Neural Computing and Application manuscript No.
(will be inserted by the editor)

Navier-Stokes Generative Adversarial Network: A Physics-Informed Deep Learning Model for Fluid Flow Generation

Pin Wu[1] · Kaikai[1] Pan · Lulu Ji[1] · Siquan Gong[1] · Weibing Feng[1] · Christopher Pain[2,3]

Received: date / Accepted: date

Abstract Numerical simulation in Computational Fluid Dynamics (CFD) mainly relies on discretizing the governing equations in time or space to obtain numerical solutions, which is expensive and time-consuming. Deep learning significantly reduces the computational cost due to its great nonlinear curve fitting capability, however, the data-driven models is agnostic to latent relationships between input and output. In this paper, a novel deep learning named Navier-Stokes Generative Adversarial Network (NSGAN) integrated with physical information is proposed. The Navier-Stokes Equation is added to the loss function of the generator in the form of residuals, which means physics loss in this paper. Then, the proposed model is trained in the framework of generative adversarial network. Experimental results show that proposed model significantly outperform similar models, mean absolute error are decreased by 62.29%, 78.42% and 78.61% on pressure and velocity components. What's more, effectiveness of physics loss is also verified.

Keywords Generative Adversarial Network · Physics-Informed Neural Network · Navier-Stokes equation · Fluid Flow · Deep Learning

✉P. Wu
wupin@shu.edu.cn

K. Pan
pankaikai@shu.edu.cn

L. Ji
fh_jilulu@163.com

S. Gong
gongsiquan@shu.edu.cn

W. Feng
wbfeng@shu.edu.cn

C. Pain
c.pain@imperial.ac.uk

[1] School of Computer Engineering and Science, Shanghai University, Shanghai, 200444, China
[2] Data Science Institute, Department of Computing, Imperial College London, UK
[3] Department of Earth Science and Engineering, Imperial College London, UK

1 Introduction

Fluid flow is ubiquitous in nature and industry. Accurate simulation of fluid flow is an important part of engineering in aerospace, transportation, civil engineering, medicine and other fields. The research methods of fluid flow mainly include experimental research, theoretical analysis and numerical computing. In numerical computing, numerical solution is obtained according to the governing equations of flow field. The traditional numerical calculation mainly relies on computer to discretize the flow field governing equations, mainly includes Finite Element Method (FEM) [5], Finite Difference Method (FDM) [7], and Finite Volume Method (FVM) [11]. However, large-scale problems in Computational Fluid Dynamics (CFD), such as uncertainty quantification, bayesian inversion, data assimilation and constrained optimization of partial differential equations etc.. It is considered to be a very challenging and time-consuming problem. Therefore, the rise of deep learning promotes the development of intelligent fluid dynamics.

Neural network is a data-driven algorithm, which has strong modeling ability of nonlinear system and can approximate any nonlinear function [6]. In recent years, neural network algorithm is widely used in image processing [21], natural language processing [25], pattern recognition [12] and other fields. The strong fitting ability of neural network has excellent performance in numerical prediction and data generation [28,3,22,13]. Therefore, simulating the fluid flow based on strong fitting ability of neural network has become a hot topic in the field of CFD. Lye et al. [10] propose a deep artificial neural network algorithm to predict the potential input parameters of observable data from several training samples (obtained by numerical calculation). Kim et al. [9] propose a new algorithm based on convolutional neural network generate models to simulate fluid from a set of simplified parameters. Both of them have a lower computational cost compared with the traditional PDE solver.

Although deep learning model has great advantages on computation cost, the numerical solution is obtained by the governing equations, conventional neural network is purely data-driven, which is considered as a "black box". Therefore, the hidden physical principle cannot be learned. Thus, deep learning model integrated with physical information has gradually aroused people's interest. In the training process of neural network, the prior knowledge based on physical laws introduces an important structure, which can effectively regulate the minimization process, and make the neural network maintain good fitting performance in the case of a small number of samples. Based on this idea, Raissi et al. [15, 14, **?**] apply neural network to approximate the solution of Partial Differential Equations (PDEs) and propose a Physics-Informed Neural Networks (PINNs). Then, their team solves the forward and inverse problems of PDEs based on PINNs, and discusses the fluid control equation as an example. For a more in-depth study, Raissi et al. [17,16] propose Hidden Fluid Mechanics (HFM), a deep learning framework based on physical information, which encodes Navier-Stokes equations into neural network, so that the model is not affected by geometry or initial and boundary conditions. The model has good performance in physical baseline experiments and medical cases, and has strong robustness to low resolution and large amount of noise in observation data.

Inspired by the work of Raissi's team, the idea of physics-informed invokes a popular research interest in research of fluid flow based on deep learning. Rao et al. [19] propose a mixed-variable scheme of physics-informed neural network for fluid dynamics to simulate steady and transient laminar flows at low Reynolds numbers. Bai et al. [1] propose DBNet which considered the discrete Boltzmann equation with BGK collision term as a regularization term. Wandel et al. [24] introduce a physics loss function based on the residuals of Navier Stokes equations on three-dimensional staggered grids, and propose an effective architecture based on 3D-U-Net. Xu et al. [26] use a control equation as a parametric constraint to recover the missing flow field data based on the idea of PINNs. Erichson et al. [4] add the Lyapunov stability regularization term to physics-informed model to reduce the prediction uncertainty in typical fluid experiments.

Generative Adversarial Network (GAN) has a great performance on data generation, while combining it with the idea of PINNs, the generation ability of GAN can be enhanced. Yang et al. [27] strength the constraints of GAN by incorporating physics equations into the loss function of the generator and verify on geometrical case and differential case. Subramaniam et al. [23] propose a Turbulence Enrichment GAN (TEGAN) by modifying the loss function to minimize the residual of the control equation of the generated data to enrich data.

GAN can effectively learn the distribution of data, but it has the problem of slow convergence. The introduction of PINNs can improve the robustness of the model and speed up the convergence of the model. Such model may have a great performance on generation of flow field. As for conventional deep learning models, target data of flow field is generated from train data with several past time steps, which means there must be time dependence. That could lead to error cumulation. Therefore, this paper proposes a Navier-Stokes General Adversarial Network (NSGAN) based on the flow field governing equation, aims to generate flow field data at low cost and high precision when the time step is given. With the idea of physics-informed introduced, flow field data can be generated more precisely and faster, which is a key issue in CFD.

Contributions in this paper are follows:

1) Data set consist of the discrete points at each moment instead of data at the whole time so that the time dependence can be ignored;
2) Train PINNs under the a particular GAN framework, *swish* activation function and weight normalization are also introduced to accelerate model training;
3) Input of discriminator is arranged in a special form so that a Convolution Neural Network (CNN) is used to improve performance.

The paper is organized as follows. Introduction are presented together in this section. In Section 2, the proposed method is described in detail. Section 3 provides

the experimental details and comparison results. Whole the paper is concluded and further discussed in Section 4.

2 Proposed Method

In this section, related models and technologies are described in detail.

2.1 Governing equations

The research object in this paper is incompressible fluid, of which density variation can be ignored, such as common water (generally regarded as incompressible fluid) and air (at low speed). Navier-Stokes equation is a typical physical model to describe this kind of fluid:

$$\nabla \cdot \boldsymbol{u} = 0 \tag{1}$$

$$\frac{\partial \boldsymbol{u}}{\partial t} - \boldsymbol{Re}^{-1} \cdot \nabla^2 \boldsymbol{u} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} + \nabla p = \boldsymbol{0} \tag{2}$$

where $\boldsymbol{u}$, p, $\boldsymbol{Re}$ are the velocity vector, pressure and the Reynolds number, respectively. For a general two-dimensional model, the governing equation can be written as follows:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \tag{3}$$

$$\frac{\partial u}{\partial t} - \frac{1}{\boldsymbol{Re}}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + \frac{\partial p}{\partial x} = 0 \tag{4}$$

$$\frac{\partial v}{\partial t} - \frac{1}{\boldsymbol{Re}}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + \frac{\partial p}{\partial y} = 0 \tag{5}$$

where $\boldsymbol{u} = [u, v]$, u, v represent the components of the fluid velocity along the x-axis and y-axis, respectively. $\frac{\partial}{\partial t}$, $\frac{\partial}{\partial x}$, $\frac{\partial}{\partial y}$ mean partial differential operations with respect to t, x, y respectively.

2.2 Physics-Informed Neural Networks

The goal of PINNs [15,14] is to infer the function $u(t, \boldsymbol{x})$ of the solution of the nonlinear PDEs. The general form of d-dimensional PDEs is given as follows:

$$\begin{cases} \frac{\partial u}{\partial t}(t, \boldsymbol{x}) + \mathcal{L}_u(t, \boldsymbol{x}) = 0, t \in [0, T], \boldsymbol{x} \in \Omega \in \mathbb{R}^d \\ u(0, \boldsymbol{x}) = u_0(\boldsymbol{x}), \boldsymbol{x} \in \partial\Omega \\ u(t, \boldsymbol{x}) = a(t, \boldsymbol{x}), t \in [0, T], \boldsymbol{x} \in \partial\Omega \end{cases} \tag{6}$$

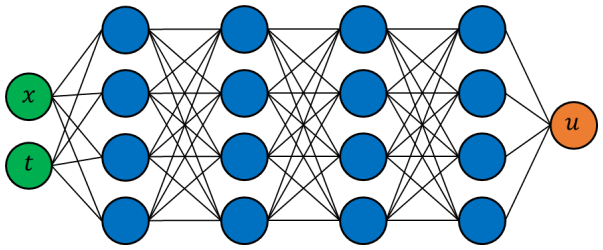

Fig. 1 Structure of neural network

where $\mathcal{L}$ denotes differential operation, $u_0(\boldsymbol{x})$, $a(t, x)$ denote known functions, respectively. A new function $f(t, \boldsymbol{x})$ is defined as follows:

$$f := \frac{\partial u}{\partial t}(t, \boldsymbol{x}) + \mathcal{L}_u(t, \boldsymbol{x}) \tag{7}$$

where the variables of $f(t, \boldsymbol{x})$ can be obtained from the output of the neural network. The structure of neural network is shown in Fig. 1. The input of the network is $(t, \boldsymbol{x}) \in \mathbb{R}^{d+1}$. The goal of network is to construct a nonlinear function $u(t, \boldsymbol{x}; \boldsymbol{\theta})$ with parameter $\boldsymbol{\theta}(\boldsymbol{\theta} \in \{\boldsymbol{W}, \boldsymbol{b}\})$ to approximate the solution $u(t, \boldsymbol{x})$ of the PDEs. The loss function is defined as follows:

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}) = & \frac{1}{N_u}\sum_{i=1}^{N_u}\left(u\left(t_u^i, \boldsymbol{x}_u^i\right) - u\right)^2 \\ & + \frac{1}{N_{inner}}\sum_{i=1}^{N_{inner}}\left(u\left(t_{inner}^i, \boldsymbol{x}_{inner}^i\right) - u\right)^2 \\ & + \frac{1}{N_f}\sum_{i=1}^{N_u} f\left(t_f^i, \boldsymbol{x}_f^i\right)^2 \end{aligned} \tag{8}$$

where $(t_u^i, \boldsymbol{x}_u^i, u^i)_{i=1}^{N_u}$ is the initial and boundary points of $u(t, \boldsymbol{x})$, $(t_{inner}^i, \boldsymbol{x}_{inner}^i, u^i)_{i=1}^{N_{inner}}$ is the training points of $u\left(t, \boldsymbol{x}\right)$ in the domain $\boldsymbol{x} \in \Omega \in \mathbb{R}^d$, and $(t_f^i, \boldsymbol{x}_f^i)_{i=1}^{N_f}$ is the fitting points from network output. Specifically, the first term of the loss function represents the boundary conditions and initial conditions of PDEs, the second term describes the discrepancy between the ground truth and the approximate solution from neural network, and the last term is a regularization constraint term, which is the key part of the PINNs. The PDEs are introduced into the loss function in the form of residual. The specific training process is shown as follows:

There should be noted that PINNs applies automatic differentiation [2] to obtain the derivatives which is needed for calculating defined function f. In the definition of automatic differentiation, all numerical calculations are ultimately the combination of a finite set of elementary operations with known derivatives, the derivative of the whole operation can be obtained by combining the derivatives of the operation which based

Algorithm 1 PINNs: Physics-Informed Neural Networks

Input:

Training points $\left(t_u^i, \boldsymbol{x}_u^i, u^i\right)_{i=1}^{N_u}$ under initial and boundary conditions;

Training points $\left(t_{\text{inner}}^i, \boldsymbol{x}_{inner}^i, u^i\right)_{i=1}^{N_{inner}}$ in domains;

Approximate points $\left(t_f^i, \boldsymbol{x}_f^i\right)_{i=1}^{N_f}$ from neural network.

Output:

Approximate solution: $u(t, \boldsymbol{x})$

1: Construct neural network $u(t, \boldsymbol{x}; \boldsymbol{\theta})$ with parameter $\boldsymbol{\theta} \in \{\boldsymbol{W}, \boldsymbol{b}\}$;
2: Define training points set $\left(t_u^i, \boldsymbol{x}_u^i, u^i\right)_{i=1}^{N_u}$, $\left(t_{inner}^i, \boldsymbol{x}_{inner}^i, u^i\right)_{i=1}^{N_{inner}}$, $\left(t_f^i, \boldsymbol{x}_f^i\right)_{i=1}^{N_f}$;
3: Define loss function $\mathcal{L}$ as Eq. 8;
4: Update parameters $\boldsymbol{\theta}$ by minimize $\mathcal{L}(\boldsymbol{\theta})$;
5: Obtain approximate function $u(t, \boldsymbol{x})$;
6: **return** $u\left(t, \boldsymbol{x}\right)$;

on the chain rule. Combined with the research of this paper, a PINNs based Navier-Stokes equation is illustrated in Fig. 2.

2.3 Navier Stokes Generative Adversarial Network

Generative Adversarial Network (GAN) is a generative model proposed by Goodflow et al. [17] in 2014. GAN consists of a generator and a discriminator. The generator learns the data distribution from real samples and generates new samples while the discriminator can be regarded as a classifier to discriminate the input between real data and generated samples. The parameters of GAN are trained alternately by two loss functions as Eq. 9 and Eq. 10:

$$\begin{aligned}\mathcal{L}_D = &\mathbb{E}_{x\sim p_x}\left[\log D\left(x;\theta_D\right)\right] \\ &+\mathbb{E}_{z\sim p_z}\left[\log\left(1-D\left(G\left(z;\theta_G\right);\theta_D\right)\right)\right]\end{aligned} \tag{9}$$

$$\mathcal{L}_G = \mathbb{E}_{z\sim p_z}\left[\log\left(1-D\left(G\left(z;\theta_G\right);\theta_D\right)\right)\right] \tag{10}$$

where $\mathbb{E}$ is expectation. x is sampled from distribution of real data p_x while z is the noise sampled from a prior distribution p_z (e.g. Gaussian noise distribution). During the whole training process, the optimization of $\mathcal{L}_D$ and $\mathcal{L}_G$ are carried out alternately. When the discriminator is been training, the parameter of generator θ_G is fixed, then the parameter of discriminator θ_D is updated through optimization of $\mathcal{L}_D$. Similarly, when training the generator, the parameter of discriminator θ_D is fixed and the parameter of generator θ_G is updated. The purpose of discriminator is to distinguish the real data from generative samples as accurately as possible. The output of discriminator is a probability, that means while the input is sampled from the real data, the probability $D\left(x\right)$ should approach 1, and while the input is from the generative samples, the probability $D\left(G(z)\right)$ should approach 0. The goal of generator is to fool discriminator, making the discriminator unable to judge the authenticity of input samples. That become a game, the performance of generator and discriminator are improved gradually. In the most ideal case, the probability $D\left(G(z)\right)$ equals 0.5. Therefore, the objective function of GAN can be defined as Eq. 11:

$$\begin{aligned}\min_G \max_D \mathcal{L}(D,G) = &\mathbb{E}_{x\sim p_x}\left[\log D\left(x;\theta_D\right)\right] \\ &+\mathbb{E}_{z\sim p_z}\left[\log\left(1-D\left(G\left(z;\theta_G\right);\theta_D\right)\right)\right]\end{aligned} \tag{11}$$

In this paper, a novel GAN named Navier-Stokes Generative Adversarial Network (NSGAN) is proposed based on the idea of PINNs. The residual of Navier-Stokes equation is transformed into generator loss function as a regularizer in the network. The loss function of generator is defined as Eq. 12:

$$\begin{aligned}\mathcal{L}_{generator} = &\mathcal{L}_{adversarial} \\ &+\mathcal{L}_{MSE}+\lambda_{physics}\mathcal{L}_{physics}\end{aligned} \tag{12}$$

where $\lambda_{physics}$ is the weight of physics loss functions, $\mathcal{L}_{MSE}$, $\mathcal{L}_{physics}$, $\mathcal{L}_{adversarial}$ are given as follows:

$$\mathcal{L}_{MSE} = \frac{1}{N}\sum_{n=1}^{N}(\hat{Y}-Y)^2 \tag{13}$$

$$\mathcal{L}_{physics} = \frac{1}{N}\sum_{i=1}^{N}\left(\widehat{\boldsymbol{r}}_i-\boldsymbol{r}_i\right)^2 \tag{14}$$

$$\mathcal{L}_{adversarial} = \frac{1}{N}\sum_{n=1}^{N}-\log D(G(X)) \tag{15}$$

where X, Y, $\hat{Y}$ are the input, the ground truth and generated data, respectively. $\hat{Y}$ is obtained by Eq. 16 :

$$\hat{Y} = G(X) \tag{16}$$

The loss function of the discriminator is defined as follows:

$$\begin{aligned}\mathcal{L}_{discriminator} = &\frac{1}{N}\sum_{n=1}^{N}-logD(Y) \\ &+\log(1-D(G(X)))\end{aligned} \tag{17}$$

Here we want to emphasize that in the loss function of generator, we have modified the conventional loss function. We choose the function $-logD(G(X))$ instead

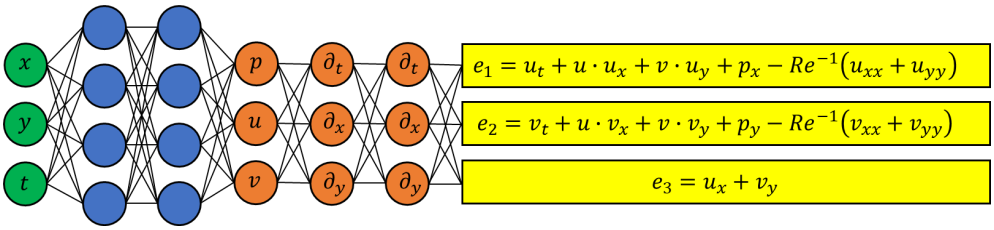

Fig. 2 Structure of Physics-Informed Neural Network

of the conventional one $\log(1 - D(G(X)))$. The latter tends to be negative infinity when approaching 1, while the former has a slightly change in gradient, which can reduce the oscillation of the loss function.

Since the input of generator in NSGAN is just composed of time variable and coordinates of node, there is no time dependence between different samples. Further more, in order to better use the automatic differentiation, the structure of generator is set to fully connected layers; Most structures of discriminator in GAN or other derivative model are set to Convolution Neural Network (CNN), for which CNN has great performance on feature extraction. However, considering that the input of discriminator in this paper is the coordinates x, y of a single node and the time step t, the pressure p, the velocity component u along the x-axis and the velocity component v along the y-axis. That is a sequence, which is different from general case for CNN. Thus, a special module is introduced. In a flow field, data of each point is closely linked to coordinates and time step, so the coordinate x, y and the time step t are duplicated and the input is reshape to 3×3, which can be regard as a low-solution picture. This module is shown in Fig. 4. The reason why duplicated variables are coordinates and time step rather than pressure and velocity component is that the goal of discriminator is to judge the authenticity of the input based on the conditions or prior x, y, t. The overall structure of NSGAN is shown in Fig. 3:

For details in Fig. 3, the generator is a fully connected neural network with 10 hidden layers and 256 neurons per hidden layer. The node coordinates x, y and the time step t are input into neural network, and pressure p, the velocity component u along the x-axis, and the velocity component v along the y-axis of the node are generated. In the lower half of Fig. 3, the detail of generator has been shown, which is same as Fig. 2. For discriminator, there is a special module named "rearrange", the structure of discriminator is show as Fig. 4. The network input the node coordinates x, y, the time step t, and pressure p, the velocity component u along the x-axis, the velocity component v along the y-axis of the node, since p, u, v are closely linked to x, y, t, these input is rearranged, which is also shown in Fig. 4. Discriminator output a probability which describes the authenticity of the input samples.

From Fig. 3, there are several differences compared with conventional structure of GAN:

1) The input of generator does not conclude a noise vector. In a general GAN, the noise vector is input into generator to generate predicted data;
2) The input of generator is also input into discriminator.

For the issue 1), since derivatives are obtained from automatic differentiation, the input of generator must conclude coordinates and time step, therefore, the noise vector is unnecessary in this generator, which is similar to a Conditional Generative Adversarial Network (CGAN) [8] for image-to-image translation(a example is shown in Fig. 5). For the issue 2), samples in this paper are nodes of field flow, and difference between these node is close, input p, u, v merely into discriminator may lead to poor performance on discrimination of input. Flow field data is closely linked to coordinates and time step, thus, x, y, t, p, u, v are input into discriminator together and a rearrangement is conducted so that the performance of discriminator can be enhanced.

To sum up, Eq. 15 and Eq. 17 can be rewritten as:

$$\mathcal{L}_{adversarial} = \frac{1}{N}\sum_{n=1}^{N} -\log D([X, G(X)]) \tag{18}$$

$$\begin{aligned}\mathcal{L}_{discriminator} = \frac{1}{N}\sum_{n=1}^{N} &\log D([X, Y]) \\ &+ \log(1 - D([X, G(X)]))\end{aligned} \tag{19}$$

The activation function of generator is set to *swish* activation function [18] which is given by:

$$swish(x) = x \cdot sigmoid(\beta x) = \frac{x}{1 - e^{-\beta x}} \tag{20}$$

in this paper, we set β for 1:

$$swish(x) = x \cdot sigmoid(x) = \frac{x}{1 - e^{-x}} \tag{21}$$

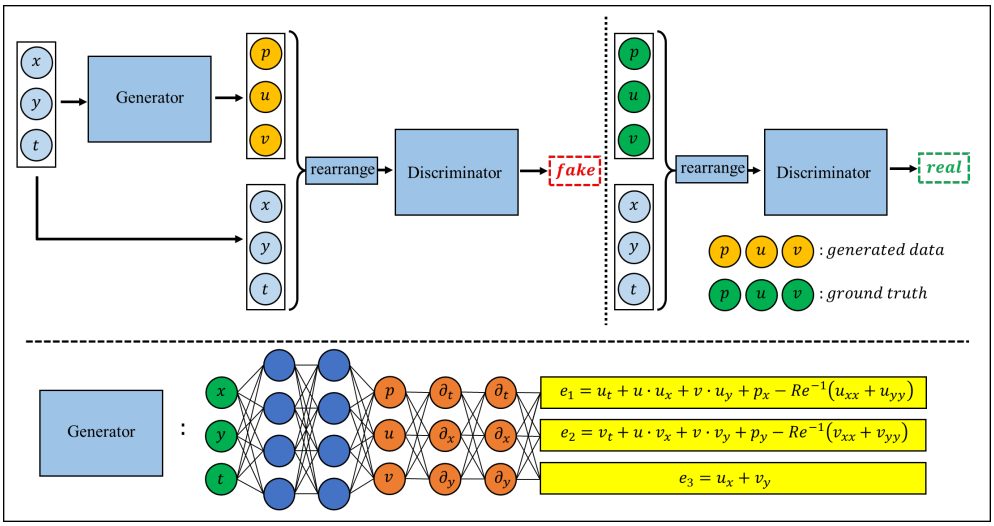

Fig. 3 Structure of Navier-Stokes Generative Adversarial Network

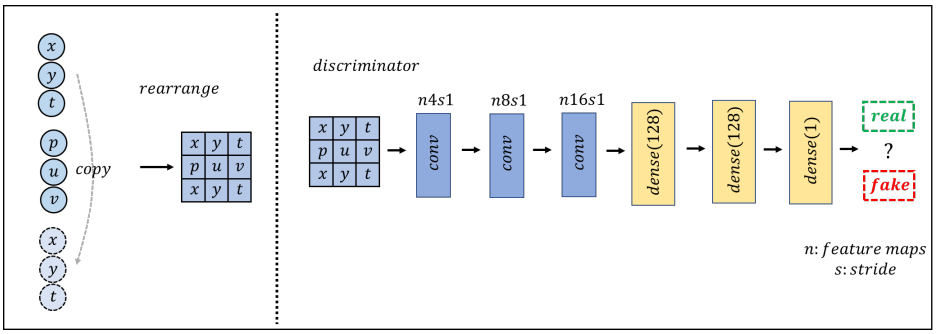

Fig. 4 Structure of discriminator

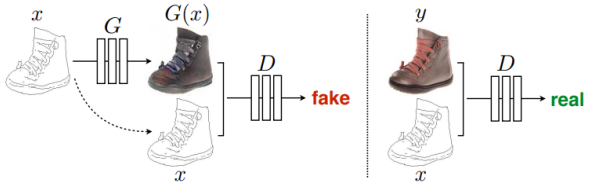

Fig. 5 Image-to-image translation on shoes with CGAN

Same as *ReLU*, *swish* activation function has lower bound but no upper bound, but *swish* activation function is smooth and non-monotonic. It is said that *ReLU* could lead to vanishing gradient on higher-order derivatives. Thus, for better computing the derivatives needed for residual calculation, the activation function of generator is set to *swish*. The activation function of discriminator is set to *sigmoid* activation function:

$$sigmoid(x) = \frac{x}{1 - e^{-x}} \tag{22}$$

In addition, this paper normalizes the weights of generator and discriminator network [20]. Given a general calculation of a neuron:

$$y = \sigma\left(\boldsymbol{W} \cdot \boldsymbol{X} + \boldsymbol{b}\right) \tag{23}$$

where $\boldsymbol{W}$, $\boldsymbol{b}$ represent the weight and bias of the neuron, respectively. The strategy of weight normalization is to decompose $\boldsymbol{W}$ into a parameter vector $\boldsymbol{V}$ and a parameter scalar g. The decomposition method is given as Eq. 24.

$$\boldsymbol{W} = \frac{g}{\|\boldsymbol{V}\|}\boldsymbol{V} \tag{24}$$

where $\|\boldsymbol{V}\|$ represents the Euclidean norm of $\boldsymbol{V}$. Both $\boldsymbol{V}$ and g can be updated by gradient descent, as is shown below:

$$\nabla_g \mathcal{L} = \frac{\nabla_W \mathcal{L} \cdot \boldsymbol{V}}{\|\boldsymbol{V}\|} \tag{25}$$

$$\nabla_{\boldsymbol{V}} \mathcal{L} = \frac{g}{\|\boldsymbol{V}\|}\nabla_{\boldsymbol{W}} \mathcal{L} - \frac{g\nabla_g \mathcal{L}}{\|\boldsymbol{V}\|^2}\boldsymbol{V} \tag{26}$$

where $\mathcal{L}$ is loss function of neural network and $\nabla_{\boldsymbol{W}}\mathcal{L}$ is gradient for $\boldsymbol{W}$ under $\mathcal{L}$. Plug Eq. 25 into Eq. 26, then $\nabla_{\boldsymbol{V}}\mathcal{L}$ can be rewritten as follows:

$$\begin{aligned}\nabla_{\boldsymbol{V}}\mathcal{L} &= \frac{g}{\|\boldsymbol{V}\|}\nabla_{\boldsymbol{W}}\mathcal{L} - \frac{g\nabla_g\mathcal{L}}{\|\boldsymbol{V}\|^2}\boldsymbol{V} \\ &= \frac{g}{\|\boldsymbol{V}\|}\nabla_{\boldsymbol{W}}\mathcal{L} - \frac{g}{\|\boldsymbol{V}\|^2}\frac{\nabla_{\boldsymbol{W}}\mathcal{L}\cdot\boldsymbol{V}}{\|\boldsymbol{V}\|}\boldsymbol{V} \\ &= \frac{g}{\|\boldsymbol{V}\|}\left(\boldsymbol{I} - \frac{\boldsymbol{V}\boldsymbol{V}'}{\|\boldsymbol{V}\|^2}\right)\nabla_{\boldsymbol{W}}\mathcal{L}\end{aligned} \tag{27}$$

Because $\boldsymbol{V}$ is direction vector of $\boldsymbol{W}$, Eq. 27 can be written as:

$$\nabla_{\boldsymbol{V}}\mathcal{L} = \frac{g}{\|\boldsymbol{V}\|}(\boldsymbol{I} - \frac{\boldsymbol{W}\boldsymbol{W}'}{\|\boldsymbol{W}\|^2})\nabla_{\boldsymbol{W}}\mathcal{L} \tag{28}$$

Let

$$\boldsymbol{M}_{\boldsymbol{W}} = \boldsymbol{I} - \frac{\boldsymbol{W}\boldsymbol{W}'}{\|\boldsymbol{W}\|^2} \tag{29}$$

$$\nabla_{\boldsymbol{V}}\mathcal{L} = \frac{g}{\|\boldsymbol{V}\|}\boldsymbol{M}_{\boldsymbol{W}}\nabla_{\boldsymbol{W}}\mathcal{L} \tag{30}$$

where $\frac{g}{\|\boldsymbol{V}\|}$ indicates scaling of weight gradient, $\boldsymbol{M}_{\boldsymbol{W}}\nabla_{\boldsymbol{W}}\mathcal{L}$ indicates that gradient will be projected to a direction away from $\nabla_{\boldsymbol{W}}\mathcal{L}$. These two factor can accelerate the convergence of the model. Weight normalization is higher robustness to learning rate, and lower sensitivity to noise, which is important in GAN.

Algorithm of NSGAN is shown as Algorithm 2, $\boldsymbol{Ep}$ denotes the epoches of overall training, $\boldsymbol{T}$ denotes the epoches of discriminator in a training cycle, which is set to 1 in this paper. $\boldsymbol{b}$ denotes the batch size.

3 Experiments and Results

To verify the proposed framework, the following experiments are conducted:

1) Comparison of generation performance with respect to different $\lambda_{physics}$;
2) Comparison of generation performance with respect to different models.

Algorithm 2 NSGAN: Navier-Stokes Generative Adversarial Network

for $\boldsymbol{Ep}$ **do**
  Sample batch size of $\boldsymbol{b}$ training samples $\{X^1, X^2, \ldots, X^{\boldsymbol{b}}\}$ and $\{Y^1, Y^2, \ldots, Y^{\boldsymbol{b}}\}$,where $X^i = [x^i, y^i, t^i]$, $Y^i = [p^i, u^i, v^i]$,X^i, Y^i are in one-to-one match;
  Update generator based on gradient descent ∇_{θ_G} for loss function of generator $\mathcal{L}_G$ as Eq. 11:
  for $\boldsymbol{T}$ **do**
    Sample batch size of $\boldsymbol{b}$ samples $\{Y^1, Y^2, \ldots, Y^{\boldsymbol{b}}\}$ from the ground truth distribution, where $Y^i = [p^i, u^i, v^i]$, with coordinates and time step of each example $X^i = [x^i, y^i, t^i]$ sampled;
    Update discriminator based on gradient descent ∇_{θ_D} for loss function of discriminator $\mathcal{L}_D$ as Eq. 19:
  end for
end for

3.1 Data Generation

The case study in this paper is the external 2D flow past a circular cylinder, experimental data are the high-precision solution of the flow field. geometric graph of this case is shown as Fig. 6. In this case, Reynolds number $\boldsymbol{Re}$ is set to 100 and time step is set to 0.1s. 500 time steps of the flow field data under stable state are selected as experimental data. At every time step, the data consist of discrete nodes of the whole flow field, each node includes the coordinate x and y , the pressure p, the velocity component u along the x axis, and the velocity component v along the y axis. The 500 time steps are shuffled, then the shuffled data set is divided into training set and test set in a ratio of 1:4.

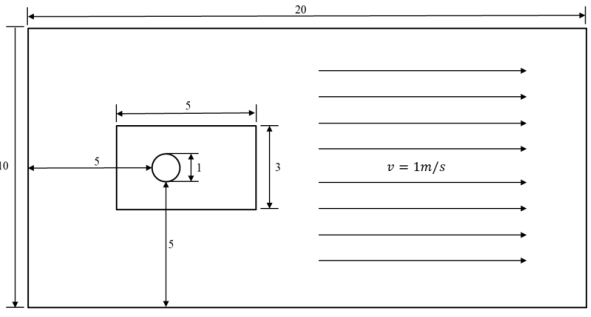

Fig. 6 The geometric graph of external 2D flow past a circular cylinder(m)

3.2 Comparison of generation performance with respect to different $\lambda_{physics}$

The NSGAN proposed in this paper introduces a physical loss, that is, partial differential equation constraint,

but the convergence rate of physical loss function is different from that of adversarial loss, thus, the model cannot achieve the optimal performance when the weights for different part of loss function are the same. Hence, in this section of the experiment, different weights for physical loss $\lambda_{physics}$ are set. The optimal parameter $\lambda_{physics}$ can be obtained through comparison of different models. In this paper, there are four groups of experiments $\boldsymbol{\mathcal{M}}_{1,2,3,4}$, and $\lambda_{physics}$ is set to $\{0.1, 0.2, 0.4, 1\}$, which is correspond to $\boldsymbol{\mathcal{M}}_1$, $\boldsymbol{\mathcal{M}}_2$, $\boldsymbol{\mathcal{M}}_3$ and $\boldsymbol{\mathcal{M}}_4$, respectively. Where $\lambda_{physics} = 1$ is used as the reference group, which means the weights for different loss function are the same. The generative model is obtained after training, and then the flow field data at a random time step in the test set are generated by generator. The cloud images of pressure p, velocity component u along the x axis and velocity component v along the y axis are respectively drawn for visualization. The comparison results are shown in Fig. 7:

From Fig. 7, all the four models can fairly accurately generate flow field data, and there are no obvious disadvantages between each other. However, the flow field data requires much higher accuracy. Therefore, the absolute error $\boldsymbol{ae}_p$, $\boldsymbol{ae}_u$ and $\boldsymbol{ae}_v$ of the pressure p, velocity component u along the x axis and velocity component v along the y axis with respect to different $\lambda_{physics}$ are calculated for further comparison, computing equations are given as follows:

$$\boldsymbol{ae} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{e}_i \tag{31}$$

where

$$\boldsymbol{e}_i = |\widehat{y}_i - y_i| \tag{32}$$

where $\boldsymbol{e}_i$ represents the absolute error of a single node in the flow field. The cloud picture is drawn for the error. Cloud picture for overall flow field is shown in Fig. 8:

In the error cloud picture, the warmer the color is, the larger the absolute error is, otherwise, the smaller the absolute error is. It can be clearly seen from Fig. 8 that the error increases significantly when $\lambda_{physics}$ changes from 0.1 to 1. In terms of flow field generated at random time step, when the parameter $\lambda_{physics} = 0.1$, the $\boldsymbol{\mathcal{M}}_1$ error cloud image appears cold on the whole, which means the absolute error of $\boldsymbol{\mathcal{M}}_1$ is much small and the flow field data generated is more accurate compared with other models. When $\lambda_{physics} = 1$, $\boldsymbol{\mathcal{M}}_4$ error cloud image presents a warm color region with certain flow field characteristics. It can be seen that when $\lambda_{physics} = 0.1$, the model generation performance is much better. However, the flow field data merely observed at a random time step is special case. Therefore, in order to better accurately compare the generation performance with respect to different $\lambda_{physics}$, this paper calculated the average absolute errors of different models on the whole test set. The results are shown in Tab. 1 below:

Table 1 The mean absolute error of models with respect to different $\lambda_{physics}$

$\boldsymbol{\lambda_{physics}}$	$\boldsymbol{ae}_p$	$\boldsymbol{ae}_u$	$\boldsymbol{ae}_v$
0.1	**0.00393**	**0.00521**	0.00671
0.2	0.00509	0.00678	**0.00530**
0.4	0.00579	0.01077	0.01055
1	0.00726	0.01304	0.01218

According to the data in Tab. 1, it can be seen that when $\lambda_{physics} = 0.1$, the average absolute error $\boldsymbol{ae}_p$ of pressure p for $\boldsymbol{\mathcal{M}}_1$ and the mean absolute error $\boldsymbol{ae}_u$ of velocity component u along the x-axis direction are the lowest. When $\lambda_{physics} = 0.2$, the mean absolute error $\boldsymbol{ae}_v$ of the velocity component v along the y-axis for $\boldsymbol{\mathcal{M}}_2$ is the lowest. $\lambda_{physics} = 0.1$ does not have an absolute advantage. However, there should be noted that in the case of flow past a cylinder, the whole flow field may focus on the surrounding and the wake part of cylinder. Therefore, a key region of flow field is divided as shown in Fig. 6, then the absolute error in this key area is calculated, and comparison results are shown in Tab. 2:

Table 2 The mean error of models with respect to different $\lambda_{physics}$ focuses on key region

$\boldsymbol{\lambda_{physics}}$	$\boldsymbol{ae}_p$	$\boldsymbol{ae}_u$	$\boldsymbol{ae}_v$
0.1	**0.0108**	**0.0145**	**0.0159**
0.2	0.0144	0.0263	0.0193
0.4	0.0200	0.0453	0.0296
1	0.0321	0.0705	0.0451

According to Tab. 2, when focusing on key region, when $\lambda_{physics} = 0.1$, all the mean absolute error $\boldsymbol{ae}_p$, $\boldsymbol{ae}_u$, $\boldsymbol{ae}_v$ are the lowest. To sum up, $\boldsymbol{\mathcal{M}}_1$ with $\lambda_{physics} = 0.1$ is considered as the standard model of NSGAN for follow-up experimental verification.

3.3 Comparison of generation performance with respect to different models

The basic structure of PINNs consists of fully connected layers, and most of the existing studies are based on

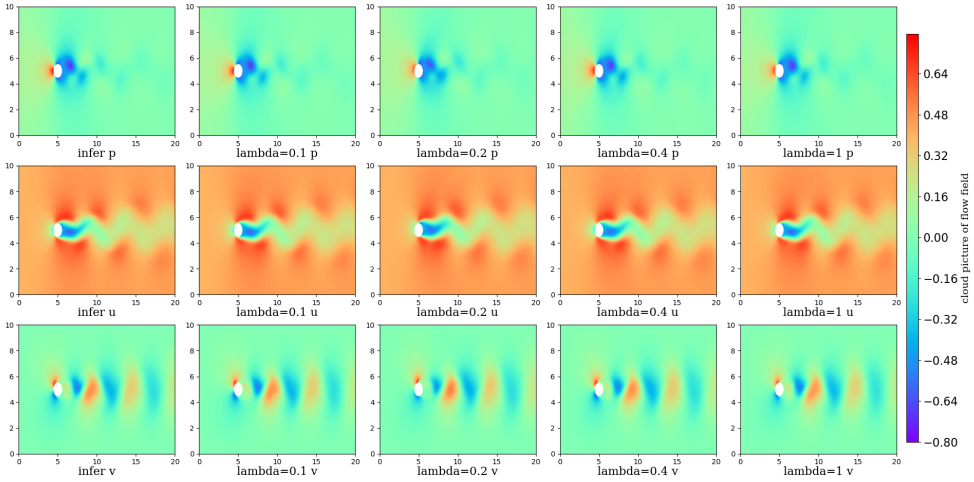

Fig. 7 Flow field cloud pictures with respect to different $\lambda_{physics}$ at a random time step

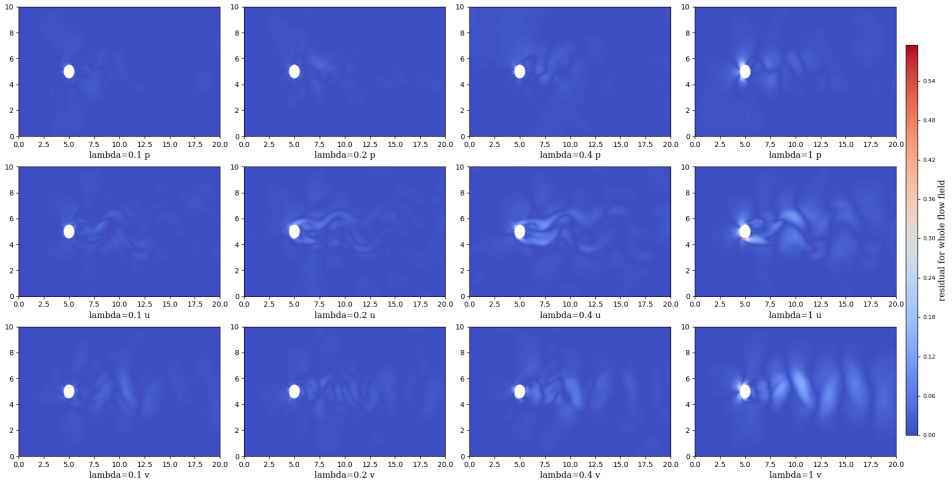

Fig. 8 Error cloud pictures of overall flow field with respect to different $\lambda_{physics}$ at a random time step

the Multi-Layer Perception (MLP). Therefore, a PINNs based on MLP is of certain reference significance and can be considered as a reference model for comparison experiments.

There are two tasks in this experiment:

1) Verify that NSGAN proposed in this paper has better prediction performance on flow field compared with the baseline model;
2) Verify the effectiveness of physical losses.

In this section, four models are compared, the proposed NSGAN, the baseline model PINNs, the ordinary GAN without physical loss and the MLP model without physical loss, which are referred to $\boldsymbol{\mathcal{M}}_{NSGAN}$, $\boldsymbol{\mathcal{M}}_{PINNs}$, $\boldsymbol{\mathcal{M}}_{GAN}$ and $\boldsymbol{\mathcal{M}}_{MLP}$, respectively. It should be added here that the network structure of $\boldsymbol{\mathcal{M}}_{PINNs}$ is the same as the generator of $\boldsymbol{\mathcal{M}}_{NSGAN}$. Compared with $\boldsymbol{\mathcal{M}}_{NSGAN}$ proposed in this paper, the structure of generator and discriminator on $\boldsymbol{\mathcal{M}}_{GAN}$ are the same except that no physical loss is added.

Same as previous works, the trained model is applied to predict the flow field data at random time step in the test set. The cloud pictures of pressure p, velocity component u along the x-axis, and velocity component v along the y-axis are respectively drawn for visualization. The comparison results are shown in Fig. 9:

From the cloud pictures, the flow field generated by either $\boldsymbol{\mathcal{M}}_{PINNs}$ or $\boldsymbol{\mathcal{M}}_{NSGAN}$ proposed is roughly not significantly different from the actual flow field, indicating that both of them predict the flow field data well. When it comes to $\boldsymbol{\mathcal{M}}_{GAN}$ and $\boldsymbol{\mathcal{M}}_{MLP}$ where no physics loss is added, $\boldsymbol{\mathcal{M}}_{MLP}$ is almost unable to generate the flow field, only a few data around the cylinder can be predicted to a certain extent, the wake of the flow is irregular. Because of excellent ability to learning sample distribution for GAN, the generation performance

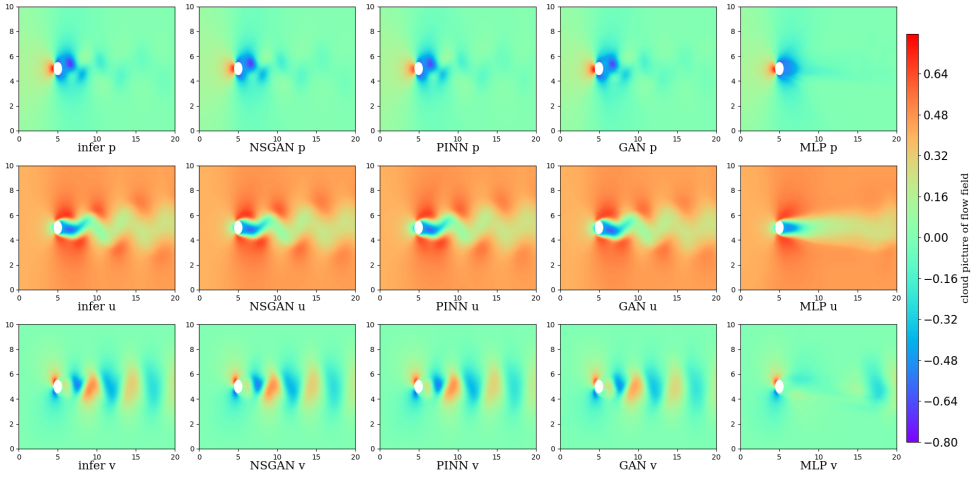

Fig. 9 Flow field cloud pictures with respect to different models at a random time step

of $\mathcal{M}_{GAN}$ is relatively well, but some details are worse than $\mathcal{M}_{NSGAN}$ and $\mathcal{M}_{PINNs}$. Like works above, absolute error $\boldsymbol{ae}_p$, $\boldsymbol{ae}_u$, $\boldsymbol{ae}_v$ of overall flow field are calculated with Eq. 31 and Eq. 32 for different model generation with respect to pressure p, velocity component u along the x-axis, and velocity component v along the y-axis. The absolute error cloud pictures s are shown in Fig. 10:

The color distribution in Fig. 10 is same as Fig. 8. The warmer the color is, the higher the error is, while the colder the color is, the lower the error is. The poor generation performance of $\mathcal{M}_{MLP}$ which is clearly shown in Fig. 9 is verified in Fig. 10. Compared with $\mathcal{M}_{NSGAN}$ and $\mathcal{M}_{GAN}$, the overall color distribution of error cloud picture on $\mathcal{M}_{NSGAN}$ is cold while the error cloud picture on $\mathcal{M}_{GAN}$ represents a clear warm region, which means performance on $\mathcal{M}_{NSGAN}$ is much better than $\mathcal{M}_{GAN}$. In comparison with $\mathcal{M}_{PINNs}$ and $\mathcal{M}_{NSGAN}$, $\mathcal{M}_{NSGAN}$ still has certain advantages. In order to make a better comparison, the absolute error on key region is calculated for visualization. As shown in Fig. 11, we can sum up that $\mathcal{M}_{NSGAN}$ proposed has the best performance, followed by $\mathcal{M}_{PINNs}$, and the error of $\mathcal{M}_{GAN}$ and $\mathcal{M}_{MLP}$ without physical loss is relatively high.

The mean absolute errors of flow field with respect to $\mathcal{M}_{NSGAN}$, $\mathcal{M}_{PINNs}$, $\mathcal{M}_{GAN}$ and $\mathcal{M}_{MLP}$ are calculated on the whole test set. Mean absolute errors in key region are also carried out. The results are shown in Tab. 3 and Tab. 4:

According to the data in Tab. 3 and Tab. 4, whether the mean absolute errors focus on overall flow field or key region of flow field, $\mathcal{M}_{NSGAN}$ has lowest mean absolute error. Specifically, when mean absolute error focus on the whole flow field, error of $\mathcal{M}_{NSGAN}$ is decreased by 42.50%, 53.53% and 43.71% compared with $\mathcal{M}_{PINNs}$, respectively. When mean absolute error focus on the key region of flow field, error of $\mathcal{M}_{NSGAN}$ is decreased by 61.15%, 67.30% and 64.17% compared with $\mathcal{M}_{PINNs}$, respectively. In summary, for task 1) in this section, It can be concluded that $\mathcal{M}_{NSGAN}$ proposed in this paper has great advantages over other similar experiments.

In a validation study for task 2), it is indicated that when focusing on the whole flow field, the mean absolute error of $\mathcal{M}_{NSGAN}$ is decreased by 62.29%, 78.42% and 78.61% respectively compared with $\mathcal{M}_{GAN}$, and is decreased by 75.36%, 83.88% and 75.87% respectively compared when focusing on key region of flow field. It can be proved that adding the physics loss is of great research significance and the effectiveness of physics loss is verified for task 2).

Table 3 The mean absolute error of models with respect to different models

models	$\boldsymbol{ae}_p$	$\boldsymbol{ae}_u$	$\boldsymbol{ae}_v$
$\mathcal{M}_{NSGAN}$	**0.00393**	**0.00521**	**0.00671**
$\mathcal{M}_{PINNs}$	0.00683	0.01121	0.01192
$\mathcal{M}_{GAN}$	0.01042	0.02415	0.03136
$\mathcal{M}_{MLP}$	0.01509	0.03649	0.05330

Table 4 The mean error of models with respect to different models focused on key region

models	$\boldsymbol{ae}_p$	$\boldsymbol{ae}_u$	$\boldsymbol{ae}_v$
$\mathcal{M}_{NSGAN}$	**0.0108**	**0.0145**	**0.0159**
$\mathcal{M}_{PINNs}$	0.0279	0.0444	0.0445
$\mathcal{M}_{GAN}$	0.0439	0.0901	0.0660
$\mathcal{M}_{MLP}$	0.0566	0.1029	0.1069

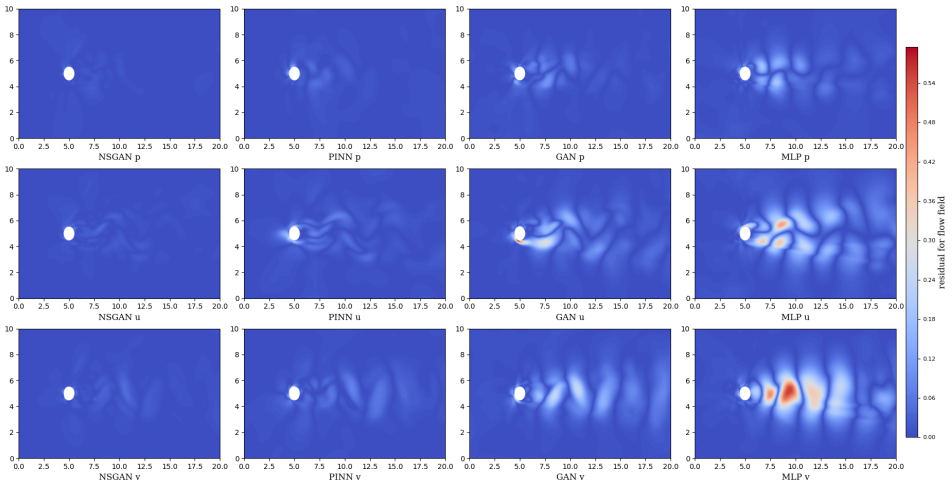

Fig. 10 Error cloud pictures of overall flow field with respect to different models at a random time step

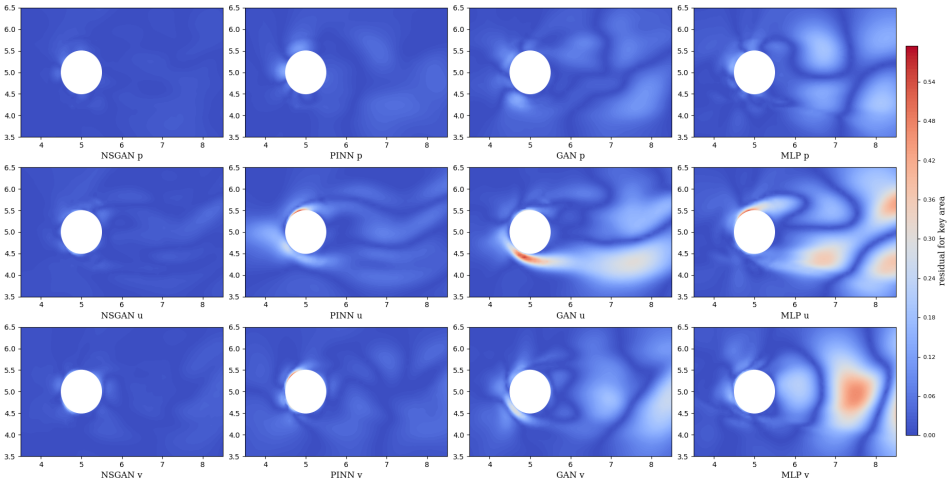

Fig. 11 Error cloud pictures of key regions of flow field with respect to different models at a random time step

4 Conclusion

In this study, a NSGAN integrated with physical loss is proposed for the generation of flow field data. Navier-Stokes equation is added to loss function in the form of residual, the generative model is trained under the GAN framework. In the comparison of weights for physics loss, an optimal weight is selected, which is used as the standard model to compared with similar experiments. In the comparison of similar models, mean absolute error of NSGAN is the lowest. What's more, effectiveness of physics loss is also verified, model with physics loss still has the lowest absolute error. The NSGAN's great performance shows that it can be applied to generate high-precision flow field with a given time step at low cost, which accelerate computation and make generated data be of certain physics significance.

Acknowledgements This work is supported by Natural Science Foundation of Shanghai under Grant 19ZR1417700, Transforming Systems through Partnership, Newton Fund under Grant TSPC1086, High Performance Computing Center of Shanghai University.

5 Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Bai, X.d., Wang, Y., Zhang, W.: Applying physics informed neural network for flow data assimilation. Journal of Hydrodynamics pp. 1–9 (2020)
2. Baydin, A.G., Pearlmutter, B.A., Radul, A.A., Siskind, J.M.: Automatic differentiation in machine learning: a survey. Journal of machine learning research **18** (2018)

3. Booz, J., Yu, W., Xu, G., Griffith, D., Golmie, N.: A deep learning-based weather forecast system for data volume and recency analysis. In: 2019 International Conference on Computing, Networking and Communications (ICNC), pp. 697–701. IEEE (2019)
4. Erichson, N.B., Muehlebach, M., Mahoney, M.W.: Physics-informed autoencoders for lyapunov-stable fluid flow prediction. arXiv preprint arXiv:1905.10866 (2019)
5. Han, Y., Chun, Q., Jin, H.: Wind-induced vibration performance of early chinese hall-style timber buildings. Journal of Wood Science **67**(1), 1–18 (2021)
6. Hornik, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. Neural networks **2**(5), 359–366 (1989)
7. Islama, M., Nasrinb, S.: Dusty fluid flow past between two parallel riga plates embedded in a porous medium (2020)
8. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks (2016)
9. Kim, B., Azevedo, V.C., Thuerey, N., Kim, T., Gross, M., Solenthaler, B.: Deep fluids: A generative network for parameterized fluid simulations. In: Computer Graphics Forum, vol. 38, pp. 59–70. Wiley Online Library (2019)
10. Lye, K.O., Mishra, S., Ray, D.: Deep learning observables in computational fluid dynamics. Journal of Computational Physics **410**, 109339 (2020)
11. Major, R., Kopernik, M., Kuźmińska, A., Imbir, G., Plutecka, H., Pomorska, M., Ciach, T., Lackner, J.M.: In vitro haemocompatibility assessment of acrylic acid deposited on solid, polyurethane substrate. Colloids and Surfaces B: Biointerfaces **199**, 111562 (2021)
12. Mao, S., Rajan, D., Chia, L.T.: Deep residual pooling network for texture recognition. Pattern Recognition **112**, 107817 (2021)
13. Noé, F., Olsson, S., Köhler, J., Wu, H.: Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning. Science **365**(6457) (2019)
14. Raissi, M., Perdikaris, P., Karniadakis, G.: Physics informed deep learning (part ii): Data-driven, discovery of nonlinear partial differential equations,", arxiv e-prints, p. arXiv preprint arXiv:1711.10566 (2017)
15. Raissi, M., Perdikaris, P., Karniadakis, G.E.: Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. arXiv preprint arXiv:1711.10561 (2017)
16. Raissi, M., Yazdani, A., Karniadakis, G.E.: Hidden fluid mechanics: A navier-stokes informed deep learning framework for assimilating flow visualization data. arXiv preprint arXiv:1808.04327 (2018)
17. Raissi, M., Yazdani, A., Karniadakis, G.E.: Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. Science **367**(6481), 1026–1030 (2020)
18. Ramachandran, P., Zoph, B., Le, Q.V.: Searching for activation functions. arXiv preprint arXiv:1710.05941 (2017)
19. Rao, C., Sun, H., Liu, Y.: Physics-informed deep learning for incompressible laminar flows. Theoretical and Applied Mechanics Letters **10**(3), 207–212 (2020)
20. Salimans, T., Kingma, D.P.: Weight normalization: A simple reparameterization to accelerate training of deep neural networks. arXiv preprint arXiv:1602.07868 (2016)
21. Saurav, S., Saini, R., Singh, S.: Emnet: a deep integrated convolutional neural network for facial emotion recognition in the wild. Applied Intelligence pp. 1–28 (2021)
22. Shorten, C., Khoshgoftaar, T.M.: A survey on image data augmentation for deep learning. Journal of Big Data **6**(1), 1–48 (2019)
23. Subramaniam, A., Wong, M.L., Borker, R.D., Nimmagadda, S., Lele, S.K.: Turbulence enrichment using physics-informed generative adversarial networks. arXiv e-prints pp. arXiv–2003 (2020)
24. Wandel, N., Weinmann, M., Klein, R.: Fast fluid simulations in 3d with physics-informed deep learning. arXiv preprint arXiv:2012.11893 (2020)
25. Xu, C., Wang, H., Wu, S., Lin, Z.: Treelstm with tag-aware hypernetwork for sentence representation. Neurocomputing **434**, 11–20 (2021)
26. Xu, H., Zhang, W., Wang, Y.: Explore missing flow dynamics by physics-informed deep learning: the parameterised governing systems. arXiv preprint arXiv:2008.12266 (2020)
27. Yang, Z., Wu, J.L., Xiao, H.: Enforcing deterministic constraints on generative adversarial networks for emulating physical systems. arXiv preprint arXiv:1911.06671 (2019)
28. Yi, X., Duan, Z., Li, R., Zhang, J., Li, T., Zheng, Y.: Predicting fine-grained air quality based on deep neural networks. IEEE Transactions on Big Data (2020)