stdev	Mean	Stdev
n=0.5								
PINN	Vanilla	.	1.19×10^{2}	7.62×10^{1}	1.38×10^{-2}	1.61×10^{-2}	7.05×10^{-2}	8.00×10^{-2}
	Symmetric	.	1.90×10^{1}	4.02×10^{-1}	7.15×10^{-31}	3.78×10^{-32}	1.71×10^{-5}	1.52×10^{-5}
VPINN	Vanilla	form1	1.73×10^{2}	1.77×10^{2}	7.02×10^{-4}	7.61×10^{-4}	8.50×10^{-3}	9.12×10^{-3}
	Vanilla	form2	1.93×10^{-2}	5.82×10^{-3}	3.05×10^{-10}	2.77×10^{-10}	1.94×10^{-7}	8.94×10^{-8}
	Symmetric	form1	2.10×10^{1}	8.75×10^{-1}	7.04×10^{-31}	2.12×10^{-32}	1.08×10^{-6}	7.34×10^{-7}
	Symmetric	form2	5.87×10^{-2}	3.07×10^{-2}	6.45×10^{-31}	7.81×10^{-33}	2.91×10^{-6}	2.32×10^{-6}
n=0.8								
PINN	Vanilla	.	1.04×10^{1}	9.25×10^{0}	2.50×10^{-4}	3.17×10^{-4}	1.18×10^{-3}	7.21×10^{-4}
	Symmetric	.	3.93×10^{0}	2.37×10^{-2}	9.57×10^{-31}	7.98×10^{-32}	2.57×10^{-6}	2.53×10^{-6}
VPINN	Vanilla	form1	4.14×10^{0}	1.61×10^{-1}	5.70×10^{-10}	1.44×10^{-9}	1.07×10^{-6}	2.50×10^{-6}
	Vanilla	form2	1.82×10^{-2}	8.15×10^{-3}	2.31×10^{-10}	1.88×10^{-10}	2.40×10^{-7}	1.93×10^{-7}
	Symmetric	form1	4.09×10^{0}	5.74×10^{-2}	9.27×10^{-31}	1.36×10^{-32}	2.04×10^{-7}	1.26×10^{-7}
	Symmetric	form2	1.37×10^{-2}	5.79×10^{-3}	9.18×10^{-31}	1.29×10^{-32}	3.02×10^{-7}	2.73×10^{-7}
n=1.2								
PINN	Vanilla	.	2.01×10^{0}	4.03×10^{-2}	8.72×10^{-8}	1.79×10^{-7}	9.31×10^{-7}	8.95×10^{-7}
	Symmetric	.	1.98×10^{0}	1.68×10^{-2}	1.21×10^{-30}	3.18×10^{-32}	2.18×10^{-6}	2.54×10^{-6}
VPINN	Vanilla	form1	1.92×10^{0}	3.34×10^{-2}	4.37×10^{-10}	8.60×10^{-10}	1.87×10^{-7}	1.73×10^{-7}
	Vanilla	form2	1.36×10^{-2}	3.22×10^{-3}	1.06×10^{-10}	5.78×10^{-11}	1.47×10^{-7}	1.58×10^{-7}
	Symmetric	form1	1.89×10^{0}	4.15×10^{-2}	1.19×10^{-30}	9.96×10^{-33}	3.70×10^{-7}	2.71×10^{-7}
	Symmetric	form2	1.66×10^{-2}	1.19×10^{-2}	1.20×10^{-30}	6.39×10^{-33}	3.39×10^{-7}	3.38×10^{-7}



(a) Vanilla PINN (b) Symmetric PINN

(c) Vanilla VPINN (d) Symmetric VPINN

Fig. 3. Velocity profile obtained from forward PINN and VPINN solutions when n=0.5 with 100,000 iterations: (a) vanilla PINN, (b) symmetric PINN, (c) vanilla VPINN, and (d) symmetric VPINN. Red solid lines represent exact solutions, dots represent predicted solutions, and the shaded parts represent standard deviation (yet invisibly small in (b) and (d)).

and VPINN. For n=0.5 and 0.8, the hub layer decreases the standard deviation of L_1 norm by more than 50%. In particular, in the PINN framework, the hub layer additionally improves the accuracy of the solutions.

Note that the learned velocity profile becomes less accurate as the power-law index (n) decreases. We postulate that the loss function landscape may become less favorable in the optimization process at a low power-law index.

1. PINN and VPINN on the Symmetric Half of Domain

The conventional method to obtain a symmetric solution is to

Table 2. L_1 norm and loss function values of forward PINN and VPINN solutions in the symmetric half of domain with n=0.8

Methods	L_1 norm		Loss	
	Mean	Stdev	Mean	Stdev
PINN	1.96×10^{0}	5.44×10^{-1}	7.50×10^{-8}	4.25×10^{-8}
VPINN	2.40×10^{0}	4.73×10^{-1}	1.12×10^{-7}	4.66×10^{-8}

solve differential equations in half of the domain and expand the solution. Using the symmetric half of the domain is advantageous in terms of lower computational cost.

In this regard, we employed PINN and VPINN on the symmetric half of the domain Ω=[0, h/2] with different boundary conditions: du/dz=0 at z=0 and u(z)=0 at z=h/2. We solved the same forward problem with the symmetric half of the domain to compare with the hub layer frameworks by applying different boundary losses ($\mathcal{L}_b$):

$$\mathcal{L}_b = \tau_d\left|\frac{du_{NN}}{dz}\bigg|_{z=0}\right|^2 + (1-\tau_d)\left|u_{NN}\left(\frac{h}{2}\right)\right|^2, \tag{19}$$

where τ_d and $(1-\tau_d)$ are weights that adjust the contributions of the boundary conditions, and we used τ_d=0.5 in this study.

When comparing the L_1 norm and loss function values presented in Table 1 and 2, the symmetry-forced VPINN solution agreed better with the analytical solution than the VPINN solution learned in the symmetric half of the domain. Owing to the hub layer, a symmetry constraint is imposed on the neural network, narrowing down the computed solutions of the neural network. On the other hand, the symmetric half-domain method technically guarantees symmetry by decomposing the domain and does not directly affect the optimization process.

We note that the symmetric PINN model shows an accuracy similar to that of the symmetric half-domain PINN model. Referring to Section 4, PINN tends to fall into local minima regardless of the existence of the hub layer and fails to achieve a well-agreeing solution as evidenced by the deviation of trained results from the exact solution despite an optimized loss function value.

INVERSE PROBLEM-LEARNING VISCOSITY OF POWER-LAW FLUID

We applied PINN and VPINN to learn the power-law consistency coefficient (m) and power-law index (n) from the velocity profile data. Only variational residual form 2 was tested in VPINN owing to its better performance in Section 3.

Kim et al. [19] used fluorescent beads to obtain the velocity data in microchannels where particle migration is observed as in the power-law fluid [20,21]. The study reported that fewer fluorescent particles were located near the walls. To this end, we first randomly collocated data points from z domain (−h/2, h/2) to follow a normal distribution of N(0, h/4) and generate velocity data at each point based on Eq. (6) with a 5% Gaussian noise. In the same study, 20 points were sampled to train the neural network. We used the same neural network structure, hyperparameters, and Adam optimizer as described in Section 3 to learn the power-law fluid parameters and train models with 100,000 iterations to compare the performance of each model. The loss functions for PINN and VPINN still take the forms presented in Eqs. (8) and (11). However, data loss terms in mean squared error sense, i.e., $1/N\Sigma_i^N |u_{NN}(x_i)-u_i|$, have to be added such that the velocity fields predicted by the network, u_{NN}, follow N experimental data points, u_i. The results of the prediction of the unknown parameters are summarized in Table 3. The accuracy of the simultaneously learned velocity solutions is described in Table 4, and the predicted velocity profiles when n=0.6 are visualized in Fig. 4.

Among the models considered in this study, the symmetric VPINN exhibits the best performance in solving the inverse problem. We observe that the VPINN with a hub layer accurately predicts m and n, values and the prediction remains robust even in several random initializations. The hub layer generally improves the accuracy of the models, and the PINN solutions do not calculate the power-law fluid properties well, even though their loss function values are small. We presume that PINNs are prone to falling

Table 3. Predicted values of power-law consistency coefficient (m) and power-law index (n) by PINN and VPINN for m=1 and n=0.6, 0.8. Only the form 2 variational residual loss function (Eq. (15)) was used in VPINN

Methods		m		n	
		Mean	Stdev	Mean	Stdev
n=0.6					
PINN	Vanilla	8.59×10^{-1}	6.16×10^{0}	-4.40×10^{-2}	6.16×10^{0}
	Symmetric	-5.47×10^{0}	8.62×10^{0}	9.72×10^{-1}	8.62×10^{0}
VPINN	Vanilla	1.07×10^{0}	2.25×10^{-1}	6.37×10^{-1}	2.25×10^{-1}
	Symmetric	1.01×10^{0}	1.54×10^{-2}	6.00×10^{-1}	1.54×10^{-2}
n=0.8					
PINN	Vanilla	4.31×10^{0}	3.01×10^{0}	-6.12×10^{-2}	3.01×10^{0}
	Symmetric	-1.27×10^{0}	4.68×10^{0}	8.13×10^{-1}	4.68×10^{0}
VPINN	Vanilla	1.03×10^{0}	8.30×10^{-2}	8.19×10^{-1}	8.30×10^{-2}
	Symmetric	1.01×10^{0}	4.37×10^{-2}	8.12×10^{-1}	4.37×10^{-2}

Table 4. L_1 norm, L_+, and loss function values of velocity profiles obtained from PINN and VPINN in inverse problem with n=0.6, 0.8

Methods		L_1 norm		S_+		Loss	
		Mean	Stdev	Mean	Stdev	Mean	Stdev
n=0.6							
PINN	Vanilla	1.37×10^{2}	4.55×10^{1}	8.21×10^{-3}	9.64×10^{-3}	4.33×10^{-3}	1.82×10^{-3}
	Symmetric	9.00×10^{1}	7.81×10^{1}	2.88×10^{-31}	2.38×10^{-31}	2.96×10^{-3}	3.78×10^{-3}
VPINN	Vanilla	8.39×10^{0}	1.01×10^{1}	4.40×10^{-5}	8.21×10^{-5}	1.36×10^{-5}	6.69×10^{-6}
	Symmetric	2.80×10^{0}	1.26×10^{0}	7.33×10^{-31}	2.29×10^{-32}	1.28×10^{-5}	7.29×10^{-6}
n=0.8							
PINN	Vanilla	1.76×10^{2}	6.46×10^{1}	1.97×10^{-2}	2.23×10^{-2}	6.18×10^{-3}	4.07×10^{-3}
	Symmetric	6.64×10^{1}	8.67×10^{1}	9.41×10^{-31}	7.61×10^{-31}	2.00×10^{-3}	3.51×10^{-3}
VPINN	Vanilla	7.10×10^{0}	5.21×10^{0}	3.14×10^{-5}	4.64×10^{-5}	1.89×10^{-5}	9.77×10^{-6}
	Symmetric	3.81×10^{0}	2.23×10^{0}	9.18×10^{-31}	5.67×10^{-32}	2.23×10^{-5}	8.20×10^{-6}

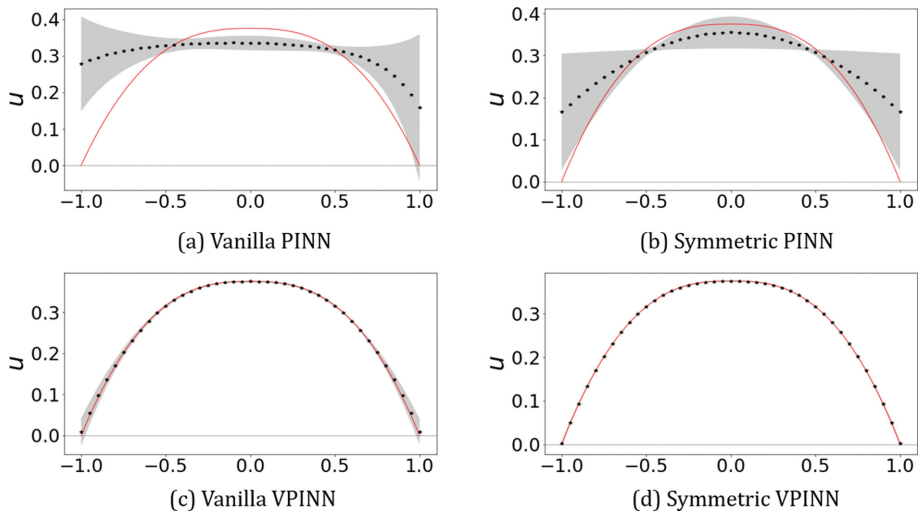


Fig. 4. Velocity profile obtained from inverse PINN and VPINN solutions when n=0.6 with 100,000 iterations: (a) vanilla PINN, (b) symmetric PINN, (c) vanilla VPINN, and (d) symmetric VPINN. Red solid lines indicate exact solutions, dots indicate predicted solutions, and shaded parts denote standard deviation (yet invisibly small in (d)).

into local minima when used with the Adam optimizer and trained for a limited number of iterations. In contrast, VPINNs are advantageous for capturing the global minima of solutions and exhibit robust performance.

According to Shin et al. [22], the total errors of the neural network solutions consist of three components: approximation, optimization, and estimation errors. The approximation error is the error between the exact solution u^* and output u_{NN} of the ideally optimized neural network that has been trained with an infinite amount of data. In particular, the approximation error is defined as the minimum error between u^* and $\mathcal{H}_m$ where $\mathcal{H}_m$ is the space that can be expressed by a neural network. The hub layer can reduce the approximation error of neural networks by imposing an additional constraint, that is, symmetry in optimizing process and guiding $\mathcal{H}_m$ to be closer to u^*. PINN-based methods can also be improved by adjusting the weights of the loss function; however, optimizing another hyperparameter requires further experiments. In contrast, introducing a hub layer can be a straightforward method for improving the PINN-based models.

Next, we varied the number of training points to solve the inverse problem. We tested PINN with 20,80, and 150 data points and VPINN with 5, 10, and 20 data points. The results are presented in Table 5.

Table 5 shows that VPINN efficiently learns power-law fluid properties, even with a far smaller data size than PINN. However, using larger datasets did not improve the performance of the PINN.

In summary, the hub layer works efficiently in reducing the approximation errors of neural networks, as shown in Section 3, and the VPINN models are well trained even with a limited amount of data.

FINAL REMARKS

We propose a symmetric VPINN to model symmetric fluid flows. In particular, we used symmetric VPINN and other PINN-based models in the learning velocity profile and power-law fluid properties in the Poiseuille flow of a power-law fluid, and evaluated the performance. Unlike PINN-based methods, symmetric VPINN

Table 5. Predicted values of power-law consistency coefficient (m) and power-law index (n) from m=1 and n=0.6, 0.8 by (a) VPINN with form 2 variational residual loss function and (b) PINN with different number of training points

(a) Inverse problem solved with VPINN

Methods		Training points	m		n	
			Mean	Stdev	Mean	Stdev
n=0.6						
VPINN	Vanilla	5	1.59×10^{0}	5.57×10^{-1}	9.13×10^{-1}	5.57×10^{-1}
	Symmetric		8.28×10^{-1}	1.98×10^{-1}	4.88×10^{-1}	1.98×10^{-1}
	Vanilla	10	1.15×10^{0}	6.03×10^{-1}	6.66×10^{-1}	6.03×10^{-1}
	Symmetric		9.79×10^{-1}	1.10×10^{-1}	5.84×10^{-1}	1.10×10^{-1}
	Vanilla	20	1.07×10^{0}	2.25×10^{-1}	6.37×10^{-1}	2.25×10^{-1}
	Symmetric		1.01×10^{0}	1.54×10^{-2}	6.00×10^{-1}	1.54×10^{-2}
n=0.8						
VPINN	Vanilla	5	-3.45×10^{0}	9.21×10^{0}	8.54×10^{-1}	9.21×10^{0}
	Symmetric		8.49×10^{-1}	2.24×10^{-1}	6.71×10^{-1}	2.24×10^{-1}
	Vanilla	10	1.05×10^{0}	9.60×10^{-2}	8.62×10^{-1}	9.60×10^{-2}
	Symmetric		9.95×10^{-1}	6.90×10^{-2}	8.02×10^{-1}	6.90×10^{-2}
	Vanilla	20	1.03×10^{0}	8.30×10^{-2}	8.19×10^{-1}	8.30×10^{-2}
	Symmetric		1.01×10^{0}	4.37×10^{-2}	8.12×10^{-1}	4.37×10^{-2}

(b) Inverse problem solved with PINN

Methods		Training points	m		n	
			Mean	Stdev	Mean	Stdev
n=0.6						
PINN	Vanilla	20	8.59×10^{-1}	6.16×10^{0}	-4.40×10^{-2}	6.16×10^{0}
	Symmetric		-5.47×10^{0}	8.62×10^{0}	9.72×10^{-1}	8.62×10^{0}
	Vanilla	80	3.29×10^{0}	2.25×10^{0}	-7.48×10^{-2}	2.25×10^{0}
	Symmetric		4.18×10^{-1}	3.01×10^{0}	9.17×10^{-1}	3.01×10^{0}
	Vanilla	150	3.16×10^{0}	4.90×10^{0}	-1.11×10^{-1}	4.90×10^{0}
	Symmetric		-4.07×10^{0}	8.51×10^{0}	9.81×10^{-1}	8.51×10^{0}
n=0.8						
PINN	Vanilla	20	4.31×10^{0}	3.01×10^{0}	-6.12×10^{-2}	3.01×10^{0}
	Symmetric		-1.27×10^{0}	4.68×10^{0}	8.13×10^{-1}	4.68×10^{0}
	Vanilla	80	2.13×10^{0}	4.18×10^{0}	-7.06×10^{-5}	4.18×10^{0}
	Symmetric		-1.83×10^{0}	5.88×10^{0}	8.35×10^{-1}	5.88×10^{0}
	Vanilla	150	3.78×10^{0}	2.06×10^{0}	4.17×10^{-2}	2.06×10^{0}
	Symmetric		-2.01×10^{0}	5.99×10^{0}	6.95×10^{-1}	5.99×10^{0}

models accurately and robustly approximate analytical velocity solutions. Overall, VPINN shows better accuracy than PINN, and the hub layer can improve the performance of neural networks compared with using the symmetry boundary condition explicitly. As in solving the forward problem, the symmetric VPINN shows high accuracy in predicting power-law fluid properties. We noticed that a symmetric VPINN could learn reasonable values even from a limited dataset. The symmetric VPINN model especially shows good performance when the power-law index n is low, in contrast to the difficulties in the prediction of other PINN-based models. In particular, the hub layer improves the performance of the VPINN by limiting the space expressed by the neural networks. Therefore, the symmetric VPINN shows higher accuracy and less deviation in the forward problem and is trained effectively with scarce data in the inverse problem than any other tested models.

PINN can be improved by strictly imposing boundary conditions; however, additional hyperparameters remain to be tuned. For example, in this study, τ_b in Eq. (8) and τ_d in Eq. (19) can be optimized; however, there is no clear consensus on the selection of hyperparameters in the absence of a final optimized solution. Instead, introducing a hub layer requires more computational load in the backpropagation step of neural networks but is advantageous as it avoids additional hyperparameter tuning.

The PINN and VPINN frameworks may not be appropriate for imposing natural boundary conditions. In the symmetric half-domain method tested in Section 3.1, we used the first derivative

of u_{NN} at z=0 as the boundary condition to impose a penalty in the loss function. However, as shown in Eq. (10), the boundary conditions of PINN and VPINN are the calculated values of the velocity profile, not the derivatives. This may result in larger errors for methods that use the symmetric half of the domain than for symmetric VPINN frameworks. While PINN essentially uses data collocation in training, VPINN is trained via projection to the test function space, leaving the possibility of improvement by appropriately applying natural boundary conditions.

Finally, including the known condition of systems directly into the network, rather than penalizing the network with the loss function can improve the predictability of the models. We further suggest that translating prior knowledge of the system under consideration, other than symmetry, into the network architecture may further enhance the accuracy of neural networks.

ACKNOWLEDGEMENTS

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (Ministry of Science and ICT, MSIT) (No. NRF-2018R1A5A1024127, NRF-2020R1A2C2008141, and NRF-2021M3H4A6A01041234).

REFERENCES

1. M. Dissanayake and N. Phan-Thien, *Commun. Numer. Methods Eng.*, **10**(3), 195 (1994).
2. J. Berg and K. Nyström, *Neurocomputing*, **317**, 28 (2018).
3. H. Sun, M. Hou, Y. Yang, T. Zhang, F. Weng and F. Han, *Neural Process. Lett.*, **50**(2), 1153 (2019).
4. S. F. Masri, A. G. Chassiakos and T. K. Caughey, *J. Appl. Mech.*, **60**(1), 123 (1993).
5. Y.-Y. Lin, J.-Y. Chang and C.-T. Lin, *IEEE Trans. Neural Networks Learning Syst.*, **24**(2), 310 (2012).
6. Y. Pan and J. Wang, *IEEE Trans. Ind. Electron.*, **59**(8), 3089 (2011).
7. B. Reyes, A. A. Howard, P. Perdikaris and A. M. Tartakovsky, *Phys. Rev. Fluids*, **6**(7), 073301 (2021).
8. J. Taskinen and J. Yliruusi, *Adv. Drug Deliv. Rev.*, **55**(9), 1163 (2003).
9. R. Cang, H. Li, H. Yao, Y. Jiao and Y. Ren, *Comput. Mater. Sci.*, **150**, 212 (2018).
10. X. Jin, S. Cai, H. Li and G. E. Karniadakis, *J. Comput. Phys.*, **426**, 109951 (2021).
11. M. M. Almajid and M. O. Abu-Al-Saud, *J. Pet. Sci. Eng.*, **208**, 109205 (2022).
12. S. Cai, Z. Wang, S. Wang, P. Perdikaris and G. E. Karniadakis, *J. Heat Transfer*, **143**(6), 060801 (2021).
13. X. Meng, Z. Li, D. Zhang and G. E. Karniadakis, *Comput. Methods Appl. Mech. Eng.*, **370**, 113250 (2020).
14. A. D. Jagtap, E. Kharazmi and G. E. Karniadakis, *Comput. Methods Appl. Mech. Eng.*, **365**, 113028 (2020).
15. 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., in: AAAI Spring Symposium: MLPS (2021).
16. E. Kharazmi, Z. Zhang and G. E. Karniadakis, *Comput. Methods Appl. Mech. Eng.*, **374**, 113547 (2021).
17. M. Yang and J. T. Foster, *J. Machine Learning Model. Computing*, **2**(2), 15 (2021).
18. M. Mattheakis, P. Protopapas, D. Sondak, M. Di Giovanni and E. Kaxiras, *Physical symmetries embedded in neural networks*, arXiv preprint arXiv:1904.08991 (2019).
19. D. Kim, J. Park and J. Nam, *Chem. Eng. Sci.*, **245**, 116972 (2021).
20. F. E. Chrit, S. Bowie and A. Alexeev, *Phys. Fluids*, **32**(8), 083103 (2020).
21. X. Hu, J. Lin, D. Chen and X. Ku, *Biomicrofluidics*, **14**(1), 014105 (2020).
22. Y. Shin, *Commun. Comput. Phys.*, **28**(5), 2042 (2020).
23. okada39, pinn cavity, https://github.com/okada39/pinn_cavity (2020).

APPENDIX A. CHOICE OF ACTIVATION FUNCTION

We note that the second derivative of the velocity profile computed by VPINN shows a deviation near the boundaries, possibly owing to the diminishing second derivative of the swish activation function. To compare the activation functions, we solved the forward problem with the symmetric VPINN using variational residual form 2 and applying several activation functions: swish, sigmoid, and sin. The network structure was constructed as described in Section 3.

Fig. A1 shows positive deviation near boundaries in the second derivative of VPINN solutions using swish or sigmoid activation functions, whose second derivatives diminish to zero. Conversely, the boundary error of the second derivative exhibits randomness when the sine activation function is used. However, this deviation near boundaries is difficult to avoid unless the swish activation function provides better accuracy in learning velocity profile than other activation functions, as shown in Table A1.

Table A1. Loss function value, L_1 norm, and SSE of forward symmetric VPINN solutions for the Poiseuille flow of power-law fluid with activation function of swish, sigmoid, and sin

Activation	L_1 norm		Loss	
Function	Mean	Stdev	Mean	Stdev
Swish	5.97×10^{-2}	2.92×10^{-2}	2.99×10^{-6}	1.81×10^{-6}
Sigmoid	1.33×10^{-2}	9.34×10^{-3}	2.85×10^{-7}	2.73×10^{-7}
Sin	6.03×10^{-2}	2.27×10^{-2}	4.38×10^{-6}	1.80×10^{-6}

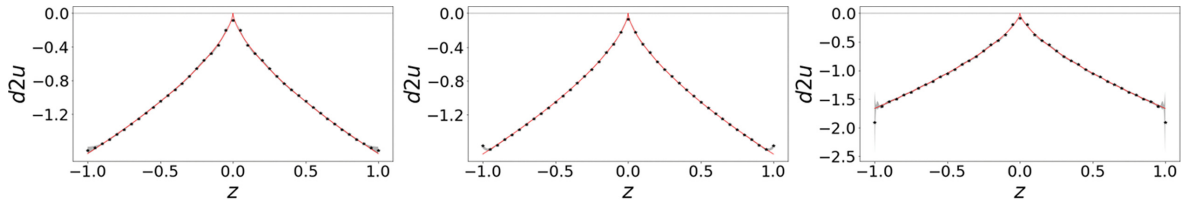


Fig. A1. Second derivative of velocity profile solved with symmetric VPINN and 100,000 iterations when n=0.6 and different activation functions are used: (a) swish, (b) sigmoid, and (c) sin. Red solid lines indicate exact solutions, dots indicate predicted solutions, and shaded parts indicate standard deviation.

APPENDIX B. CHOICE OF HYPERPARAMETERS OF VPINN

We selected the hyperparameters used in PINN and VPINN through the case study as shown in Table B1 and B2, respectively. We used lid-driven cavity flow and one-dimensional Poisson's equation as test cases to solve with PINN and VPINN, respectively [23,16].

Table B1. Sum of squared error and L_{inf} norm of PINN solutions for lid-driven cavity flow with different hyperparameter values

Initializer	Activation function	Layer of neural network	Sum of squared error	L_{inf} norm
He	swish	32.16.16.32	7.62×10^{0}	9.49×10^{-1}
Xavier	swish	32.16.16.32	7.30×10^{0}	9.25×10^{-1}
zeros	swish	32.16.16.32	2.68×10^{1}	1.00×10^{0}
He	tanh	32.16.16.32	8.06×10^{0}	9.96×10^{-1}
He	swish	20.20.20	9.77×10^{0}	9.73×10^{-1}
He	swish	20.20.20.20	6.11×10^{0}	8.85×10^{-1}
He	swish	40.40.40	7.04×10^{0}	9.2×10^{-1}
He	swish	40.40.40.40	6.08×10^{0}	9.1×10^{-1}
He	swish	32.16.16.16.32	6.67×10^{0}	8.98×10^{-1}

Table B2. Loss function value and L_1 norm of VPINN solutions for one-dimensional Poisson's equation with different hyperparameter values

Hyperparameter		Loss	Norm
Initializer	He	1.67×10^{0}	3.67×10^{-5}
	Xavier	3.51×10^{0}	1.34×10^{-4}
	zeros	1.98×10^{3}	1.05×10^{2}
Layer of neural network	20.20.20	6.94×10^{-1}	9.74×10^{-4}
	20.20.20.20	1.67×10^{0}	3.67×10^{-5}
	32.16.16.32	4.78×10^{-2}	8.76×10^{-5}
	32.16.16.16.32	3.90×10^{-2}	1.79×10^{-5}
	40.40.40	1.20×10^{-1}	1.25×10^{-4}
	40.40.40.40	2.22×10^{-1}	1.39×10^{-5}
Number of test functions	30	3.13×10^{0}	7.37×10^{-6}
	60	1.67×10^{0}	3.67×10^{-5}
	100	8.09×10^{-1}	3.68×10^{-5}
	150	1.93×10^{2}	1.92×10^{1}
	200	1.94×10^{3}	3.44×10^{1}
Number of quadrature points	40	1.78×10^{1}	3.75×10^{0}
	80	1.67×10^{0}	3.67×10^{-5}
	150	1.62×10^{0}	3.77×10^{-5}
	200	1.64×10^{0}	3.81×10^{-5}
	250	1.28×10^{0}	3.79×10^{-5}
	300	1.66×10^{0}	3.88×10^{-5}