JUSTC

Article

http://justc.ustc.edu.cn

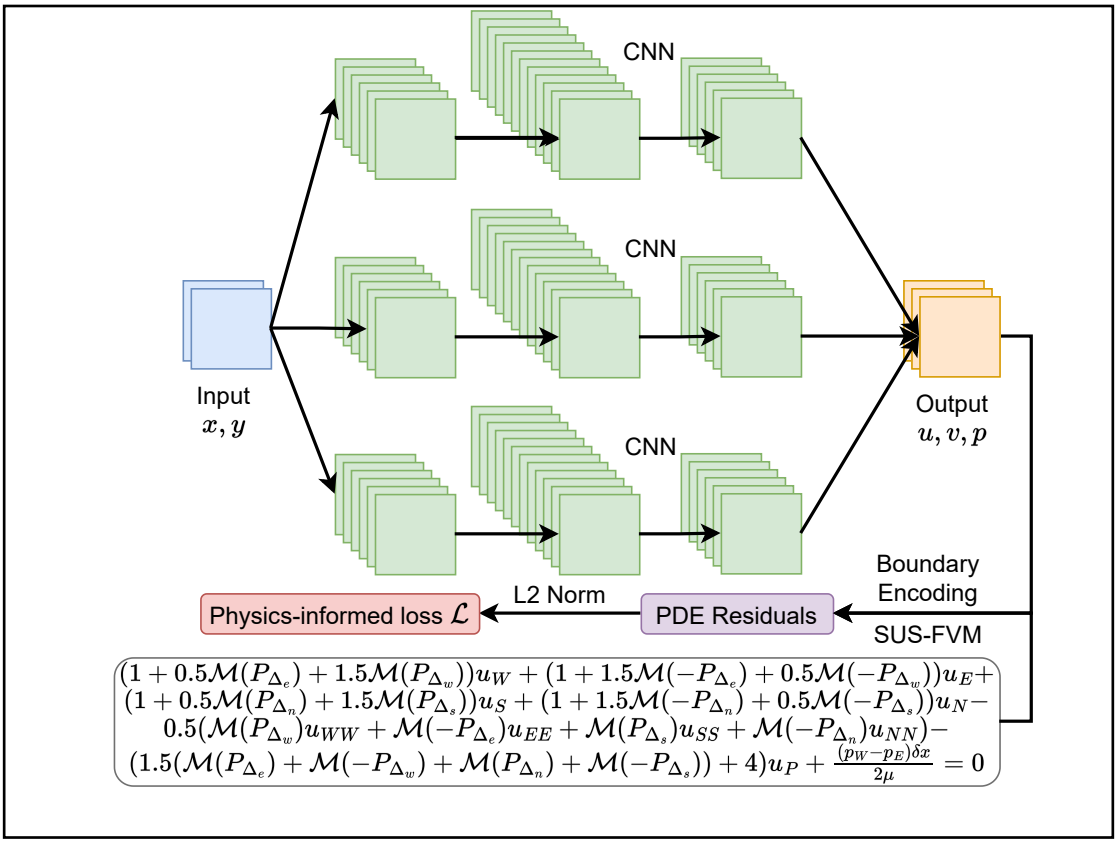
Physically plausible and conservative solutions to Navier-Stokes equations using Physics-Informed CNNs

Jianfeng Li, Liangying Zhou, Jingwei Sun ✉, and Guangzhong Sun

[1]*School of Computer Science and Technology, University of Science and Technology of China, Hefei, 230026, China*

✉Correspondence: Jingwei Sun, E-mail: sunjw@ustc.edu.cn

Graphical abstract



Schematic illustration for solving the Navier-Stokes equations using physics-informed CNN.

Public summary

- We use the finite volume method to derive the discrete schemes of the conserved Navier-Stokes equations to ensure that the resulting discrete equations are conserved.
- We use the second-order upwind difference scheme to discrete the Navier-Stokes equations for constructing the loss function, which avoids the introduction of downstream information into the discrete equations.
- We hard impose boundary conditions to allow the physics-informed convolutional neural network model to train and predict without any labeled data.

Citation: Li J F, Zhou L Y, Sun J W, et al. Physically plausible and conservative solutions to Navier-Stokes equations using Physics-Informed CNNs. *JUSTC*, **2023**, 53(0): . DOI: 10.52396/JUSTC-2022-0174

JUSTC

Article

http://justc.ustc.edu.cn

Physically plausible and conservative solutions to Navier-Stokes equations using Physics-Informed CNNs

Jianfeng Li, Liangying Zhou, Jingwei Sun ✉, and Guangzhong Sun

[1]*School of Computer Science and Technology, University of Science and Technology of China, Hefei, 230026, China*

✉Correspondence: Jingwei Sun, E-mail: sunjw@ustc.edu.cn

Cite This: *JUSTC*, **2023**, 53(X): (12pp) Read Online SI Supporting Information

Abstract: Physics-informed Neural Network (PINN) is an emerging approach for efficiently solving partial differential equations (PDEs) using neural networks. Physics-informed Convolutional Neural Network (PICNN), a variant of PINN enhanced by convolutional neural networks (CNNs), has achieved better results on a series of PDEs since the parameter-sharing property of CNNs is effective to learn spatial dependencies. However, applying existing PICNN-based methods to solve Navier-Stokes equations can generate oscillating predictions, which are inconsistent with the laws of physics and the conservation properties. To address this issue, we propose a novel method that combines PICNN with the finite volume method to obtain physically plausible and conservative solutions to Navier-Stokes equations. We derive the second-order upwind difference scheme of Navier-Stokes equations using the finite volume method. Then we use the derived scheme to calculate the partial derivatives and construct the physics-informed loss function. The proposed method is assessed by experiments on steady-state Navier-Stokes equations under different scenarios, including convective heat transfer, lid-driven cavity flow, etc. The experimental results demonstrate that our method can effectively improve the plausibility and the accuracy of the predicted solutions from PICNN.

Keywords: Finite volume method; Navier-Stokes equation; Partial differential equation; Physics-informed convolutional neural network

CLC number: **Document code:** A

1 Introduction

Partial differential equations (PDEs) are ubiquitous in scientific fields, such as mechanics, engineering, and economics, and are often solved using numerical methods. Although numerical methods, including the finite difference method (FDM), finite volume method (FVM), and finite element method (FEM), have made successful progress in solving PDEs over the past few decades, their high computational overhead still makes the simulation difficult in many scenarios. For instance, conventional numerical solvers are computationally expensive for complex systems with multiscale/multiphysics features, e.g., optimization [1], inverse problem [2], and uncertainty quantification [3].

In recent years, with the development of deep learning, solving partial differential equations using neural networks has attracted researchers' interest. Neural networks show strong competitiveness compared with numerical methods in solving PDEs because of their generalization ability, lower implementation cost, and lower computational cost [4]. Since the actual values of PDEs' solution are expensive to acquire in real-world scenarios, it is not practical to train neural network models with a large amount of labeled data. Besides, in the absence of physical constraints, neural networks may produce predictions that violate the laws of physics and do not generalize well. Raissi et al. [5] proposed physics-informed neural network (PINN) for solving PDEs. PINN uses the automatic differentiation technique [5] to calculate partial derivatives and uses the equation residuals and boundary/initial conditions as penalty terms to construct the neural network's loss function. By combining physical information with neural networks, PINN effectively reduces the amount of labeled data required for training. However, the slow convergence of PINN limits its performance in solving PDEs. Physics-informed convolutional neural network (PICNN), a variant of PINN, achieves better results on a series of PDEs due to better scalability and faster convergence of CNNs. Besides, the parameter-sharing property of CNNs makes PICNN more effective to learn spatial dependencies. PICNN computes partial derivatives using central difference convolutional filters, whose parameters are derived by the finite difference method and encodes PDEs into the loss function [6, 7]. Compared to PINN, PICNN has the following three advantages. **First**, PICNN is more efficient in computing higher-order partial derivatives. PINN uses automatic differentiation to calculate partial derivatives, which makes the computational overhead increase exponentially as the order of PDE increases. While PICNN uses numerical methods to calculate partial derivatives, which makes the computational overhead increase linearly as the order of PDE increases. **Second**, PICNN imposes boundary conditions instead of using them as penalty terms of the loss function, so that the boundary conditions can be en-

DOI: 10.52396/JUSTC-2022-0174
JUSTC, **2023**, 53(X):

forced. **Third**, PICNN leverages convolutional NNs for modeling nonlinear dynamics, which is capable of well portraying the local solution details [8].

Although PICNN has achieved promising results in solving many PDEs, its performance is still unsatisfactory for equations containing convection terms such as convection-diffusion equations and Navier-Stokes equations. Navier-Stokes equations are widely used in the field of computational fluid dynamics and numerical heat transfer. The characteristic of the Navier-Stokes equations is that the convection terms they contain have strong directionality, which describes the directional motion of the fluid. Under the action of convection, the disturbance at a certain point can only be transmitted downstream but not reversely. Existing PICNN-based methods [9, 10] discretize the equations using a central difference scheme, which in some cases leads to implausible solutions, i.e., oscillations of the solutions. Besides, the discrete scheme derived from the FDM does not have conservation properties, which can further lead to inaccurate predictions.

In this study, we try to properly combine numerical methods with CNNs to ensure that more accurate physical information is encoded into the neural networks so that the predicted solutions of the Navier-Stokes equations are more in line with physical laws. **First**, we use the FVM to derive the discrete schemes of the conserved PDEs to ensure that the resulting discrete equations are conserved. The use of discrete equations with conservative forms can describe physical systems more accurately than equations with non-conservative forms, resulting in solutions that are more in line with physical laws [11]. **Second**, we use the second-order upwind difference scheme to discrete the Navier-Stokes equations for constructing the loss function. The second-order upwind difference scheme can obtain information from the upstream properly, which avoids the introduction of downstream information into the discrete equations. **Third**, we hard impose boundary conditions to allow the PICNN to train and predict without any labeled data.

2 Related work

In many scientific computing scenarios, it is difficult to obtain labeled data (e.g., coordinates and corresponding physical quantities). Using purely data-driven deep learning methods to solve scientific computing problems is neither cost-effective nor able to obtain solutions that obey the laws of physics. By combining prior knowledge with neural networks, solutions can be learned with less data [2, 12–14] or even without any data [6, 15, 16]. Based on PINN [2], a series of PINN variants have been proposed to deal with more challenging problems, such as UQPINN [17], PPINN [18], fPINN [19], vPINN [20], and B-PINNs [13, 21].

Recent studies have shown that CNNs are suitable for solving large-scale, high-dimensional spatiotemporal problems due to their parameter-sharing features [22]. For steady-state PDEs, Zhu et al. design a physics-informed convolutional encoder-decoder structure to learn solutions without using any labeled data [15, 23]. Gao et al. design a geometric adaptation strategy [9], which transforms the irregular geometric domain into a regular rectangular domain through a coordinate transformation so that the irregular domain can be processed by CNN. Geneva et al. propose the AR-DenseED method [6], which uses a PICNN-based deep autoregressive model to predict dynamical PDEs. Ren et al. design a surrogate model [10] based on the AR-DenseED method to learn the time evolution of dynamical partial differential equations.

Existing studies have achieved promising results in solving fluid and heat transfer problems using PICNN [24–28]. Most of these methods use the FDM-based central difference scheme to discretize the convection term. Bar-Sinai et al. [29] introduces the FVM to CNN, enabling the model to generate coefficients for approximating spatial derivatives. However, the method proposed in [29] is data-driven and does not work in scenarios where labeled data is expensive and scarce.

3 Methodology

3.1 Navier-Stokes Equations

Equation (1) represents the steady-state Navier-Stokes equations in a two-dimensional Cartesian coordinate system, where ρ is the fluid density, u and v are the flow velocity in the x-axis and y-axis directions, respectively, μ is the viscosity and p is the pressure. The terms $\frac{\partial(\rho uu)}{\partial x}$, $\frac{\partial(\rho vu)}{\partial y}$, $\frac{\partial(\rho uv)}{\partial x}$, and $\frac{\partial(\rho vv)}{\partial y}$ in the first two rows of Equation (1) are convection terms, which have strong directionality and describe the directional motion of the fluid. The terms $\frac{\partial}{\partial x}\left(\mu\frac{\partial u}{\partial x}\right)+\frac{\partial}{\partial y}\left(\mu\frac{\partial u}{\partial y}\right)$ and the $\frac{\partial}{\partial x}\left(\mu\frac{\partial v}{\partial x}\right)+\frac{\partial}{\partial y}\left(\mu\frac{\partial v}{\partial y}\right)$ in the first two rows of Equation (1) are the diffusion terms, which describe the irregular thermal motion of molecules.

$$\begin{aligned}
&\frac{\partial(\rho uu)}{\partial x}+\frac{\partial(\rho vu)}{\partial y}=-\frac{\partial p}{\partial x}+\frac{\partial}{\partial x}(\mu\frac{\partial u}{\partial x})+\frac{\partial}{\partial y}(\mu\frac{\partial u}{\partial y})\\
&\frac{\partial(\rho uv)}{\partial x}+\frac{\partial(\rho vv)}{\partial y}=-\frac{\partial p}{\partial y}+\frac{\partial}{\partial x}(\mu\frac{\partial v}{\partial x})+\frac{\partial}{\partial y}(\mu\frac{\partial v}{\partial y})\\
&\frac{\partial(\rho u)}{\partial x}+\frac{\partial(\rho v)}{\partial y}=0
\end{aligned}\tag{1}$$

In Equation (1), the first two rows represent the momentum conservation equations and the third row represents the mass conservation equation. In order to use CNN to solve the Navier-Stokes equations, we first need to divide the domain into grids and use the coordinates of the grid points as the input to the CNN. The output of the CNN is set to the values of the unknown function at the corresponding coordinates. And then, we need to embed the equations into the loss function, which is why we call it physics-informed CNN (PICNN). More details will be introduced in the following subsections.

3.2 Second-order upwind difference scheme

In order to embed the Navier-Stokes equations into the loss function of PICNN, we first need to discretize the equations. This subsection presents the procedure for discretizing the Navier-Stokes equations using the second-order upwind dif-

DOI: 10.52396/JUSTC-2022-0174
JUSTC, **2023**, 53(X):

ference scheme derived by FVM. It is worth mentioning that Equation (1) is the conserved form of the Navier-Stokes equations and FVM is more suitable than FDM for discretizing equations in conserved form. Detailed analysis of the limitations of existing FDM-based methods and central difference scheme is presented in the appendix A.1 and A.2.

First, we discretize the momentum equation in the horizontal direction (i.e., the first row in Equation (1)). As shown in Figure. 1, the upper, lower, left, and right boundaries of the control volume of point P are n, s, w, and e, respectively. By integrating the equation over the control volume of P, we can obtain Equation (2). Further expanding Equation (2), we can get Equation (3).

$$\int_s^n \int_w^e [\frac{\partial(\rho uu)}{\partial x} + \frac{\partial(\rho vu)}{\partial y}]dxdy = \int_s^n \int_w^e [-\frac{\partial p}{\partial x} + \frac{\partial}{\partial x}(\mu\frac{\partial u}{\partial x}) + \frac{\partial}{\partial y}(\mu\frac{\partial u}{\partial y})]dxdy \tag{2}$$

$$(\rho u)_e \Delta y u_e - (\rho u)_w \Delta y u_w + (\rho v)_n \Delta x u_n - (\rho v)_s \Delta x u_s = -\Delta y(p|_w^e) + \mu\Delta y(\frac{\partial u}{\partial x}|_w^e) + \mu\Delta x(\frac{\partial u}{\partial y}|_s^n) \tag{3}$$

We define the flow rate on interface e as $F_e = (\rho u)_e \Delta y$, and similarly the flow rate on interface w, n, and s are defined as $F_w = (\rho u)_w \Delta y$, $F_n = (\rho v)_n \Delta x$, and $F_s = (\rho v)_s \Delta x$, respectively. And we define the diffusion conductance on interface e, w, n, and s as $D_e = \frac{\mu\Delta y}{(\delta x)_e}$, $D_w = \frac{\mu\Delta y}{(\delta x)_w}$, $D_n = \frac{\mu\Delta x}{(\delta y)_n}$, and $D_s = \frac{\mu\Delta x}{(\delta y)_s}$, respectively [30]. In this work, we divide the domain equally so that $(\delta x)_e = (\delta x)_w = (\delta y)_n = (\delta y)_s = \Delta x = \Delta y$. Thus we can get $D_e = D_w = D_n = D_s$, which we write as D for brevity. We discretize the $\frac{\partial u}{\partial x}$ and $\frac{\partial u}{\partial y}$ at the interface using linear profile, i.e., discretize them using central difference. And we use the mean value of the pressure of the adjacent grid points to the left and right to represent the pressure value p_e and p_w on the interface. By processing Equation (3) as described above, we can obtain Equation (4).

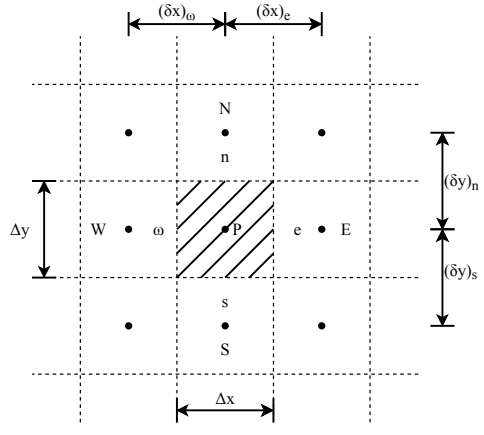


Fig. 1. Domain meshing of two-dimensional equations.

$$F_e u_e - F_w u_w + F_n u_n - F_s u_s = \Delta y(\frac{p_W - p_E}{2}) + D(u_E + u_W + u_N + u_S - 4u_P) \tag{4}$$

The value of velocity u_e at the interface e in Equation (4) is shown as Equation (5). The value of u_w, u_n, and u_s are similar to u_e, that is, take two points in the upwind direction and combine them according to the coefficients in Equation (5). This is where the second-order upwind difference scheme comes in.

$$u_e = \begin{cases} 1.5u_P - 0.5u_W, & u_e > 0 \\ 1.5u_E - 0.5u_{EE}, & u_e < 0 \end{cases} \tag{5}$$

By bringing Equation (5) into Equation (4), we can replace all the velocities at boundaries in Equation (4) with the velocities at the grid points, which are the predicted values of PICNN. Note that the velocity at boundary in Equation (5) vary with its sign. To simplify the equation, we use a compact form as shown in Equation (6).

$$\begin{aligned} &F_e u_e = (1.5u_P - 0.5u_W)\mathcal{M}(F_e) - (1.5u_E - 0.5u_{EE})\mathcal{M}(-F_e) \\ &define: \mathcal{M}(F) = max(F, 0) \end{aligned} \tag{6}$$

By substituting the compact form given by Equation (6) into Equation (4), we can obtain the 2nd-order upwind scheme of the momentum equation in the horizontal direction, as shown in Equation (7), where $P_\Delta = F/D$ and δx represents the grid size. P_Δ is the Peclet number [30], which represents the relative magnitude of convection and diffusion. Peclet number can help us analyze the performance of different discrete schemes.

$$\begin{aligned} &(1 + 0.5\mathcal{M}(P_{\Delta_e}) + 1.5\mathcal{M}(P_{\Delta_w}))u_W + (1 + 1.5\mathcal{M}(-P_{\Delta_e}) + \\ &0.5\mathcal{M}(-P_{\Delta_w}))u_E + (1 + 0.5\mathcal{M}(P_{\Delta_n}) + 1.5\mathcal{M}(P_{\Delta_s}))u_S + \\ &(1 + 1.5\mathcal{M}(-P_{\Delta_n}) + 0.5\mathcal{M}(-P_{\Delta_s}))u_N - 0.5(\mathcal{M}(P_{\Delta_w})u_{WW} + \\ &\mathcal{M}(-P_{\Delta_e})u_{EE} + \mathcal{M}(P_{\Delta_s})u_{SS} + \mathcal{M}(-P_{\Delta_n})u_{NN}) - \\ &(1.5(\mathcal{M}(P_{\Delta_e}) + \mathcal{M}(-P_{\Delta_w}) + \mathcal{M}(P_{\Delta_n}) + \mathcal{M}(-P_{\Delta_s})) + 4)u_P + \\ &\frac{(p_W - p_E)\delta x}{2\mu} = 0 \end{aligned} \tag{7}$$

The 2nd-order upwind scheme of the momentum equation in the vertical direction (i.e., the second row in Equation (1)) is similar to the equation in the horizontal direction and is detailed in B.2. We discretize the mass equation (i.e., the third row in Equation (1)) using the central difference scheme because it does not contain convection terms. The derivation of the discrete scheme of the mass equation is presented in B.1. And we give the derivation of the central difference scheme in B.3 for comparison.

3.3 Construction of loss function

Equation (7) gives the discrete scheme of the Navier-Stokes equations at grid point P, which is formed by combining the predicted values at point P and its neighbors. When the predicted value of PICNN at point P is close to its real value, the value calculated by the left side of Equation (7) should be close to 0. We set the left side of Equation (7) as the function $\mathcal{F}_u$, and $\mathcal{F}_u(P)$ represents the value of the function $\mathcal{F}_u$ at point P, that is, the degree to which the predicted value of PICNN satisfies the first equation in the Navier-Stokes equations.

DOI: 10.52396/JUSTC-2022-0174
JUSTC, **2023**, 53(X):

Similarly, we set the left side of the discrete scheme of the vertical direction equation (See Equation (19)) to be the function $\mathcal{F}_v$, and the left side of the discrete scheme of the mass equation (See Equation (18)) to be the function $\mathcal{F}_{mass}$.

$$\mathcal{L}_u = \frac{1}{n}\sum_{P_i\in\Omega}\mathcal{F}_u(P_i)^2, \mathcal{L}_v = \frac{1}{n}\sum_{P_i\in\Omega}\mathcal{F}_v(P_i)^2,$$
$$\mathcal{L}_{mass} = \frac{1}{n}\sum_{P_i\in\Omega}\mathcal{F}_{mass}(P_i)^2 \mathcal{L} = \mathcal{L}_u + \mathcal{L}_v + \mathcal{L}_{mass} \quad (8)$$

The loss function consists of three parts, which respectively represent the degree of satisfaction of the predicted value to the three equations in the Navier-Stokes equations. For each part, we compute the $\mathcal{F}$ values at all grid points and compute the MSE value, as shown in the first row of Equation (8), where P_i represents the grid point, Ω represents the definition domain, and n represents the number of grid points. Because the $\mathcal{F}(P_i)$ is compared with 0, we abbreviate $(\mathcal{F}(P_i)-0)^2$ as $\mathcal{F}(P_i)^2$. Finally, we set the total loss function $\mathcal{L}$ to be the sum of the three partial loss functions, as shown in the second row of Equation (8).

3.4 Boundary encoding

The solutions of the steady-state PDEs are determined by both the equations and the boundary conditions. A Dirichlet boundary condition gives the value of the solution at the boundary, and a Neumann boundary condition gives the derivative or partial derivative of the solution at the boundary. In this work, we adopt the encoding method of boundary conditions proposed by Gao et al [9]. As shown in Figure. 2 (left), Dirichlet boundary conditions can be imposed by applying constant padding, which is independent of the internal field and does not vary during each iteration. For Neumann boundary conditions, we can use the finite difference method to calculate the values at the boundary points from the predicted value of the interior points adjacent to the boundary points. Because the Neumann boundary conditions depend on the internal field, the padding values are different at each iteration. By hard imposing the boundary conditions, we can guarantee that they are enforced during training. Therefore, the PICNN model can be trained without any labeled data when the boundary conditions are determined.

$$\frac{\partial v}{\partial y} = 0 \quad \frac{v_{boundary} - v_{internal}}{\Delta y} = 0 \quad (9)$$

The first row of Equation (9) is the commonly used fluid boundary condition at the outlet, which is a Neumann boundary condition. By using the FDM, we can obtain the numerical relationship of the boundary point and its adjacent interior point, as shown in the second row of Equation (9). Further, we can obtain $v_{boundary} = v_{internal}$, which we can use for boundary padding.

3.5 Model training and predicting

Figure. 3 illustrates our PICNN architecture for solving Navier-Stokes equations. The input of the model is image-like data of 2 channels, which are composed of x-coordinates and y-coordinates respectively. Considering the order of magnitude difference in the values of different variables predicted by the model (e.g., streamwise velocity component can be orders of magnitude greater than spanwise velocity component), we use sub-CNNs to predict them separately. Reference [16, 31] demonstrate the superiority of using subnets for multivariate regression in data-driven and data-free scenarios. We use three sub-CNNs to predict velocity components and pressure (u, v, and p), each with the same four-layer convolutional network structure. Each layer of the sub-CNN has trainable filters with the kernel size of 5×5, and 2D convolutional operations with padding of 2 and stride of 1. It is important to emphasize that the use of sub-CNNs is not to avoid the use of normalization. If necessary, sub-CNNs and normalization can be used at the same time. In our cases, which are represented in Section 4, the input data range is between 0-1, so we did not use normalization.

For the first three convolutional layers, each layer is followed by a Relu activation function and a batch normalization layer. The last convolutional layer is followed by an upsampling layer to expand the final output to the desired size.

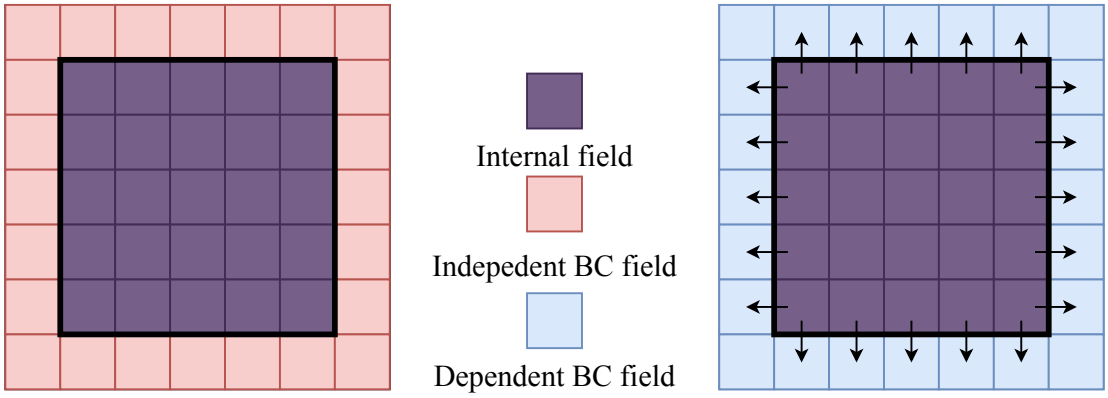


Fig. 2. Hard-imposition of boundary conditions (BC). The left side represents the Dirichlet boundary conditions, the right side represents the Neumann boundary conditions.

DOI: 10.52396/JUSTC-2022-0174
JUSTC, **2023**, 53(X):

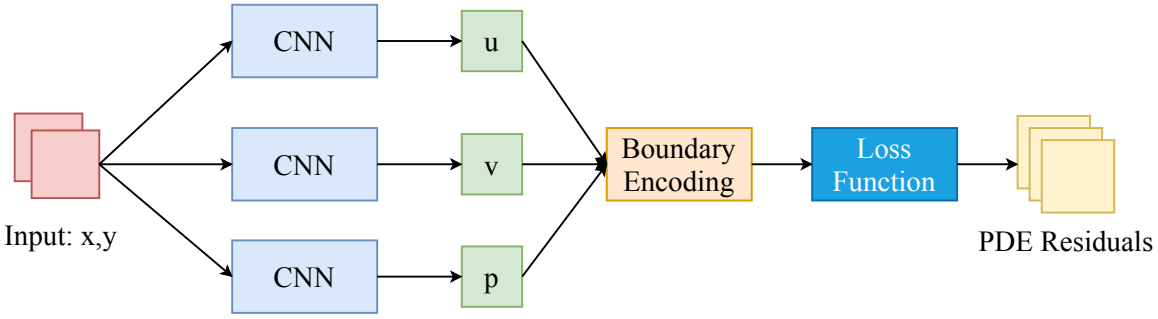


Fig. 3. The PICNN architecture for Navier-Stokes equations.

The variables predicted by the sub-CNNs are used to calculate the loss function after boundary encoding. After minimizing the PDE residuals, the prediction from the PICNN can approximate the true values.

The predicting process is similar to the training process, taking the coordinate values and passing them through the already trained sub-CNNs. And the final predictions can be obtained after boundary encoding the outputs of the sub-CNNs.

4 Experiments

Datasets and evaluation metric. We evaluate the accuracy and physical plausibility of our proposed methods on steady-state Navier-Stokes equations under different scenarios, including flow through a cavity, convective heat transfer, and lid-driven cavity flow. We use the numerical solutions calculated by COMSOL, a commercial numerical calculation software, as the groundtruth for comparison. The relative error metric is defined as $Error = \|\tilde{u} - u\|_{L2} / \|\tilde{u}\|_{L2}$, where $\tilde{u}$ is the numerical solution and u is the predicted solution.

Baselines. The proposed method (we abbreviate it as SUS-FVM) is compared to the CD-FVM method, which is a PICNN model with a central difference scheme derived from FVM to build the loss function. We also compare our proposed method with existing convolutional filter-based PICNN methods [9], including methods based on FDM-derived central difference scheme (CD-FDM) and second-order upwind difference scheme (SUS-FDM), to demonstrate that our method achieves higher prediction accuracy by using FVM. In addition, we also compare the PINN method, which is a MLP model based on automatic differentiation.

Implementation. We train all the models using the Adam optimizer with 200k epochs. The learning rate is set as 0.01 and is reduced by 50% every 50k epochs. We use the kaiming normalization to initialize the weights of convolutional layers. Other details for training can be found in the following subsections. In order to demonstrate the advantages of our proposed method over the baselines introduced above, we only optimize the loss function while keeping the network structure unchanged for a fair comparison. The codes and datasets for this work are available at https://github.com/USTC-ADA/Physically-Plausible-and-Conservative-Solutions-to-NS-Equations-Using-PICNN.

4.1 Flow through a cavity

We choose the spatial domain size as $[0,1]\times[0,1]$ and divide it into 50×50 ($\delta x = 0.02$) grids evenly. We set the inlet on the left side of the domain and the outlet on the upper side. The lengths of inlet and outlet are both set to $d = 0.3$. The remaining boundaries are set as no-slip walls on which the velocity is 0. The fluid at the inlet is set to a constant velocity of $init\_u = 1$ and the pressure at the outlet is 0. The velocity at the outlet meets the Neumann boundary condition $\frac{\partial v}{\partial y} = 0$. We select Navier-Stokes equations with Reynolds numbers (Re) of 30,150, and 300 for evaluation. The Reynolds number is used to describe the flow of fluids, defined by $Re = \rho(init\_u)d/\mu$, where $\rho = 1$, $init\_u = 1$, $d = 3$, and $\mu = 0.01, 0.002, 0.001$ for $Re = 30, 150, 300$, respectively. We change the Reynolds number by adjusting the viscosity μ of the fluid. As the Reynolds number increases, the convection becomes stronger.

Table. 1 shows the relative error of predicted pressure and velocity ($\sqrt{u^2+v^2}$) for the Navier-Stokes equations. We can find that as the Reynolds number increases from $Re = 150$ to $Re = 300$, the relative error of prediction for the CD-FVM increases dramatically, while the relative error of prediction for the SUS-FVM increases slightly. When the Re is low ($Re = 30$), CD-FVM has a higher prediction accuracy. And when the Re is high ($Re = 300$), the SUS-FVM scheme we use shows higher accuracy in the prediction of velocity and pressure.

Table 1. Relative errors of predictions for the flow through a cavity.

		CD-FVM	SUS-FVM
$Re = 30$	pressure	**0.0478**	0.0675
	velocity	**0.0318**	0.0393
$Re = 150$	pressure	**0.0456**	0.0855
	velocity	0.0404	**0.0389**
$Re = 300$	pressure	0.0967	**0.0839**
	velocity	0.1280	**0.0486**

DOI: 10.52396/JUSTC-2022-0174
JUSTC, **2023**, 53(X):

Figure. 4 shows the comparison of solutions between CD-FVM and SUS-FVM for the flow through a cavity. The fluid velocity is close to 0 in most areas of the domain, and only close to 1 on the way from the inlet to the outlet. SUS-FVM predicts the velocities well and has a higher prediction accuracy and better physical plausibility in the regions we care about (inlet, outlet, and pathways between them). In order to better illustrate the good prediction performance of SUS-FVM in local areas, we select a section parallel to the outlet ($y = 0.96$) and a section perpendicular to the outlet ($x = 0.50$) to observe the velocity distribution on them, as shown in Figure. 5. From Figure. 5, we can find that the predicted solutions obtained using SUS-FVM can perfectly capture the changes at locations with large velocity gradients. Compared to SUS-FVM, the solutions obtained using CD-FVM exhibit strong oscillations near the outlet, which extend vertically up to the position of $y = 0.5$ (Figure. 5 (b)). The predicted solutions from $x = 0.45$ to $x = 0.65$ in Figure. 5 (a) more clearly show the difference between the solutions obtained using CD-FVM and the real values. In practice, we cannot tolerate the oscillating solutions, and the SUS-FVM scheme we use effectively reduces the oscillation of the predicted solutions, making PICNN's prediction of the Navier-Stokes equations reliable.

Advantage of conservative schemes. Both CD-FDM and SUS-FDM use discrete schemes derived from non-conservative equations. The method we proposed and the CD-FVM use the discrete schemes derived from conservative equations. All the schemes are tested on dataset $Re = 30$ and $Re = 300$. From Table 2 we can find that the scheme derived from the conservative equations gives better predictions on both pressure and velocity than those derived from or using the non-conservative equations. When $Re = 30$, the relative errors for CD-FDM and SUS-FDM are an order of magnitude larger than those for CD-FVM and SUS-FVM, respectively. When $Re = 300$, the CD-FDM and SUS-FDM methods even diverse. The experimental results illustrate the advantages of deriving discrete schemes with conservation properties in prediction accuracy.

Comparison with PINN methods. We contrast the proposed method with DeepXDE [32] as shown in the Table. 3. DeepXDE is a library for scientific machine learning and physics-informed learning, which includes the PINN method and integrates various strategies to improve the accuracy of PINN, including residual-based adaptive sampling [33], gradient-enhanced PINN [34], and PINN with multi-scale Fourier features [35].

We train the DeepXDE model with two data volumes separately. One uses a small amount of data (DeepXDE-s), the amount of which is the same as the amount of training data used by the SUS-FVM model, i.e. 49×49 interior points and 200 boundary points. The other uses a large amount of data (DeepXDE-l), that is, 50000 interior points and 5000 boundary points. From Table. 3, we can find that the prediction accuracy of DeepXDE-l is higher than that of DeepXDE-s, thanks to its amount of training data. However, the prediction error of SUS-FVM is an order of magnitude smaller than that of DeepXDE-l and DeepXDE-s. The experimental result shows that although a larger amount of data increases the generalization of PINN models, limited by the fully-connected

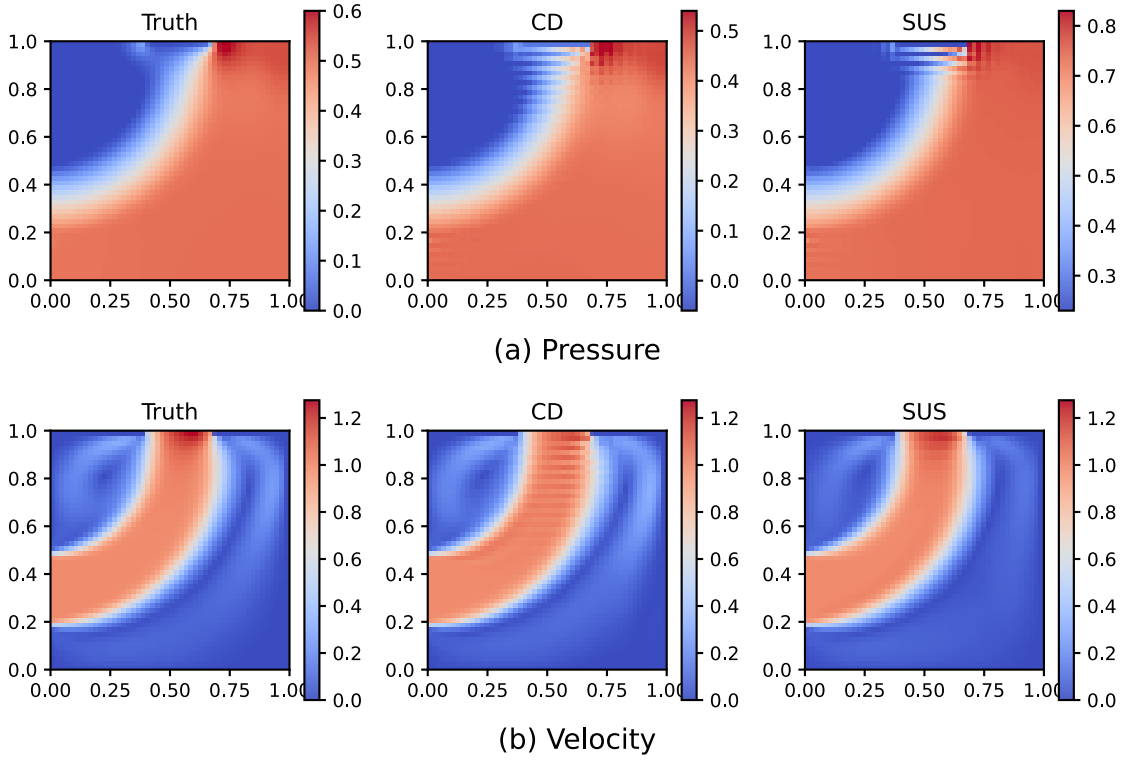


Fig. 4. Comparison of solutions between CD-FVM and SUS-FVM for flow through a cavity. (Re = 300)

DOI: 10.52396/JUSTC-2022-0174
JUSTC, **2023**, 53(X):

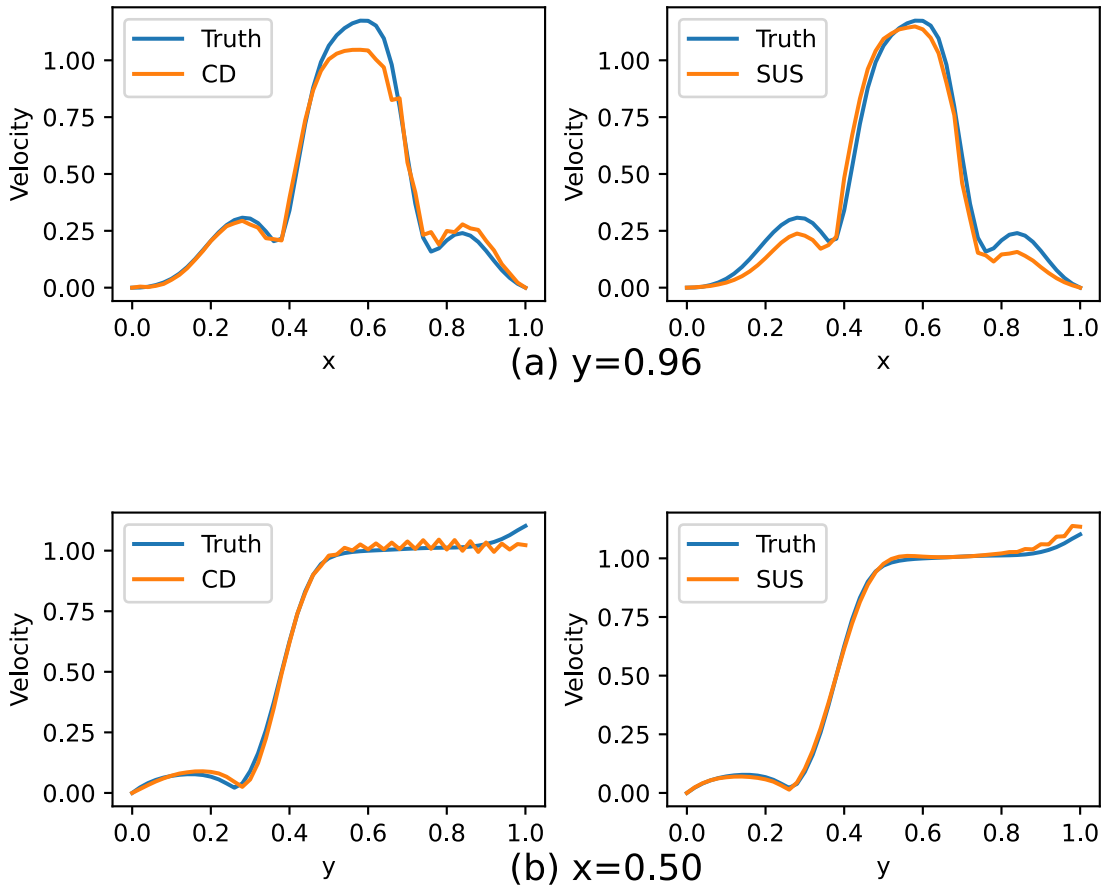


Fig. 5. Comparison of velocity accuracy on section $y=0.96$ and section $x = 0.50$. ($Re = 300$)

Table 2. Relative errors of predictions for conservative and non-conservative schemes of the Navier-Stokes equations.

		CD-FVM	CD-FDM	SUS-FVM	SUS-FDM
$Re = 30$	pressure	**0.0478**	0.1392	0.0675	0.7760
	velocity	**0.0318**	0.1078	0.0393	0.3281
$Re = 300$	pressure	0.0967	-	**0.0839**	-
	velocity	0.1280	-	**0.0486**	-

Dash "-" means divergence and error cannot be calculated.

Table 3. Relative errors of predictions for DeepXDE and SUS-FVM for the flow through a cavity.

		DeepXDE-s	DeepXDE-l	SUS-FVM
$Re = 30$	pressure	0.4420	0.3588	**0.0675**
	velocity	0.2233	0.1715	**0.0393**
$Re = 150$	pressure	0.9956	0.5955	**0.0855**
	velocity	0.6086	0.4062	**0.0389**
$Re = 300$	pressure	1.5442	0.6548	**0.0839**
	velocity	0.7859	0.4522	**0.0486**

neural networks and the soft imposition of boundary conditions, the prediction accuracy of PINN is not as good as that of PICNN.

4.2 Convective heat transfer

Compared with the Navier-Stokes equations, the convective heat transfer problems add a temperature equation, as shown in Equation (10), where T represents the temperature, C_p represents the heat capacity at constant pressure, k represents the thermal conductivity and ρ represents the density of the medium. We set the inlet of the fluid to the left side of the domain and the outlet to the right side. The velocities u and v are calculated by commercial software on the Navier-Stokes equations with $Re = 500$. By varying the value of thermal conductivity k, we obtain different datasets. For temperature, we choose the Dirichlet boundary conditions and set the temperature of the left wall to 293.15 and the temperature of the remaining walls to 333.15.

$$\frac{\partial(\rho C_p u T)}{\partial x} + \frac{\partial(\rho C_p v T)}{\partial y} = \frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(k\frac{\partial T}{\partial y}\right) \tag{10}$$

Table. 4 shows the relative errors of predicted temperature for the convective heat transfer problems. Since the maxim-

DOI: 10.52396/JUSTC-2022-0174
JUSTC, **2023**, 53(X):

Table 4. Relative errors of predictions (T) for the convective heat transfer problems.

	CD-FVM	SUS-FVM
Data1: $k = 10$	**0.0023**	0.0027
Data2: $k = 1$	0.0196	**0.0042**
Data3: $k = 0.5$	0.0338	**0.0076**
Data4: $k = 0.1$	0.0922	**0.0293**

$Re = 500$, $\rho = 1$, $C_p = 1000$

um velocity, in this case, is around 1, we can infer that when $k = 10$ (Data1), the grid Peclet number $P_\Delta = 2$. As k gradually decreases to 0.1 (Data2 to Data4), P_Δ gradually increases to 200. As P_Δ increases, the relative error of the prediction of the CD-FVM increases dramatically, while the relative error of the SUS-FVM increases slowly, which is an order of magnitude smaller than that of the CD-FVM. Using Data3 as an example, as shown in Figure. 6, we can see that using CD-FVM for prediction yields solutions with strong oscillations near the outlet. We select a section perpendicular to the outlet ($y = 0.64$) in Figure. 6 to further observe the temperature distribution on it, as shown in Figure. 7. The strongly oscillating solutions obtained using CD-FVM completely lose the physical plausibility, while the solutions obtained using SUS-FVM fit the real solutions precisely.

4.3 Lid-driven cavity flow

Lid-driven cavity flow is a common benchmarking case in computational fluid dynamics. We use it to show the performance of our proposed method at high Reynolds numbers. We set the horizontal movement speed of the lip (upper boundary of the domain) to 1, and set the other walls to be no-slip walls. By adjusting the viscosity μ, we obtained three datasets with Reynolds numbers of 100, 500, and 1000, respectively.

As shown in Table. 5, SUS-FVM outperforms CD-FVM in prediction on all three datasets. When the Reynolds number is low ($Re = 100$), the vortex in the cavity is not obvious, which is shown in Figure. 5, and both CD-FVM and SUS-FVM perform well in velocity prediction. When the Reynolds number is high ($Re = 500, 1000$), the vortex in the cavity is obvious, which makes the prediction performance of CD-FVM and

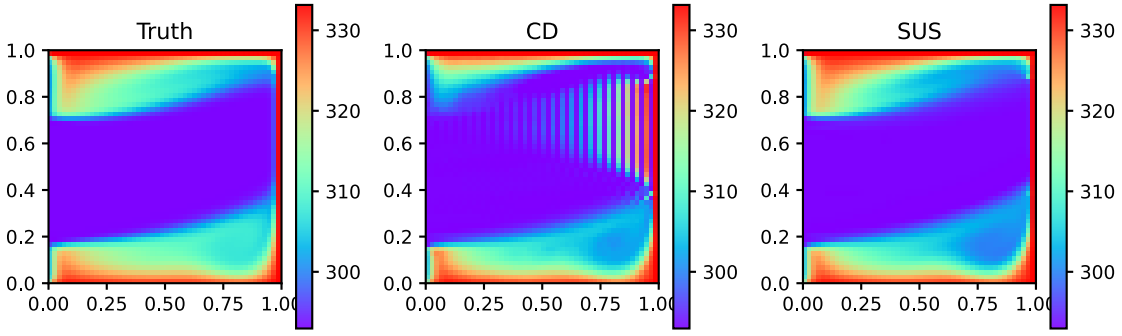


Fig. 6. Comparison of solution accuracy between SUS-FVM and CD-FVM schemes for the convective heat transfer problems. (Data3)

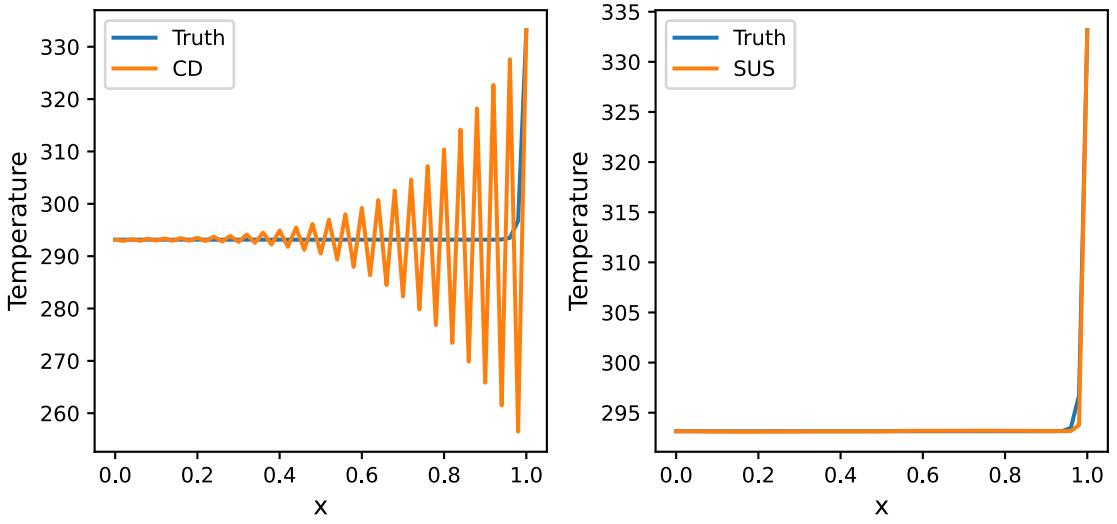


Fig. 7. Comparison of solution accuracy on section $y = 0.64$ between SUS-FVM and CD-FVM schemes. (Data3)

DOI: 10.52396/JUSTC-2022-0174
JUSTC, **2023**, 53(X):

Table 5. Relative errors of predictions for the lid-driven cavity flow.

		CD-FVM	SUS-FVM
Re = 100	pressure	0.4277	**0.3986**
	velocity	0.1082	**0.1074**
Re = 500	pressure	0.2873	**0.2125**
	velocity	0.1644	**0.1317**
Re = 1000	pressure	0.3737	**0.3395**
	velocity	0.2936	**0.2768**

SUS-FVM for velocity decrease. But it can be found that the prediction accuracy of SUS-FVM decreases more slowly than CD-FVM. The results show that our proposed method is still effective in predicting the Navier-Stokes equations in the case of high Reynolds numbers and outperforms the existing central difference based methods.

4.4 Discussion

In this work, our main study is solving the incompressible Navier-Stokes equations under laminar flow, i.e., Navier-Stokes equations under low Reynolds number, using physics-informed CNN. For turbulent flow scenarios, i.e., high Reynolds number, the prediction difficulty of Navier-Stokes equations is much higher than that of laminar flow scenarios. We need to optimize the neural network structure so that the model can predict the local details of turbulent flow well, which is our future research direction.

Furthermore, for compressible Navier-Stokes equations such as shock wave, our method is currently not suitable to solve such problems. There is a huge difference in fluid flow between shock wave and laminar flow. To better simulate the shock wave, it is necessary to accurately predict the fluid properties near the obstacle. However, the PICNN method we use is currently unable to describe the scene with obstacles in the domain of definition. It is a challenging problem and deserves further investigation in our future work.

5 Conclusions

In this study, we focus on solving Navier-Stokes equations using PICNN. Aiming at the problem that PICNN will generate oscillating predictions when solving the Navier-Stokes equations, we propose to use the FVM-based second order upwind difference scheme to construct the loss function. We give formulation derivations and conduct experiments on the Navier-Stokes equations under different scenarios to demonstrate that our proposed method effectively improves the physical plausibility and accuracy of the predictions.

Our proposed method is not limited to convolutional neural networks but is also applicable to other network architectures using physically constrained loss functions. Because of the introduction of grid-scale, our method can solve PDEs under variable-step rectangular grid division. Furthermore, our method can be extended to graph neural networks to solve irregular spatial domains and grids. Besides, the proposed method can be used for the loss function calculation of PDEs solving models for different purposes, such as data-driven PDEs discovery models and boundary surrogate models, to improve the solution accuracy and physical plausibility. Finally, this work focuses on solving steady-state PDEs. For solving dynamic PDEs, we can further derive their discrete

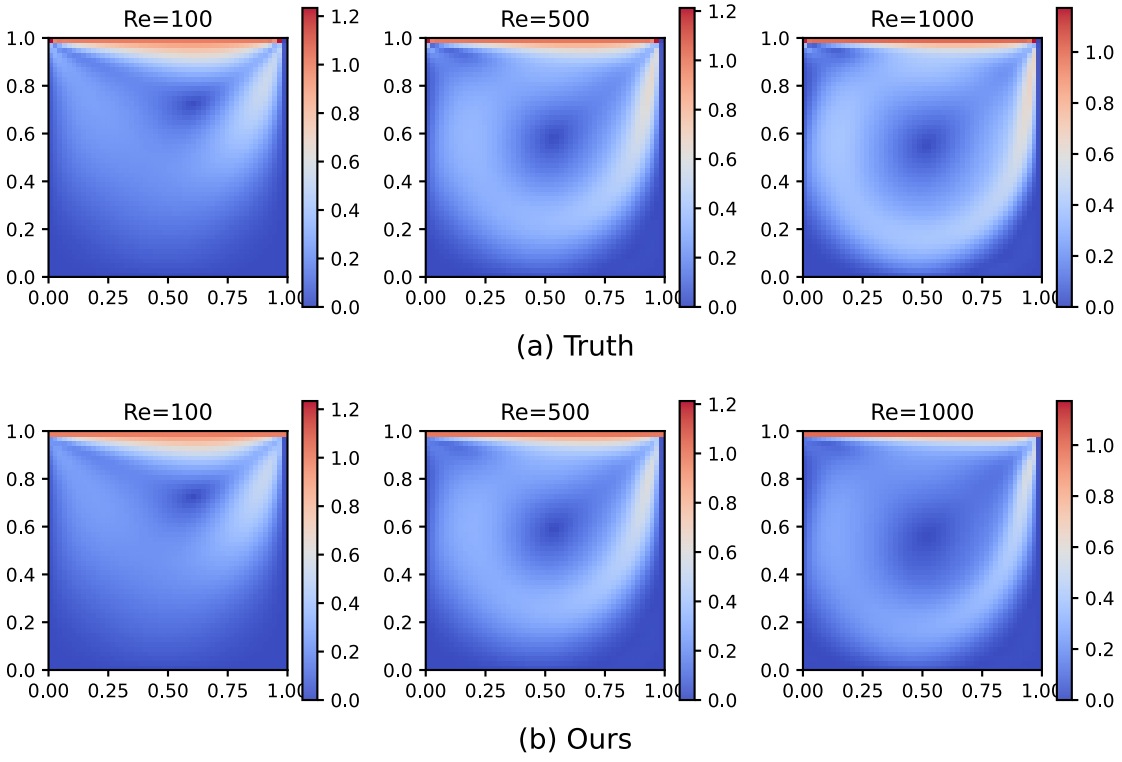


Fig. 8. Comparison of predicted and true velocities for the lid-driven cavity flow

DOI: 10.52396/JUSTC-2022-0174
JUSTC, **2023**, 53(X):

schemes and leverage suitable network architectures (e.g., LSTM) in future work.

Acknowledgments

This work is funded by Youth Innovation Promotion Association of Chinese Academy of Sciences (CAS).

Conflict of Interest

The authors declare that they have no conflict of interest.

Biographies

Jianfeng Li is a graduate student in the School of Computer Science and Technology, University of Science and Technology of China. His research interests include machine learning and scientific computing.

Jingwei Sun is currently an associate researcher in the School of Computer Science and Technology, University of Science and Technology of China. His research interests include high performance computing, parallel algorithms, performance profiling, and modeling.

References

[1] Wang N, Chang H, Zhang D, et al. Efficient well placement optimization based on theory-guided convolutional neural network. *Journal of Petroleum Science and Engineering,* **2022**, *208*: 109545.

[2] Raissi M, Perdikaris P, Karniadakis G E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics,* **2019**, *378*: 686–707.

[3] Wang N, Chang H, Zhang D. Efficient uncertainty quantification and data assimilation via theory-guided convolutional neural network. *SPE Journal,* **2021**, *26* (06): 4128–4156.

[4] Meng C, Seo S, Cao D, et al. When physics meets machine learning: A survey of physics-informed machine learning. arXiv: 2203.16797, **2022**.

[5] Baydin A G, Pearlmutter B A, Radul A A, et al. Automatic differentiation in machine learning: A survey. *Journal of Marchine Learning Research,* **2017**, *18* (1): 5595–5637.

[6] Geneva N, Zabaras N. Modeling the dynamics of PDE systems with physics-constrained deep auto-regressive networks. *Journal of Computational Physics,* **2020**, *403*: 109056.

[7] Winovich N, Ramani K, Lin G. ConvPDE-UQ: Convolutional neural networks with quantified uncertainty for heterogeneous elliptic partial differential equations on varied domains. *Journal of Computational Physics,* **2019**, *394*: 263–279.

[8] Ren P, Rao C, Liu Y, et al. Physics-informed deep super-resolution for spatiotemporal data. arXiv: 2208.01462, **2022.**.

[9] Gao H, Sun L, Wang J X. PhyGeoNet: Physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state PDEs on irregular domain. *Journal of Computational Physics,* **2021**, *428*: 110079.

[10] Ren P, Rao C, Liu Y, et al. PhyCRNet: Physics-informed convolutional-recurrent network for solving spatiotemporal PDEs. *Computer Methods in Applied Mechanics and Engineering,* **2022**, *389*: 114399.

[11] Jaluria Y, Torrance K E. Computational Heat Transfer. Heidelberg: Springer Berlin, **2017.**.

[12] Zhang R, Liu Y, Sun H. Physics-guided convolutional neural network (PhyCNN) for data-driven seismic response modeling. *Engineering Structures,* **2020**, *215*: 110704.

[13] Sun L, Wang J X. Physics-constrained Bayesian neural network for fluid flow reconstruction with sparse and noisy data. *Theoretical and Applied Mechanics Letters,* **2020**, *10*: 161–169.

[14] Wang Y, Sun H, Sun G. DSP-PIGAN: A precision-consistency machine learning algorithm for solving partial differential equations. In: 2021 13th International Conference on Machine Learning and Computing. New York: ACM, **2021**: 21–26.

[15] Zhu Y, Zabaras N, Koutsourelakis P S, et al. Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. *Journal of Computational Physics,* **2019**, *394*: 56–81.

[16] Sun L, Gao H, Pan S, et al. Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data. *Computer Methods in Applied Mechanics and Engineering,* **2020**, *361*: 112732.

[17] Yang Y, Perdikaris P. Adversarial uncertainty quantification in physics-informed neural networks. *Journal of Computational Physics,* **2019**, *394*: 136–152.

[18] Meng X, Li Z, Zhang D, et al. PPINN: Parareal physics-informed neural network for time-dependent PDEs. *Computer Methods in Applied Mechanics and Engineering,* **2020**, *370*: 113250.

[19] Pang G, Lu L, Karniadakis G E. fPINNs: Fractional physics-informed neural networks. *SIAM Journal on Scientific Computing,* **2019**, *41* (4): A2603–A2626.

[20] Kharazmi E, Zhang Z, Karniadakis G E. Variational physics-informed neural networks for solving partial differential equations. arXiv: 1912.00873, **2019**.

[21] Yang L, Meng X, Karniadakis G E. B-PINNs: Bayesian physics-informed neural networks for forward and inverse PDE problems with noisy data. *Journal of Computational Physics,* **2021**, *425*: 109913.

[22] Rao C, Liu Y. Three-dimensional convolutional neural network (3D-CNN) for heterogeneous material homogenization. *Computational Materials Science,* **2020**, *184*: 109850.

[23] Zhu Y, Zabaras N. Bayesian deep convolutional encoder-decoder networks for surrogate modeling and uncertainty quantification. *Journal of Computational Physics,* **2018**, *366*: 415–447.

[24] Kim B, Azevedo V C, Thuerey N, et al. Deep fluids: A generative network for parameterized fluid simulations. *Computer Graphics Forum,* **2019**, *38*: 59–70.

[25] Sharma R, Farimani A B, Gomes J, et al. Weakly-supervised deep learning of heat transport via physics informed loss. arXiv: 1807.11374, **2018**.

[26] Fukui K I, Tanaka J, Tomita T, et al. Physics-guided neural network with model discrepancy based on upper troposphere wind prediction. In: 2019 18th IEEE International Conference on Machine Learning and Applications (ICMLA). Boca Raton, USA: IEEE, **2019**: 414–419.

[27] Subramaniam A, Wong M L, Borker R D, et al. Turbulence enrichment using physics-informed generative adversarial networks. arXiv: 2003.01907, **2020**.

[28] Mohan A T, Lubbers N, Livescu D, et al. Embedding hard physical constraints in neural network coarse-graining of 3D turbulence. arXiv: 2002.00021, **2020**.

[29] Bar-Sinai Y, Hoyer S, Hickey J, et al. Learning data-driven discretizations for partial differential equations. *Proceedings of the National Academy of Sciences of the United States of America,* **2019**, *116*: 15344–15349.

[30] Tao W Q. Numerical Heat Transfer, second edtion. Xi'an: Xi'an Jiaotong University Press, **2001**.

[31] Guo L, Ye S, Han J, et al. SSR-VFD: Spatial super-resolution for vector field data analysis and visualization. In: 2020 IEEE Pacific Visualization Symposium (PacificVis). Tianjin, China: IEEE, **2020**: 71–80.

[32] Lu L, Meng X, Mao Z, et al. DeepXDE: A deep learning library for solving differential equations. *SIAM Review,* **2021**, *63*: 208–228.

[33] Wu C, Zhu M, Tan Q, et al. A comprehensive study of non-adaptive

DOI: 10.52396/JUSTC-2022-0174
JUSTC, **2023**, 53(X):

and residual-based adaptive sampling for physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, **2023**, *403*: 115671.
[34] Yu J, Lu L, Meng X, et al. Gradient-enhanced physics-informed neural networks for forward and inverse PDE problems. *Computer Methods in Applied Mechanics and Engineering*, **2022**, *393*: 114823.
[35] Wang S, Wang H, Perdikaris P. On the eigenvector bias of Fourier feature networks: From regression to solving multi-scale PDEs with physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, **2021**, *384*: 113938.

Appendix A Limitations of FDM-based central difference scheme

A.1 Limitations of central difference scheme

When using the central difference scheme to discretize the convection-diffusion equation, an oscillating solution may arise due to an excessively large space step or an excessively large flow velocity. This phenomenon is also called the instability of the discretization scheme. Instability is an inherent property of a discrete scheme, and it is affected by multiple factors, including fluid properties, flow velocity, and grid size. We use a one-dimensional steady-state convection-diffusion equation (11) with no source to illustrate the cause of the instability and its negative effects.

$$\frac{d(\rho u\phi)}{dx}=\frac{d}{dx}\left(\Gamma\frac{d\phi}{dx}\right),x\in[0,L] \tag{11}$$

We divide the domain $[0,L]$ into grids, which are actually line segments. Then we use the central difference scheme to discretize the convection term and the diffusion term, respectively, as shown in Equation (13). For simplicity, we integrate various factors that affect stability as the grid Peclet number P_Δ [], as shown in Equation (12), where δx represents the grid size. By combining Equation (13) and Equation (11), we can obtain a central difference based discrete equation (14), where $i-1$ and $i+1$ represent the left and right neighbors of i, respectively.

$$P_\Delta=\frac{\rho u\delta x}{\Gamma} \tag{12}$$

$$\begin{aligned}\frac{d\phi}{dx}&=\frac{\phi_{i+1}-\phi_{i-1}}{2\delta x}\\ \frac{d}{dx}\left(\frac{d\phi}{dx}\right)&=\frac{\phi_{i+1}+\phi_{i-1}-2\phi_i}{(\delta x)^2}\end{aligned} \tag{13}$$

$$\phi_i=\frac{(1-0.5P_\Delta)\phi_{i+1}+(1+0.5P_\Delta)\phi_{i-1}}{2} \tag{14}$$

From Equation (14) we can prove that ϕ_i is less than the values of its two neighbors if $0<\phi_{i-1}<\phi_{i+1}$ and $P_\Delta>2$, which is not possible for the case without source. The coefficients of ϕ_{i-1} and ϕ_{i+1} represent the influence of the ϕ at the adjacent point on the point i through convection and diffusion. When $P_\Delta>2$, the coefficient of ϕ_{i+1} is less than zero, causing the influence of ϕ_{i+1} to ϕ_i as negative, which is physically meaningless.

The stability of the central difference scheme is affected by P_Δ so it is called conditional stability. The conditional stability, as an inherent property of the scheme, can lead to oscillations of the solutions in specific cases in both numerical and deep learning methods. The cause of the conditional stability is the introduction of downstream information when discretizing the convection term using central difference, which introduces a negative term in the coefficient of ϕ_{i+1}. The Navier-Stokes equations are a special form of the convection-diffusion equation, so oscillations also occur when it is discretized using the central difference scheme.

A.2 Limitations of existing FDM-based scheme

Conservativeness is the guarantee of obtaining physically meaningful solutions, which is described as follows [30]:

If a discrete equation is summed in any finite space of the definition domain, and the obtained expression satisfies the relation of conservation of physical quantities in this region, then the discrete scheme has conservativeness property.

Discrete equations with conservativeness can produce more accurate and more physically plausible solutions [11]. To ensure the conservativeness of discrete equations, the following two conditions need to be satisfied. 1) The governing equations from which discrete equations are derived are conservative. 2) Each physical quantity (ϕ and physical properties) and the first derivative of ϕ are continuous on the same interface.

The existing FDM-based scheme violates the first condition to ensure conservativeness. For instance, the first row in Equation (15) is a conservative governing equation for the one-dimensional convection problem, while the second is a non-conservative governing equation. The FDM-based discrete schemes derived from the second row in Equation (15) are widely used in the PICNN models to calculate the parameters of the convolutional filters. However, the existing FDM-based method cannot deal with the first row in Equation (15), and thus cannot obtain discrete equations with conservation properties.

$$\begin{aligned}\frac{\partial\phi}{\partial t}+\frac{\partial(u\phi)}{\partial x}&=0\\ \frac{\partial\phi}{\partial t}+u\frac{\partial\phi}{\partial x}&=0\end{aligned} \tag{15}$$

Appendix B Derivation of second-order upwind and central difference schemes

B.1 Mass equation discretization

For the third row in Equation (1), we integrate it over the control volume of P and obtain Equation (16).

$$\int_s^n\int_w^e\left[\frac{\partial(\rho u)}{\partial x}+\frac{\partial(\rho v)}{\partial y}\right]dxdy=0 \tag{16}$$

We use the mean value of the u of the adjacent grid points to the left and right to represent u_e and u_w. And we use the mean value of the v of the adjacent grid points to the up and down to represent v_n and v_s. Finally, we can obtain the discrete scheme (17) of the third equation in Navier-Stokes

DOI: 10.52396/JUSTC-2022-0174
JUSTC, **2023**, 53(X):

equations.

$$\rho\Delta y\left(\frac{u_E-u_W}{2}\right)+\rho\Delta x\left(\frac{v_N-v_S}{2}\right)=0 \tag{17}$$

To keep the calculated value on the left side of Equation (17) on the same order of magnitude as Equation (7), we divide Equation (17) by D and get Equation (18).

$$\frac{\rho\delta x}{\mu}\left(\frac{u_E-u_W}{2}+\frac{v_N-v_S}{2}\right)=0 \tag{18}$$

B.2 2nd-order upwind scheme of the momentum equation in the vertical direction

Replace the horizontal velocity u in Equation (7) to the vertical velocity v and we can get the 2nd-order upwind scheme of the momentum equation in the vertical direction, as shown in Equation (19). The derivation process is exactly the same as shown in 3.2.

$$\begin{aligned}&(1+0.5\mathcal{M}(P_{\Delta_e})+1.5\mathcal{M}(P_{\Delta_w}))v_W+(1+1.5\mathcal{M}(-P_{\Delta_e})+\\&0.5\mathcal{M}(-P_{\Delta_w}))v_E+(1+0.5\mathcal{M}(P_{\Delta_n})+1.5\mathcal{M}(P_{\Delta_s}))v_S+\\&(1+1.5\mathcal{M}(-P_{\Delta_n})+0.5\mathcal{M}(-P_{\Delta_s}))v_N-0.5(\mathcal{M}(P_{\Delta_w})v_{WW}+\\&\mathcal{M}(-P_{\Delta_e})v_{EE}+\mathcal{M}(P_{\Delta_s})v_{SS}+\mathcal{M}(-P_{\Delta_n})v_{NN})-\\&(1.5(\mathcal{M}(P_{\Delta_e})+\mathcal{M}(-P_{\Delta_w})+\mathcal{M}(P_{\Delta_n})+\mathcal{M}(-P_{\Delta_s}))+4)v_P+\\&\frac{(p_S-p_N)\delta x}{2\mu}=0\end{aligned} \tag{19}$$

B.3 Derivation of central difference scheme

The difference between the central difference scheme and the second-order upwind difference scheme lies in the value of the velocity at the boundary. The central difference, as the name implies, takes the velocity at the boundary as the average value of the velocity of its adjacent grid points. Equation (20) shows the values of the velocities at the four boundaries of the control volume at point P. Bringing Equation (20) into Equation (4) and dividing it by D, we can organize the equation as shown in Equation (21), where $A(|P_\Delta|)$ is given by Equation (22).

$$\begin{cases}u_e=\dfrac{u_P+u_E}{2}\\ u_w=\dfrac{u_P+u_W}{2}\\ u_n=\dfrac{u_P+u_N}{2}\\ u_s=\dfrac{u_P+u_S}{2}\end{cases} \tag{20}$$

$$\begin{aligned}&a_Pu_P=a_Eu_E+a_Wu_W+a_Nu_N+a_Su_S+\frac{(p_W-p_E)\delta x}{2\mu}\\&a_E=A(|P_{\Delta_e}|)+\mathcal{M}(-P_{\Delta_e})\\&a_W=A(|P_{\Delta_w}|)+\mathcal{M}(P_{\Delta_w})\\&a_N=A(|P_{\Delta_n}|)+\mathcal{M}(-P_{\Delta_n})\\&a_S=A(|P_{\Delta_s}|)+\mathcal{M}(P_{\Delta_s})\\&a_P=a_E+a_W+a_N+a_S+(P_{\Delta_e}-P_{\Delta_w})+(P_{\Delta_n}-P_{\Delta_s})\end{aligned} \tag{21}$$

$$A(|P_\Delta|)=\begin{cases}1, & FUS\\ 1-0.5|P_\Delta|, & CD\end{cases} \tag{22}$$

Equation (21) is actually a general discrete scheme controlled by $A(|P_\Delta|)$. By modifying the value of $A(|P_\Delta|)$, we can get the central difference scheme (CD), the first-order upwind difference scheme (FUS), the power-law scheme (PLS), etc. In this work, we use the central difference scheme as the benchmark, that is, bring $A(|P_\Delta|)=1-0.5|P_\Delta|$ into Equation (21) to obtain the central difference discrete scheme.

$$a_Pv_P=a_Ev_E+a_Wv_W+a_Nv_N+a_Sv_S+\frac{(p_S-p_N)\delta x}{2\mu} \tag{23}$$

The central difference scheme of the momentum equation in the vertical direction (i.e., the second row in Equation (1)) is the same as the equation in the horizontal direction and is shown in Equation (23), where the coefficients are the same as those in Equation (21). And the central difference scheme of the mass equation is the same as Equation (18).

DOI: 10.52396/JUSTC-2022-0174
JUSTC, **2023**, 53(X):