# PHYSICALLY PLAUSIBLE AND CONSERVATIVE SOLUTIONS TO NAVIER-STOKES EQUATIONS USING PHYSICS-INFORMED CNNS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Physics-informed Neural Network (PINN) is an emerging approach for efficiently solving partial differential equations (PDEs) using neural networks. 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.

## 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 multi-scale/multiphysics features, e.g., optimization (Wang et al., 2022), inverse problem (Raissi et al., 2019), and uncertainty quantification (Wang et al., 2021a).

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 (Meng et al., 2022). 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. (Raissi et al., 2019) proposed physics-informed neural network (PINN) for solving PDEs. PINN uses the automatic differentiation technique (Baydin et al., 2018) 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 (Geneva & Zabaras, 2020)(Winovich et al., 2019).

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 (Gao et al., 2021)(Ren et al., 2022) 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 (Jaluria & Torrance, 2017). **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 (Raissi et al., 2019)(Zhang et al., 2020)(Sun & Wang, 2020)(Wang et al., 2021b) or even without any data (Zhu et al., 2019)(Sun et al., 2020)(Geneva & Zabaras, 2020). Based on PINN (Raissi et al., 2019), a series of PINN variants have been proposed to deal with more challenging problems, such as UQPINN (Yang & Perdikaris, 2019), PPINN (Meng et al., 2020), fPINN (Pang et al., 2019), vPINN (Kharazmi et al., 2019), and B-PINNs (Sun & Wang, 2020)(Yang et al., 2021).

Recent studies have shown that CNNs are suitable for solving large-scale, high-dimensional spatiotemporal problems due to their parameter-sharing features (Rao & Liu, 2020). For steady-state PDEs, Zhu et al. design a physics-informed convolutional encoder-decoder structure to learn solutions without using any labeled data (Zhu et al., 2019)(Zhu & Zabaras, 2018). Gao et al. design a geometric adaptation strategy (Gao et al., 2021), 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 (Geneva & Zabaras, 2020), which uses a PICNN-based deep autoregressive model to predict dynamical PDEs. Ren et al. design a surrogate model (Ren et al., 2022) 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 (Kim et al., 2019)(Sharma et al., 2018)(Fukui et al., 2019)(Subramaniam et al., 2020)(Mohan et al., 2020). Most of these methods use the FDM-based central difference scheme to discretize the convection term. Bar-Sinai et al. (Bar-Sinai et al., 2019) introduces the FVM to CNN, enabling the model to generate coefficients for approximating spatial derivatives. However, the method proposed in (Bar-Sinai et al., 2019) 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 $\rho$ is the fluid density, $u$ and $v$ are the flow velocity in the x-axis and y-axis directions, respectively, $\mu$ 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 (1) are convection terms, which have strong directionality and describe the directional motion of the fluid. The terms $\frac{\partial}{\partial x}(\mu\frac{\partial u}{\partial x}) + \frac{\partial}{\partial y}(\mu\frac{\partial u}{\partial y})$ and the $\frac{\partial}{\partial x}(\mu\frac{\partial v}{\partial x}) + \frac{\partial}{\partial y}(\mu\frac{\partial v}{\partial y})$ in the first two rows of (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 difference 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 equation in (1)). As shown in Fig. 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 (2). Further expanding (2), we can get (3).

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

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

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 (Tao, 2001). 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,

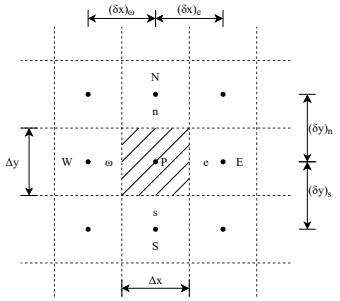


Figure 1: Domain meshing of two-dimensional equations.

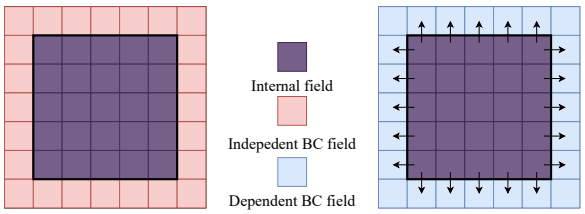


Figure 2: Hard-imposition of boundary conditions (BC). The left side represents the Dirichlet boundary conditions, the right side represents the Neumann boundary conditions.

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 (3) as described above, we can obtain (4).

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

The value of velocity $u_e$ at the interface $e$ in (4) is shown as (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 (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 (5) into (4), we can replace all the velocities at boundaries in (4) with the velocities at the grid points, which are the predicted values of PICNN. Note that the velocity at boundary in (5) vary with its sign. To simplify the equation, we use a compact form as shown in (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 using the compact form given by (6), we can expand (4) to (7).

$$\begin{aligned} &(1.5u_P - 0.5u_W)\mathcal{M}(F_e) - (1.5u_E - 0.5u_{EE})\mathcal{M}(-F_e) \\ -&(1.5u_W - 0.5u_{WW})\mathcal{M}(F_w) + (1.5u_P - 0.5u_E)\mathcal{M}(-F_w) \\ +&(1.5u_P - 0.5u_S)\mathcal{M}(F_n) - (1.5u_N - 0.5u_{NN})\mathcal{M}(-F_n) \\ -&(1.5u_S - 0.5u_{SS})\mathcal{M}(F_s) + (1.5u_P - 0.5u_N)\mathcal{M}(-F_s) \\ =&\Delta y(\frac{p_W - p_E}{2}) + D(u_E + u_W + u_N + u_S - 4u_P) \end{aligned} \tag{7}$$

Move all terms of (7) to one side and divide it by $D$, we can obtain the 2nd-order upwind scheme of the momentum equation in the horizontal direction, as shown in (8), where $P_\Delta = F/D$ and $\delta x$ represents the grid size. $P_\Delta$ is the Peclet number (Tao, 2001), which represents the relative magnitude of convection and diffusion. Peclet number can help us analyze the performance of different discrete schemes.

$$\begin{aligned} &(1 + 0.5\mathcal{M}(P_{\Delta_e}) + 1.5\mathcal{M}(P_{\Delta_w}))u_W + (1 + 1.5\mathcal{M}(-P_{\Delta_e}) + 0.5\mathcal{M}(-P_{\Delta_w}))u_E + \\ &(1 + 0.5\mathcal{M}(P_{\Delta_n}) + 1.5\mathcal{M}(P_{\Delta_s}))u_S + (1 + 1.5\mathcal{M}(-P_{\Delta_n}) + 0.5\mathcal{M}(-P_{\Delta_s}))u_N - \\ &0.5(\mathcal{M}(P_{\Delta_w})u_{WW} + \mathcal{M}(-P_{\Delta_e})u_{EE} + \mathcal{M}(P_{\Delta_s})u_{SS} + \mathcal{M}(-P_{\Delta_n})u_{NN}) - \\ &(1.5(\mathcal{M}(P_{\Delta_e}) + \mathcal{M}(-P_{\Delta_w}) + \mathcal{M}(P_{\Delta_n}) + \mathcal{M}(-P_{\Delta_s})) + 4)u_P + \frac{(p_W - p_E)\delta x}{2\mu} = 0 \end{aligned} \tag{8}$$

The 2nd-order upwind scheme of the momentum equation in the vertical direction (i.e., the second equation in (1)) is similar to the equation in the horizontal direction and is detailed in A.4. We discretize the mass equation (i.e., the third equation in (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 A.3. And we give the derivation of the central difference scheme in A.5 for comparison.

### 3.3 CONSTRUCTION OF LOSS FUNCTION

Equation (8) 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 (8) should be close to 0. We set the left side of (8) 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. Similarly, we set the left side of the discrete scheme of the vertical direction equation (See equation (20)) to be the function $\mathcal{F}_v$, and the left side of the discrete scheme of the mass equation (See equation (19)) to be the function $\mathcal{F}_{mass}$.

$$\begin{gathered}\mathcal{L}_u = \frac{1}{n}\sum_{P_i\in\Omega}\mathcal{F}_u(P_i)^2, \quad \mathcal{L}_v = \frac{1}{n}\sum_{P_i\in\Omega}\mathcal{F}_v(P_i)^2, \quad \mathcal{L}_{mass} = \frac{1}{n}\sum_{P_i\in\Omega}\mathcal{F}_{mass}(P_i)^2 \\ \mathcal{L} = \mathcal{L}_u + \mathcal{L}_v + \mathcal{L}_{mass}\end{gathered} \tag{9}$$

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 (9), where $P_i$ represents the grid point, $\Omega$ 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 (9).

### 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 (Gao et al., 2021). As shown in Fig. 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.

$$\begin{aligned}&\frac{\partial v}{\partial y} = 0 \\ &\frac{v_{boundary} - v_{internal}}{\Delta y} = 0\end{aligned} \tag{10}$$

The first row of (10) 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 (10). Further, we can obtain $v_{boundary} = v_{internal}$, which we can use for boundary padding.

### 3.5 MODEL TRAINING AND PREDICTING

Fig. 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. 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 $5 \times 5$, and 2D convolutional operations with padding of 2 and stride of 1.

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. 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.

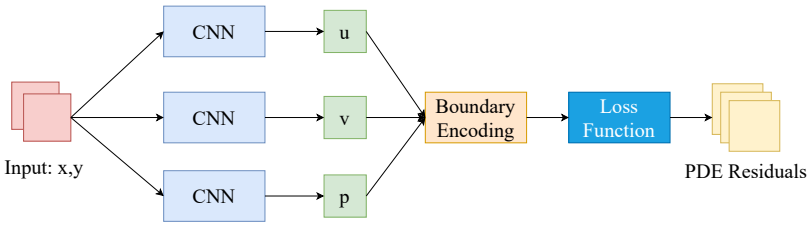


Figure 3: The PICNN architecture for Navier-Stokes equations.

## 4 EXPERIMENTS

We evaluate the accuracy and physical plausibility of our proposed methods on steady-state Navier-Stokes equations under different scenarios. 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.

**Methods.** 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 (Gao et al., 2021), 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.

**Flow through a cavity.** We choose the spatial domain size as $[0, 1] \times [0, 1]$ and divide it into $50 \times 50$ ($\delta x = 0.02$) 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 $\mu$ of the fluid. As the Reynolds number increases, the convection becomes stronger.

Table 1: Relative errors of predicted solutions 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** |

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.

Fig. 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 Fig. 5. From Fig. 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 $y = 0.5$ (Fig. 5 (b)). The predicted solutions from $x = 0.45$ to $x = 0.65$ in Fig. 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.

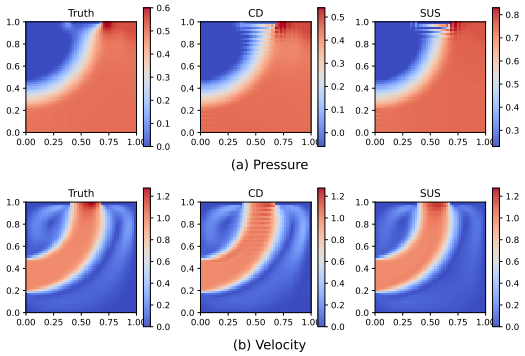


Figure 4: Comparison of solutions between CD-FVM and SUS-FVM for flow through a cavity. (Re = 300)

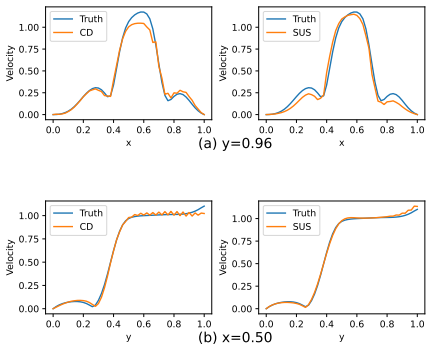


Figure 5: Comparison of velocity accuracy on section $y = 0.96$ and section $x = 0.50$. ($Re = 300$)

**Advantage of conservative schemes.** Both CD-FDM and SUS-FDM use discrete schemes derived from non-conservative equations. And PINN also use the non-conservative equations to construct the loss function. 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. And in both cases, the prediction accuracy of PINN is lower than SUS-FVM. The experimental results illustrate the advantages of deriving discrete schemes with conservation properties in prediction accuracy.

Table 2: Relative errors of predicted solutions for conservative and non-conservative schemes of the Navier-Stokes equations. Dash "-" means divergence and error cannot be calculated.

| | | CD-FVM | CD-FDM | SUS-FVM | SUS-FDM | PINN |
|---|---|---|---|---|---|---|
| $Re = 30$ | pressure | **0.0478** | 0.1392 | 0.0675 | 0.7760 | 0.1178 |
| | velocity | **0.0318** | 0.1078 | 0.0393 | 0.3281 | 0.2734 |
| $Re = 300$ | pressure | 0.0967 | - | **0.0839** | - | 0.2021 |
| | velocity | 0.1280 | - | **0.0486** | - | 0.3084 |

**Convective heat transfer.** Compared with the Navier-Stokes equations, the convective heat transfer problems add a temperature equation, as shown in (11), where $T$ represents the temperature, $C_p$ represents the heat capacity at constant pressure, $k$ represents the thermal conductivity and $\rho$ 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 $333.15$.

$$\frac{\partial(\rho C_p u T)}{\partial x} + \frac{\partial(\rho C_p v T)}{\partial y} = \frac{\partial}{\partial x}(k\frac{\partial T}{\partial x}) + \frac{\partial}{\partial y}(k\frac{\partial T}{\partial y}) \tag{11}$$

Table. 3 shows the relative errors of predicted temperature for the convective heat transfer problems. Since the maximum 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), $P_\Delta$ gradually increases to 200. As $P_\Delta$ 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 Fig. 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 Fig. 6 to further observe the temperature distribution on it, as shown in Fig. 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.

**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 $\mu$, we obtained three datasets with Reynolds numbers of $100$, $500$, and $1000$, respectively.

As shown in Table. 4, 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 Fig. 8, 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

Table 3: Relative errors of predicted solutions ($T$) for the convective heat transfer problems. ($Re = 500$, $\rho = 1$, $C_p = 1000$)

| | 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** |

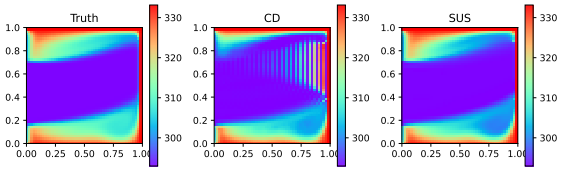


Figure 6: Comparison of solution accuracy between SUS-FVM and CD-FVM schemes for the convective heat transfer problems. (Data3)

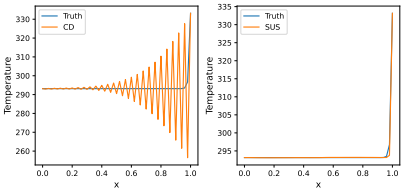


Figure 7: Comparison of solution accuracy on section $y = 0.64$ between SUS-FVM and CD-FVM schemes. (Data3)

Table 4: Relative errors of predicted solutions 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** |

performance of CD-FVM and 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.

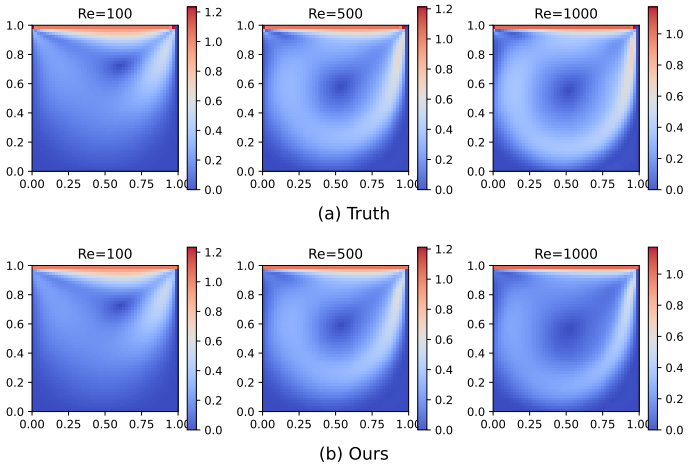


Figure 8: Comparison of predicted and true velocities for the lid-driven cavity flow.

## 5 Conclusion

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.

## REFERENCES

Yohai Bar-Sinai, Stephan Hoyer, Jason Hickey, and Michael P Brenner. Learning data-driven discretizations for partial differential equations. *Proceedings of the National Academy of Sciences*, 116(31):15344–15349, 2019.

Atilim Gunes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *Journal of Marchine Learning Research*, 18:1–43, 2018.

Ken-ichi Fukui, Junya Tanaka, Tomohiko Tomita, and Masayuki Numao. 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)*, pp. 414–419. IEEE, 2019.

Han Gao, Luning Sun, and Jian-Xun Wang. Phygeonet: physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state pdes on irregular domain. *Journal of Computational Physics*, 428:110079, 2021.

Nicholas Geneva and Nicholas Zabaras. Modeling the dynamics of pde systems with physics-constrained deep auto-regressive networks. *Journal of Computational Physics*, 403:109056, 2020.

Yogesh Jaluria and Kenneth E Torrance. *Computational heat transfer*. Routledge, 2017.

Ehsan Kharazmi, Zhongqiang Zhang, and George Em Karniadakis. Variational physics-informed neural networks for solving partial differential equations. *arXiv preprint arXiv:1912.00873*, 2019.

Byungsoo Kim, Vinicius C Azevedo, Nils Thuerey, Theodore Kim, Markus Gross, and Barbara Solenthaler. Deep fluids: A generative network for parameterized fluid simulations. In *Computer Graphics Forum*, volume 38, pp. 59–70. Wiley Online Library, 2019.

Chuizheng Meng, Sungyong Seo, Defu Cao, Sam Griesemer, and Yan Liu. When physics meets machine learning: A survey of physics-informed machine learning. *arXiv preprint arXiv:2203.16797*, 2022.

Xuhui Meng, Zhen Li, Dongkun Zhang, and George Em Karniadakis. Ppinn: Parareal physics-informed neural network for time-dependent pdes. *Computer Methods in Applied Mechanics and Engineering*, 370:113250, 2020.

Arvind T Mohan, Nicholas Lubbers, Daniel Livescu, and Michael Chertkov. Embedding hard physical constraints in neural network coarse-graining of 3d turbulence. *arXiv preprint arXiv:2002.00021*, 2020.

Guofei Pang, Lu Lu, and George Em Karniadakis. fpinns: Fractional physics-informed neural networks. *SIAM Journal on Scientific Computing*, 41(4):A2603–A2626, 2019.

Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

Chengping Rao and Yang Liu. Three-dimensional convolutional neural network (3d-cnn) for heterogeneous material homogenization. *Computational Materials Science*, 184:109850, 2020.

Pu Ren, Chengping Rao, Yang Liu, Jian-Xun Wang, and Hao Sun. Phycrnet: Physics-informed convolutional-recurrent network for solving spatiotemporal pdes. *Computer Methods in Applied Mechanics and Engineering*, 389:114399, 2022.

Rishi Sharma, Amir Barati Farimani, Joe Gomes, Peter Eastman, and Vijay Pande. Weakly-supervised deep learning of heat transport via physics informed loss. *arXiv preprint arXiv:1807.11374*, 2018.

Akshay Subramaniam, Man Long Wong, Raunak D Borker, Sravya Nimmagadda, and Sanjiva K Lele. Turbulence enrichment using physics-informed generative adversarial networks. *arXiv preprint arXiv:2003.01907*, 2020.

Luning Sun and Jian-Xun Wang. Physics-constrained bayesian neural network for fluid flow reconstruction with sparse and noisy data. *Theoretical and Applied Mechanics Letters*, 10(3):161–169, 2020.

Luning Sun, Han Gao, Shaowu Pan, and Jian-Xun Wang. Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data. *Computer Methods in Applied Mechanics and Engineering*, 361:112732, 2020.

Wenquan Tao. *Numerical Heat Transfer, Second ed.* Xi'an Jiaotong University Press, Xi'an, China, 2001.

Nanzhe Wang, Haibin Chang, and Dongxiao Zhang. Efficient uncertainty quantification and data assimilation via theory-guided convolutional neural network. *SPE Journal*, 26(06):4128–4156, 2021a.

Nanzhe Wang, Haibin Chang, Dongxiao Zhang, Liang Xue, and Yuntian Chen. Efficient well placement optimization based on theory-guided convolutional neural network. *Journal of Petroleum Science and Engineering*, 208:109545, 2022.

Yunzhuo Wang, Hao Sun, and Guangzhong Sun. Dsp-pigan: A precision-consistency machine learning algorithm for solving partial differential equations. In *2021 13th International Conference on Machine Learning and Computing*, pp. 21–26, 2021b.

Nick Winovich, Karthik Ramani, and Guang Lin. Convpde-uq: Convolutional neural networks with quantified uncertainty for heterogeneous elliptic partial differential equations on varied domains. *Journal of Computational Physics*, 394:263–279, 2019.

Liu Yang, Xuhui Meng, and George Em Karniadakis. B-pinns: Bayesian physics-informed neural networks for forward and inverse pde problems with noisy data. *Journal of Computational Physics*, 425:109913, 2021.

Yibo Yang and Paris Perdikaris. Adversarial uncertainty quantification in physics-informed neural networks. *Journal of Computational Physics*, 394:136–152, 2019.

Ruiyang Zhang, Yang Liu, and Hao Sun. Physics-guided convolutional neural network (phycnn) for data-driven seismic response modeling. *Engineering Structures*, 215:110704, 2020.

Yinhao Zhu and Nicholas Zabaras. Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification. *Journal of Computational Physics*, 366:415–447, 2018.

Yinhao Zhu, Nicholas Zabaras, Phaedon-Stelios Koutsourelakis, and Paris Perdikaris. Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. *Journal of Computational Physics*, 394:56–81, 2019.

# A APPENDIX

## 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 (12) with no source to illustrate the cause of the instability and its negative effects.

$$\frac{d(\rho u \phi)}{dx} = \frac{d}{dx}(\Gamma \frac{d\phi}{dx}), x \in [0, L] \tag{12}$$

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 (14). For simplicity, we integrate various factors that affect stability as the grid Peclet number $P_\Delta$ (Tao, 2001), as shown in (13), where $\delta x$ represents the grid size. By combining (14) and (12), we can obtain a central difference based discrete equation (15), where $i-1$ and $i+1$ represent the left and right neighbors of $i$, respectively.

$$P_\Delta = \frac{\rho u \delta x}{\Gamma} \tag{13}$$

$$\begin{aligned} \frac{d\phi}{dx} &= \frac{\phi_{i+1} - \phi_{i-1}}{2\delta x} \\ \frac{d}{dx}(\frac{d\phi}{dx}) &= \frac{\phi_{i+1} + \phi_{i-1} - 2\phi_i}{(\delta x)^2} \end{aligned} \tag{14}$$

$$\phi_i = \frac{(1 - 0.5P_\Delta)\phi_{i+1} + (1 + 0.5P_\Delta)\phi_{i-1}}{2} \tag{15}$$

From (15) we can prove that $\phi_i$ 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 $\phi_{i-1}$ and $\phi_{i+1}$ represent the influence of the $\phi$ at the adjacent point on the point $i$ through convection and diffusion. When $P_\Delta > 2$, the coefficient of $\phi_{i+1}$ is less than zero, causing the influence of $\phi_{i+1}$ to $\phi_i$ as negative, which is physically meaningless.

The stability of the central difference scheme is affected by $P_\Delta$ 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 $\phi_{i+1}$. 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 (Tao, 2001):

*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 (Jaluria & Torrance, 2017). 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 ($\phi$ and physical properties) and the first derivative of $\phi$ are continuous on the same interface.

The existing FDM-based scheme violates the first condition to ensure conservativeness. For instance, the first equation in (16) is a conservative governing equation for the one-dimensional con-

vection problem, while the second is a non-conservative governing equation. The FDM-based discrete schemes derived from the second equation in (16) 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 equation in (16), and thus cannot obtain discrete equations with conservation properties.

$$\begin{aligned} \frac{\partial \phi}{\partial t} + \frac{\partial (u\phi)}{\partial x} &= 0 \\ \frac{\partial \phi}{\partial t} + u\frac{\partial \phi}{\partial x} &= 0 \end{aligned} \tag{16}$$

### A.3 Mass Equation Discretization

For the third equation in (1), we integrate it over the control volume of $P$ and obtain (17).

$$\int_s^n \int_w^e [\frac{\partial (\rho u)}{\partial x} + \frac{\partial (\rho v)}{\partial y}] dxdy = 0 \tag{17}$$

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 (18) of the third equation in Navier-Stokes equations.

$$\rho \Delta y(\frac{u_E - u_W}{2}) + \rho \Delta x(\frac{v_N - v_S}{2}) = 0 \tag{18}$$

To keep the calculated value on the left side of (18) on the same order of magnitude as (8), we divide (18) by $D$ and get (19).

$$\frac{\rho \delta x}{\mu}(\frac{u_E - u_W}{2} + \frac{v_N - v_S}{2}) = 0 \tag{19}$$

### A.4 2nd-order upwind scheme of the momentum equation in the vertical direction

Replace the horizontal velocity $u$ in (8) 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 (20). The derivation process is exactly the same as shown in 3.2.

$$\begin{aligned} &(1 + 0.5\mathcal{M}(P_{\Delta_e}) + 1.5\mathcal{M}(P_{\Delta_w}))v_W + (1 + 1.5\mathcal{M}(-P_{\Delta_e}) + 0.5\mathcal{M}(-P_{\Delta_w}))v_E + \\ &(1 + 0.5\mathcal{M}(P_{\Delta_n}) + 1.5\mathcal{M}(P_{\Delta_s}))v_S + (1 + 1.5\mathcal{M}(-P_{\Delta_n}) + 0.5\mathcal{M}(-P_{\Delta_s}))v_N - \\ &0.5(\mathcal{M}(P_{\Delta_w})v_{WW} + \mathcal{M}(-P_{\Delta_e})v_{EE} + \mathcal{M}(P_{\Delta_s})v_{SS} + \mathcal{M}(-P_{\Delta_n})v_{NN}) - \\ &(1.5(\mathcal{M}(P_{\Delta_e}) + \mathcal{M}(-P_{\Delta_w}) + \mathcal{M}(P_{\Delta_n}) + \mathcal{M}(-P_{\Delta_s})) + 4)v_P + \frac{(p_S - p_N)\delta x}{2\mu} = 0 \end{aligned} \tag{20}$$

### A.5 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 (21) shows the values of the velocities at the four boundaries of the control volume at point $P$. Bringing (21) into (4) and dividing it by $D$, we can organize the equation as shown in (22), where $A(|P_\Delta|)$ is given by (23).

$$\begin{cases} u_e = \dfrac{u_P + u_E}{2} \\ u_w = \dfrac{u_P + u_W}{2} \\ u_n = \dfrac{u_P + u_N}{2} \\ u_s = \dfrac{u_P + u_S}{2} \end{cases} \tag{21}$$

$$
\begin{aligned}
a_P u_P &= a_E u_E + a_W u_W + a_N u_N + a_S u_S + \frac{(p_W - p_E)\delta x}{2\mu} \\
a_E &= A(|P_{\Delta_e}|) + \mathcal{M}(-P_{\Delta_e}) \\
a_W &= A(|P_{\Delta_w}|) + \mathcal{M}(P_{\Delta_w}) \\
a_N &= A(|P_{\Delta_n}|) + \mathcal{M}(-P_{\Delta_n}) \\
a_S &= A(|P_{\Delta_s}|) + \mathcal{M}(P_{\Delta_s}) \\
a_P &= a_E + a_W + a_N + a_S + (P_{\Delta_e} - P_{\Delta_w}) + (P_{\Delta_n} - P_{\Delta_s})
\end{aligned} \tag{22}
$$

$$
A(|P_\Delta|) = \begin{cases} 1, & FUS \\ 1 - 0.5|P_\Delta|, & CD \end{cases} \tag{23}
$$

Equation (22) 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 (22) to obtain the central difference discrete scheme.

$$
a_P v_P = a_E v_E + a_W v_W + a_N v_N + a_S v_S + \frac{(p_S - p_N)\delta x}{2\mu} \tag{24}
$$

The central difference scheme of the momentum equation in the vertical direction (i.e., the second equation in (1)) is the same as the equation in the horizontal direction and is shown in (24), where the coefficients are the same as those in (22). And the central difference scheme of the mass equation is the same as (19).

### A.6 EXPERIMENT ON 1D CONVECTION-DIFFUSION EQUATION

In A.1 and A.2, we analyzed the limitations of central difference scheme using (12) as an example. We use experimental results to verify the results of the analysis. We select the domain interval as $[0, 2]$, and specify the boundary conditions $\phi_0 = 50$ and $\phi_2 = 55$. We divide the domain into 20 points evenly and set $P_\Delta = 5$ to better demonstrate the difference between the central difference scheme (CD) and the 1st-order upwind difference scheme (FUS).

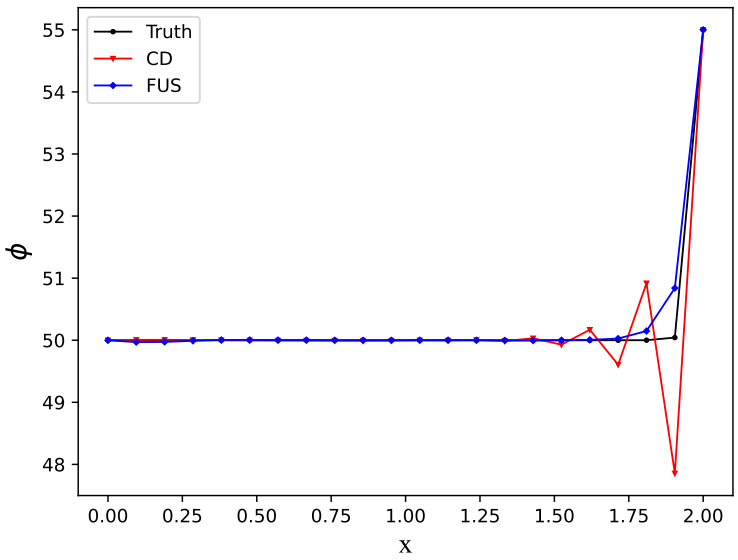


Figure 9: Comparison of solution accuracy between FUS-FVM and CD-FVM schemes for the one-dimensional steady-state convection-diffusion equation.

Table 5: Parameter settings for the 2D convection-diffusion equations. $P_{\Delta u} = \frac{\rho u \delta x}{\Gamma}$,$P_{\Delta v} = \frac{\rho v \delta x}{\Gamma}$. Dirichlet boundaries: $\phi = 10, 7, 5, 1$ for up, down, left, and right boundary, respectively.

| | $\frac{\rho u}{\Gamma}$ | $\frac{\rho v}{\Gamma}$ | $\delta x$ | $P_{\Delta u}$ | $P_{\Delta v}$ |
|---|---|---|---|---|---|
| Data1 | 100 | 200 | 0.01 | 1 | 2 |
| Data2 | 100 | 200 | 0.02 | 2 | 4 |
| Data3 | 1000 | 2000 | 0.01 | 10 | 20 |
| Data4 | 10000 | 20000 | 0.01 | 100 | 200 |

Table 6: Relative errors of predicted solutions ($\phi$) for the 2D convection-diffusion equations.

| | CD-FVM | FUS-FVM |
|---|---|---|
| Data1 | **0.0156** | 0.0240 |
| Data2 | 0.0435 | **0.0363** |
| Data3 | 0.1737 | **0.0368** |
| Data4 | 0.3445 | **0.0318** |

As shown in Fig. 9, near the right boundary of the domain, the predicted solutions using the central difference scheme exhibit strong oscillations, which are physically unreasonable when the velocity $u$ is greater than zero. Although the predicted solutions using the 1st-order upwind difference scheme have large prediction errors near the right boundary, they still conform to the laws of physics. The experimental results show the necessity of using the upwind scheme with higher-order truncation errors when solving equations with convection terms (i.e., second-order upwind difference scheme). That is, to improve the prediction accuracy while ensuring the physical plausibility.

### A.7 EXPERIMENT ON 2D CONVECTION-DIFFUSION EQUATIONS

We evaluate our proposed method on the steady-state two-dimensional convection-diffusion equations without source, whose conserved form is defined as (25). We choose the spatial domain size as $[0, 1] \times [0, 1]$ and divide it into $50 \times 50$ ($\delta x = 0.02$) or $100 \times 100$ ($\delta x = 0.01$) grids evenly. We set the velocity components in the horizontal and vertical directions as constants, and combine them with physical properties to obtain $\frac{\rho u}{\Gamma}$ and $\frac{\rho v}{\Gamma}$, as shown in Table. 5. By adjusting the size of the grids and $\frac{\rho u}{\Gamma}/\frac{\rho v}{\Gamma}$, we can get different values of $P_{\Delta u}$ and $P_{\Delta v}$. We choose Dirichlet boundary conditions and give different values of $\phi$ at each of the four boundaries. The central difference scheme and the first-order upwind scheme derived using FVM are used for comparison with the numerical results.

$$\frac{\partial(\rho u \phi)}{\partial x} + \frac{\partial(\rho v \phi)}{\partial y} = \frac{\partial}{\partial x}(\Gamma \frac{\partial \phi}{\partial x}) + \frac{\partial}{\partial y}(\Gamma \frac{\partial \phi}{\partial y}) \tag{25}$$

From the Table. 6, we can find that when the value of $P_\Delta$ in the horizontal and vertical directions is less than or equal to 2 (Data1), the prediction accuracy of the central difference scheme is higher than that of the first-order upwind scheme. Compared with Data1, Data2 doubles the grid size, making $P_{\Delta v} > 2$, which leads to an increase in the prediction error of the central difference scheme. By changing the value of $\frac{\rho u}{\Gamma}$ and $\frac{\rho v}{\Gamma}$, the value of $P_\Delta$ is further increased, which is much larger than 2 (Data3 and Data4). We can see that the prediction error of the central difference scheme increases significantly as $P_\Delta$ increases, while the error of the first-order upwind difference scheme remains basically the same. Fig. 10 visually shows the predictions of the model within the domain of definition. As $P_\Delta$ increases, the central difference scheme gives completely wrong predictions, while the first-order upwind scheme gives accurate predictions.

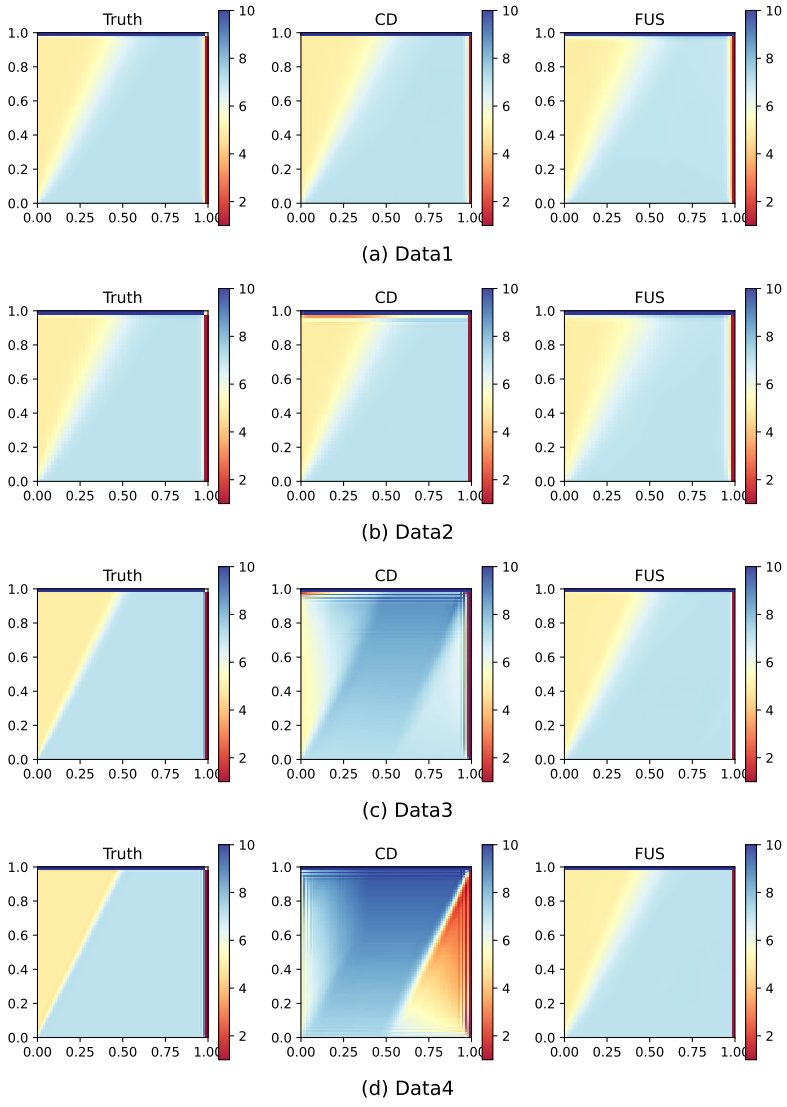


Figure 10: Comparison of solution accuracy between FUS-FVM and CD-FVM schemes for the two-dimensional steady-state convection-diffusion equations.