

The Deep Learning Galerkin Method for the General Stokes Equations

Jian Li^{a,*}, Jing Yue^b, Wen Zhang^a, Wansuo Duan^c

^aDepartment of Mathematics, Shaanxi University of Science and Technology, Xi'an

^bSchool of electrical and control engineering, Shaanxi University of Science and Technology, Xi'an

^cInstitute of Atmospheric Physics, Chinese Academy of Sciences, Beijing

Abstract

The finite element method, finite difference method, finite volume method and spectral method have achieved great success in solving partial differential equations. However, the high accuracy of traditional numerical methods is at the cost of high efficiency. Especially in the face of high-dimensional problems, the traditional numerical methods are often not feasible in the subdivision of high-dimensional meshes and the differentiability and integrability of high-order terms. In deep learning, neural network can deal with high-dimensional problems by adding the number of layers or expanding the number of neurons. Compared with traditional numerical methods, it has great advantages. In this article, we consider the Deep Galerkin Method (DGM) for solving the general Stokes equations by using deep neural network without generating mesh grid. The DGM can reduce the computational complexity and achieve the competitive results. Here, depending on the L^2 error we construct the objective function to control the performance of the approximation solution. Then, we prove the convergence of the objective function and the convergence of the neural network to the exact solution. Finally, the effectiveness of the proposed framework is demonstrated through some numerical experiments.

Keywords: general Stokes equations; Deep Galerkin Method; convergence; neural network; deep learning.

2020 MSC: 00-01, 99-00

1. Introduction

Partial Differential Equations (PDEs) can mathematically model and describe certain objective laws in the fields of physical chemistry, finance, natural phenomenon and engineering technology *et al.* However, most of them are difficult to obtain the analytical solution. Consequently, numerical methods such as finite element method have been flourishing in the past decades for modeling mechanics problems via solving PDEs [1]. Alternatively, the other methods, just like generalized finite element basis functions [2] and construction of multiple difference schemes [3] have broad applications in the same way. Although these methods are well used in PDEs and achieved good results, almost all of them have obvious drawbacks, complexity in general problems and no longer apparent since lots of mesh grid are generated especially for high dimensional problems. Besides, there has many large problems in computational fluid dynamics, such as uncertainty quantification, Bayesian inversion, data assimilation and constrained optimization of partial differential equations, which are considered to be very challenging because they require a large number of numerical solutions of the corresponding PDEs.

Inspired by machine learning, the deep learning method can learn the parameters of neural network from the sampled data which can avoid mesh generation to some certain extent. Deep

learning method has certain adaptability for unknown data, guarantee the high accuracy through training the models and currently gains a lot of interests for efficiently solving PDEs. It has been considered in various forms previously since the 1990s. Cellular Neural Network and Distributed Parameter Neural Network are used for one-dimensional PDEs [4, 5, 6]. Apart from these, single layer chebyshev neural network [7], recurrent neural network and ansatz method [8, 9] can also solve the PDEs similarly. More generally, Sun *et al.* [10, 11] used Bernstein neural network and extreme learning machine to solve first and second order ordinary differential equations and elliptic PDEs.

Due to the rapid development of computer and gradient optimization methods in recent decades, many approaches for solving high-dimensional problems are actively proposed based on deep learning techniques. Raissi *et al.* [12] solved the Black-Scholes-Barenblatt and Hamilton-Jacobi-Bellman equations, both in 100 dimensions. Besides, they proposed and developed a typical method physics-informed neural networks which combines observed data with PDE models [13, 14, 15] in many problems [16, 17, 18, 19, 20, 21]. As for higher-dimensional parametric PDEs system, Sirignano and Spiliopoulos proposed a continuous time stochastic gradient descent method [22] and DGM [23] for PDEs both in 200 dimensions. They applied a deep neural network instead of a linear combination of basis functions. Recently, Zhu *et al.* and Xu *et al.* [24, 25, 26, 27] proposed Bayesian deep convolutional encoder-decoder network and the combination of genetic algorithm and adaptive method to solve the problems with high-dimensional

*Corresponding author(Jian Li) email: jianli@sust.edu.cn

Partly supported by the NSF of China 11771259 and Shaanxi special support plan for regional development of talents. Department of Mathematics, Shaanxi University of Science and Technology, Xi'an 710021, China.

random inputs and sparse noisy data. As we all know, there has mathematical guarantees called universal approximation theorems [28] which stating that a single layer neural network could approximate many functions in Sobolev spaces. It still lack theoretical method to explain the effectiveness of multilayer neural networks.

In this paper, the DGM is first applied to solve the general d -dimensional incompressible Stokes problems, which is trained on batches of randomly sampled points satisfying the differential operator, initial condition and boundary condition without generating mesh grid. The optimal solution is obtained by using the stochastic gradient descent method instead of a linear combination of basic functions. In particular, this method overcomes the infeasibility and limitations of the traditional numerical methods especially for the high dimensional incompressible Stokes equations. Based on the objective function, the DGM numerically manifests the efficiency and flexibility. Moreover, we prove the convergence of the objective function and the convergence of the neural network and the exact solution.

The rest of the paper is developed into four sections. In the next section, we provide the preliminaries of methodology. In Section III, we prove the convergence of the objective function and the convergence of the neural network to the exact solution. In Section IV, numerical examples demonstrate the efficiency of the proposed framework and justify our theoretical analysis. Finally, we summarize our paper with a short discussion.

2. Methodology

Let Ω be a bounded, compact and open subset of \mathbb{R}^d ($d = 2, 3, \dots$). With regular boundary $\partial\Omega \subset \mathbb{R}^{d-1}$. We consider the general Stokes equations with Dirichlet boundary condition.

$$\alpha u - \nu \nabla^2 u + \nabla p = f, \quad \text{in } \Omega, \quad (1)$$

$$\nabla \cdot u = 0, \quad \text{in } \Omega, \quad (2)$$

$$u = g, \quad \text{on } \partial\Omega, \quad (3)$$

where $\alpha > 0$ is a positive constant, ν denotes the viscosity coefficient, u and p represent velocity and pressure respectively, f and g are source terms. For notational brevity, we set $\bar{u} = (u, p)$ and define

$$\mathcal{G}[\bar{u}] = \alpha u - \nu \nabla^2 u + \nabla p - f. \quad (4)$$

Here, we recall the classical Sobolev spaces

$$H^k(\Omega) = \{v \in L^2(\Omega) : D_w^\alpha v \in L^2(\Omega), \forall \alpha : |\alpha| \leq k\},$$

$$H_0^k(\Omega) = \{v \in H^k(\Omega) : v|_{\partial\Omega} = 0\},$$

and their norm

$$\|v\|_k = \sqrt{(v, v)_k} = \left\{ \sum_{|\alpha|=0}^k \int_{\Omega} (D_w^\alpha v)^2 dx \right\}^{\frac{1}{2}},$$

where $k > 0$ is a positive integer, $D_w^\alpha v$ is the generalized derivative of v , and (\cdot, \cdot) represents the inner product.

In order to obtain a well-posedness of the general Stokes equations, we have the following result.

Lemma 2.1. Assume that Ω is a bounded and connected open subset of \mathbb{R}^d with a Lipschitz-continuous boundary Γ , $f \in [L^2(\Omega)]^d$ and $g \in [H^{1/2}(\Gamma)]^d$ such that

$$\int_{\Gamma} g \cdot \vec{n} ds = 0,$$

there exists a unique pair $\bar{u} \in [H_0^1(\Omega)]^d \times L_0^2(\Omega)$ of the general Stokes equations (1)-(3). Furthermore, we have

$$\|u\|_2 + \|p\|_1 \leq C(\|f\|_{-1} + \|g\|_{3/2, \partial\Omega}). \quad (5)$$

Generally, assuming that $\bar{U} = (U(\mathbf{x}; \theta_1), P(\mathbf{x}; \theta_2))$ is the neural network solution to the general Stokes equations (1)-(3), θ_1 and θ_2 are the stacked components of the neural network's parameters θ for velocity and pressure respectively. Define the objective function

$$\begin{aligned} J(\bar{U}) = & \left\| \mathcal{G}[\bar{U}](x; \theta) - \mathcal{G}[\bar{u}](x) \right\|_{0, \Omega, \omega_1}^2 + \left\| \nabla \cdot U(x; \theta_1) \right\|_{0, \Omega, \omega_1}^2 \\ & + \left\| U(x; \theta_1) - g(x) \right\|_{0, \partial\Omega, \omega_2}^2. \end{aligned} \quad (6)$$

It should be noted that $J(\bar{U})$ can measure how well the approximate solution satisfies differential operator, divergence condition and boundary condition. Notice that

$$\|f(y)\|_{0, \mathcal{Y}, \omega} = \int_{\mathcal{Y}} |f(y)|^2 \omega(y) dy,$$

where $\omega(y)$ is the probability density of y in \mathcal{Y} .

Our goal is to find the parameters θ such that \bar{U} minimizes the objective function $J(\bar{U})$. Especially, if $J(\bar{U}) = 0$ then \bar{U} is the solution to the general Stokes equations (1)-(3). However, it is computationally infeasible to estimate θ by directly minimizing $J(\bar{U})$ when integrated over a higher dimensional region. Here, we apply a sequence of random sampled points from Ω and $\partial\Omega$ to avoid forming mesh grid. The algorithm of the DGM for the general Stokes equations are presented as Algorithm 1.

In this process, the "learning rate" $\alpha_n \in (0, 1)$ decreases as $n \rightarrow \infty$. The term $\nabla_{\theta} G(\theta_n, s_n)$ is unbiased estimate of $\nabla_{\theta} J(\bar{U}(\cdot; \theta_n))$ because we can estimate the population parameters by sample mathematical expectations such as

$$\mathbb{E}[\nabla_{\theta} G(\theta_n, s_n) | \theta_n] = \nabla_{\theta} J(\bar{U}(\cdot; \theta_n)). \quad (7)$$

In order to illustrate more vividly, the flowchart displayed in Figure 1.

3. Convergence

Undoubtedly, the objective function $J(\bar{U})$ can measure how well the neural network \bar{U} satisfies the differential operator, boundary condition and divergence condition. As we known from [28], if there is only one hidden layer and one output, then

Algorithm 1: Deep Learning Galerkin Method

Input: $s_n = (x_n, r_n)$, Max Iterations M , learning rate α_n

Output: θ_{n+1}

Randomly generated sample points $s_n = (x_n, r_n)$;

Initialize the parameters θ ;

while iterations $\leq M$ **do**

 read current;

$$G(\theta_n, s_n) = \left(\mathcal{G}[\bar{U}](x_n; \theta) \right)^2 + (\nabla \cdot U(x_n; \theta_1))^2 + (U(r_n; \theta_1) - g(x))^2$$

and

$$\theta_{n+1} = \theta_n - \alpha_n \nabla_\theta G(\theta_n, s_n).$$

if $\lim_{n \rightarrow \infty} \|\nabla_\theta G(\theta_n, s_n)\| = 0$ **then**

 return the parameters θ_{n+1} ;

else

 go back to the beginning of current section;

the set of all functions implemented by such a network with m and n hidden units for velocity and pressure are

$$[\mathfrak{C}_u^m(\varphi)]^d = \left\{ \Phi(x) : \mathbb{R}^d \mapsto \mathbb{R}^d \mid \Phi_\ell(x) = \sum_{i=1}^m \beta_i \varphi \left(\sum_{j=1}^d \sigma_{ji} x_j + c_i \right) \right\},$$

and

$$\mathfrak{C}_p^n(\psi) = \left\{ \Psi(x) : \mathbb{R}^d \mapsto \mathbb{R} \mid \Psi(x) = \sum_{i=1}^n \beta'_i \psi \left(\sum_{j=1}^d \sigma'_{ji} x_j + c'_i \right) \right\},$$

where $\ell = 1, 2, \dots, d$, $\Phi(x) = (\Phi_1(x), \Phi_2(x), \dots, \Phi_d(x))$, φ and ψ are the shared activation functions of the hidden units in $C^2(\Omega)$, bounded and non-constant. x_j is input, $\beta_i, \beta'_i, \sigma_{ji}$ and σ'_{ji} are weights, c_i and c'_i are thresholds of neural network.

More generally, we still use the similar notation

$$[\mathfrak{C}_u(\varphi)]^d \times \mathfrak{C}_p(\psi)$$

for the multilayer neural networks with an arbitrarily large number of hidden units m and n respectively. In particular, we use the same parameters $\beta_i, \sigma_{ji}, c_i$ and common activation function φ in each dimension of $[\mathfrak{C}_u^m(\varphi)]^d$. The parameters can be written as follows

$$\begin{aligned} \theta_1^\ell &= (\beta_1, \dots, \beta_m, \sigma_{11}, \dots, \sigma_{dm}, c_1, \dots, c_m), \\ \theta_2 &= (\beta'_1, \dots, \beta'_n, \sigma'_{11}, \dots, \sigma'_{dn}, c'_1, \dots, c'_n), \end{aligned} \quad (8)$$

where $\ell = 1, 2, \dots, d$, $\theta_1 \in \mathbb{R}^{(2+d)md}$ and $\theta_2 \in \mathbb{R}^{(2+d)n}$.

In this section, we show that the neural network \bar{U}^n with n hidden units for U and P satisfies the differential operator, boundary condition and divergence condition arbitrarily well for sufficiently large n . Moreover, we prove that there exists

$\bar{U}^n \in [\mathfrak{C}_u^n(\varphi)]^d \times \mathfrak{C}_p^n(\psi)$ such that $J(\bar{U}^n) \rightarrow 0$ as $n \rightarrow \infty$. Another significant consideration, we give the convergence of $\bar{U}^n \rightarrow \bar{u}$ as $n \rightarrow \infty$ where \bar{u} is the exact solution to the general Stokes equations (1)-(3).

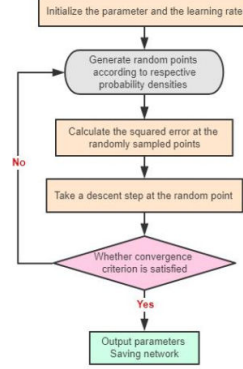


Figure 1: Flowchart of the DGM for the general Stokes equations.

3.1. Convergence of the objective function $J(\bar{U})$

A particularly important processing, we use the multilayer feed forward networks \bar{U} to universally approximate the solution to the general Stokes equations. Certainly, the neural network \bar{U} can make the objective function $J(\bar{U})$ arbitrarily small. Thus, using the results of [28] and the following lemma, we obtain the convergence of the objective function $J(\bar{U})$. First, we give the following assumption.

Lemma 3.1. Assume that $\nabla u(x)$, $\Delta u(x)$ and $\nabla p(x)$ are locally Lipschitz with Lipschitz coefficient that they have at most polynomial growth on $u(x)$ and $p(x)$. Then, for some constants $0 \leq q_i \leq \infty (i = 1, 2, 3, 4)$ we have

$$|\Delta U - \Delta u| \leq (|\nabla U|^{q_1/2} + |\nabla u|^{q_2/2}) |\nabla U - \nabla u|, \quad (9)$$

$$|\nabla P - \nabla p| \leq (|P|^{q_3/2} + |p|^{q_4/2}) |P - p|. \quad (10)$$

Theorem 3.1. Under the assumption of Lemma 3.1, there exists a neural network $\bar{U} \in [\mathfrak{C}_u(\varphi)]^d \times \mathfrak{C}_p(\psi)$, satisfying

$$J(\bar{U}) \leq C\epsilon, \quad \forall \epsilon > 0, \quad (11)$$

where C depends on the data $\{\Omega, \alpha, \nu, \omega_1, \omega_2, f\}$.

Proof. By Theorem 3 of [28], we can conclude that there exists $\bar{U} \in [\mathfrak{C}_u(\varphi)]^d \times \mathfrak{C}_p(\psi)$ which are uniformly 2-dense on compacts of $C^2(\bar{\Omega}) \times C^1(\bar{\Omega})$. It means that for $\bar{u} \in C^2(\bar{\Omega}) \times C^1(\bar{\Omega})$, $\forall \epsilon > 0$, it follows that

$$\max_{a \leq 2} \sup_{x \in \Omega} |\partial_x^a U(x) - \partial_x^a u(x)| < \epsilon, \quad (12)$$

$$\sup_{x \in \Omega} |P(x) - p(x)| < \epsilon. \quad (13)$$

According to the Lemma 3.1, using the Hölder inequality and Young inequality, setting r_1 and r_2 are conjugate numbers such

that $\frac{1}{r_1} + \frac{1}{r_2} = 1$, we find that

$$\begin{aligned}
& \int_{\Omega} |\Delta U - \Delta u|^2 d\omega_1(x) \\
& \leq \int_{\Omega} (|\nabla U|^{q_1} + |\nabla u|^{q_2}) (\nabla U - \nabla u)^2 d\omega_1(x) \\
& \leq \left[\int_{\Omega} (|\nabla U|^{q_1} + |\nabla u|^{q_2})^{r_1} d\omega_1(x) \right]^{1/r_1} \\
& \quad \times \left[\int_{\Omega} (\nabla U - \nabla u)^{2r_2} d\omega_1(x) \right]^{1/r_2} \\
& \leq \left[\int_{\Omega} (|\nabla U - \nabla u|^{q_1} + |\nabla u|^{q_1 \vee q_2})^{r_1} d\omega_1(x) \right]^{1/r_1} \\
& \quad \times \left[\int_{\Omega} (\nabla U - \nabla u)^{2r_2} d\omega_1(x) \right]^{1/r_2} \\
& \leq C\epsilon^2,
\end{aligned} \tag{14}$$

where $q_1 \vee q_2 = \max\{q_1, q_2\}$.

Similarly,

$$\begin{aligned}
& \int_{\Omega} |\nabla P - \nabla p|^2 d\omega_1(x) \\
& \leq \int_{\Omega} (|P|^{q_3} + |p|^{q_4}) (P - p)^2 d\omega_1(x) \\
& \leq \left[\int_{\Omega} (|P|^{q_3} + |p|^{q_4})^{r_3} d\omega_1(x) \right]^{1/r_3} \\
& \quad \times \left[\int_{\Omega} (P - p)^{2r_4} d\omega_1(x) \right]^{1/r_4} \\
& \leq \left[\int_{\Omega} (|P - p|^{q_3} + |p|^{q_3 \vee q_4})^{r_3} d\omega_1(x) \right]^{1/r_3} \\
& \quad \times \left[\int_{\Omega} (P - p)^{2r_4} d\omega_1(x) \right]^{1/r_4} \\
& \leq C\epsilon^2,
\end{aligned} \tag{15}$$

where $\frac{1}{r_3} + \frac{1}{r_4} = 1$ and $q_3 \vee q_4 = \max\{q_3, q_4\}$.

For the boundary condition, we have

$$\int_{\partial\Omega} |U - u|^2 d\omega_2(x) \leq C\epsilon^2. \tag{16}$$

Thanks to (14)-(16), we obtain

$$\begin{aligned}
J(\bar{U}) &= \|\mathcal{G}[\bar{U}](x; \theta) - \mathcal{G}[\bar{u}](x)\|_{\Omega, \omega_1}^2 \\
& \quad + \|\nabla \cdot U(x; \theta)\|_{\Omega, \omega_1}^2 + \|U(x; \theta) - g(x)\|_{\partial\Omega, \omega_2}^2 \\
&= \|\mathcal{G}[\bar{U}](x; \theta)\|_{\Omega, \omega_1}^2 \\
& \quad + \|\nabla \cdot U(x; \theta)\|_{\Omega, \omega_1}^2 + \|U(x; \theta) - g(x)\|_{\partial\Omega, \omega_2}^2 \\
&= \int_{\Omega} |\Delta U - \Delta u|^2 d\omega_1(x) + \int_{\Omega} |\nabla P - \nabla p|^2 d\omega_1(x) \\
& \quad + \int_{\Omega} |\alpha U - \alpha u|^2 d\omega_1(x) + \int_{\Omega} |\nabla \cdot u|^2 d\omega_1(x) \\
& \quad + \int_{\Omega} |\nabla \cdot (U - u)|^2 d\omega_1(x) + \int_{\partial\Omega} |U - u|^2 d\omega_2(x) \\
& \leq C\epsilon^2,
\end{aligned} \tag{17}$$

which implies (11). \square

3.2. Convergence of the neural network to the general Stokes solution

We have discussed the convergence of the objective function $J(\bar{U})$ in the last subsection. Next we give the convergence of the neural network \bar{U}^n to the exact solution \bar{u} for the general Stokes equations with homogeneous boundary condition

$$\alpha u - \nu \nabla^2 u + \nabla p = f, \quad \text{in } \Omega, \tag{18}$$

$$\nabla \cdot u = 0, \quad \text{in } \Omega, \tag{19}$$

$$u = 0, \quad \text{on } \partial\Omega. \tag{20}$$

Recall the form of the objective function with

$$J(\bar{U}) = \|\mathcal{G}[\bar{U}]\|_{0, \Omega}^2 + \|\nabla \cdot U\|_{0, \Omega}^2 + \|U\|_{0, \partial\Omega}^2.$$

By Theorem 3.1, we obtain

$$J(\bar{U}^n) \rightarrow 0 \quad \text{as } n \rightarrow \infty.$$

Furthermore, each neural network $\bar{U}^n = (U^n, P^n)$ satisfies the following equations

$$\mathcal{G}[\bar{U}^n] = h^n, \quad \text{in } \Omega, \tag{21}$$

$$\nabla \cdot U^n = 0, \quad \text{in } \Omega, \tag{22}$$

$$U^n = g^n, \quad \text{on } \partial\Omega, \tag{23}$$

for some h^n and g^n such that

$$\|h^n\|_{0, \Omega}^2 + \|g^n\|_{0, \partial\Omega}^2 \rightarrow 0 \quad \text{as } n \rightarrow \infty. \tag{24}$$

In this subsection, we do not explore more discussions on inhomogeneous problems since the inhomogeneous problems can be solved by the corresponding homogeneous method (See Section 4 of Chapter V in [29] or Chapter 8 of [30] for details). For convenience, we provide a theorem to guarantee the convergence of the neural network \bar{U}^n and the exact solution \bar{u} to the equations (18)-(20).

Theorem 3.2. *Under the assumptions of Lemma 3.1, Theorem 3.1 and (24), the neural network U^n can converge strongly to u in $L^2(\Omega)$, and the P^n converges strongly to p in $H^{-1}(\Omega)$. In addition, if the sequences $\{U^n\}_{n \in \mathbb{N}}$ and $\{P^n\}_{n \in \mathbb{N}}$ are uniformly bounded and equicontinuous in Ω , they can converge to u and p uniformly in Ω respectively.*

Proof. The existence and uniqueness for the solution of (18)-(20) are proved by the Saddle point theorem (See Lemma 2.1).

Note that $\widehat{\bar{U}}^n$ satisfies the equations (21)-(23) with $g^n = 0$. Firstly, we know that the variational formulation of the equations (21)-(23) is to find $(\widehat{U}^n, \widehat{P}^n) \in [\mathfrak{C}_u^n(\varphi)]^d \times \mathfrak{C}_p^n(\psi)$ for $\forall (\widehat{V}, \widehat{Q}) \in [\mathfrak{C}_u^n(\varphi)]^d \times \mathfrak{C}_p^n(\psi)$ such that

$$\begin{aligned}
& \alpha(\widehat{U}^n, \widehat{V}) + \nu(\nabla \widehat{U}^n, \nabla \widehat{V}) + (\nabla \widehat{P}^n, \widehat{V}) + (\nabla \cdot \widehat{U}^n, \widehat{Q}) \\
& = (f, \widehat{V}) + (h^n, \widehat{V}),
\end{aligned} \tag{25}$$

in addition, we have

$$(\nabla \widehat{P}^n, \widehat{V}) = -(\nabla \cdot \widehat{V}, \widehat{P}^n). \tag{26}$$

Taking $\widehat{V} = \widehat{U}^n$ and $\widehat{Q} = \widehat{P}^n$ in (25), it follows that

$$\alpha(\widehat{U}^n, \widehat{U}^n) + \nu(\nabla \widehat{U}^n, \nabla \widehat{U}^n) = (f, \widehat{U}^n) + (h^n, \widehat{U}^n). \quad (27)$$

Using the definition of the H^1 norm, (24) and setting $\alpha_0 = \min\{\alpha, \nu\}$, we obtain

$$\begin{aligned} \alpha_0 \|\widehat{U}^n\|_{1,\Omega} &\leq C(\|f\|_{0,\Omega} + \|h^n\|_{0,\Omega}) \\ &\leq C\|f\|_{0,\Omega}. \end{aligned} \quad (28)$$

The convergence of \widehat{U}^n and \overline{U}^n is desirable to discuss yet. By using the uniformly boundedness of \widehat{U}^n , we can extract a subsequence $\{\widehat{U}^n\}_{n \in \mathbb{N}}$ of \widehat{U}^n which can converge weakly in $H^1(\Omega)$. Due to the compact embedding $H^1(\Omega) \hookrightarrow L^2(\Omega)$, we have $\lim_{n \rightarrow \infty} \|\widehat{U}^n - u\|_{0,\Omega} = 0$.

Nevertheless, we remain to discuss $\lim_{n \rightarrow \infty} \|U^n - \widehat{U}^n\|_{0,\Omega} = 0$, where U^n and \widehat{U}^n satisfy (21)-(23) with homogeneous and inhomogeneous boundary respectively. Afterwards, since $\lim_{n \rightarrow \infty} \|g^n\|_{0,\partial\Omega} = 0$, U^n converges to zero at least along a subsequence on the boundary. Besides, it will be identical with \widehat{U}^n almost everywhere. Indeed, define $F_n = |U^n - \widehat{U}^n|^2$. $\{F_n(x)\}_{n \in \mathbb{N}}$ is uniformly bounded in $L^2(\Omega)$ by the reason of the uniformly boundedness of $\{U^n\}_{n \in \mathbb{N}}$ and $\{\widehat{U}^n\}_{n \in \mathbb{N}}$. In addition, $\{F_n(x)\}_{n \in \mathbb{N}}$ can be integrated on domain $\bar{\Omega}$ and converges to zero almost everywhere. By the definition of F_n , the uniformly boundedness and equicontinuity of $\{U^n\}_{n \in \mathbb{N}}$ and $\{\widehat{U}^n\}_{n \in \mathbb{N}}$, for $\forall x, y \in \bar{\Omega}$, $\forall \epsilon' > 0$, $\exists \delta > 0$, if $|x - y| < \delta$, there holds that

$$\begin{aligned} &|F_n(x) - F_n(y)| \\ &= \left| |U^n(x) - \widehat{U}^n(x)|^2 - |U^n(y) - \widehat{U}^n(y)|^2 \right| \\ &= \left| |U^n(x) - \widehat{U}^n(x)| + |U^n(y) - \widehat{U}^n(y)| \right| \\ &\quad \times \left| |U^n(x) - \widehat{U}^n(x)| - |U^n(y) - \widehat{U}^n(y)| \right| \\ &\leq \left| |U^n(x) - \widehat{U}^n(x)| + |U^n(y) - \widehat{U}^n(y)| \right| \\ &\quad \times \left| |U^n(x) - \widehat{U}^n(x) - U^n(y) - \widehat{U}^n(y)| \right| \\ &\leq \left| |U^n(x) - \widehat{U}^n(x)| + |U^n(y) - \widehat{U}^n(y)| \right| \\ &\quad \times \left| |U^n(x) - U^n(y)| + |\widehat{U}^n(x) - \widehat{U}^n(y)| \right| \\ &< C\epsilon', \end{aligned} \quad (29)$$

where $\epsilon' > 0$ is an arbitrarily small constant. In conclusion, $\{F_n(x)\}_{n \in \mathbb{N}}$ is equicontinuous. Based on the above preparation, we can obtain $\lim_{n \rightarrow \infty} \|U^n - \widehat{U}^n\|_{0,\Omega} = 0$ by using Vitali's theorem. Thus through a triangle inequality there holds that

$$\begin{aligned} &\lim_{n \rightarrow \infty} \|U^n - u\|_{0,\Omega} \\ &\leq \lim_{n \rightarrow \infty} \|U^n - \widehat{U}^n\|_{0,\Omega} + \lim_{n \rightarrow \infty} \|\widehat{U}^n - u\|_{0,\Omega} \\ &= 0 \end{aligned} \quad (30)$$

since $\{\widehat{U}^n\}_{n \in \mathbb{N}}$ strongly converges to u in $L^2(\Omega)$.

In order to study the pressure of the general Stokes problem, we define

$$\begin{aligned} L(\widehat{v}) &= (f, \widehat{v}) + (h^n, \widehat{v}) - \alpha(\widehat{U}^n, \widehat{v}) - \nu(\nabla \widehat{U}^n, \nabla \widehat{v}) \\ &= 0, \end{aligned} \quad (31)$$

where $\widehat{v} \in [\mathfrak{C}_u^n(\varphi)]^d \cap [H_0^1(\Omega)]^d$. Then

$$\langle L, \widehat{v} \rangle = 0, \quad \forall \widehat{v} \in [\mathfrak{C}_u^n(\varphi)]^d \cap [H_0^1(\Omega)]^d,$$

where $\langle \cdot, \cdot \rangle$ stands for the duality pairing between $[\mathfrak{C}_u^n(\varphi)]^d \cap [H_0^1(\Omega)]^d$ and its dual space.

In addition, there exists $\widehat{P}^n \in \mathfrak{C}_p^n(\psi) \cap L^2(\Omega)$, for $\forall \widehat{v} \in [\mathfrak{C}_u^n(\varphi)]^d \cap [H_0^1(\Omega)]^d$ such that

$$\langle L, \widehat{v} \rangle = \int_{\Omega} \widehat{P}^n \operatorname{div} \widehat{v} dx = -(\widehat{P}^n, \operatorname{div} \widehat{v}) = d(\widehat{v}, \widehat{P}^n).$$

Namely,

$$d(\widehat{v}, \widehat{P}^n) = (f, \widehat{v}) + (h^n, \widehat{v}) - \alpha(\widehat{U}^n, \widehat{v}) - \nu(\nabla \widehat{U}^n, \nabla \widehat{v}). \quad (32)$$

What's more, as in Theorem 3.3 of [31], we find that

$$\nabla \widehat{U}^n \rightarrow \nabla u \text{ almost everywhere in } \Omega,$$

which concludes that \widehat{P}^n can converge weakly to p since $d(\widehat{v}, \widehat{P}^n) \rightharpoonup d(\widehat{v}, p)$. Applying the same approach as for the strong convergence of $\{\widehat{U}^n\}_{n \in \mathbb{N}}$ to u in $L^2(\Omega)$. Consequently, due to the compact embedding $L^2(\Omega) \hookrightarrow H^{-1}(\Omega)$, we can obtain

$$\lim_{n \rightarrow \infty} \|\widehat{P}^n - p\|_{-1,\Omega} = 0.$$

Using a triangle inequality, it follows that

$$\begin{aligned} &\lim_{n \rightarrow \infty} \|P^n - p\|_{-1,\Omega} \\ &\leq \lim_{n \rightarrow \infty} \|P^n - \widehat{P}^n\|_{-1,\Omega} + \lim_{n \rightarrow \infty} \|\widehat{P}^n - p\|_{-1,\Omega} \\ &= 0. \end{aligned} \quad (33)$$

For all these reasons, $\{U^n\}_{n \in \mathbb{N}}$ can converge strongly to u in $L^2(\Omega)$, $\{P^n\}_{n \in \mathbb{N}}$ converges strongly to p in $H^{-1}(\Omega)$. Noting that $\{U^n\}_{n \in \mathbb{N}}$ and $\{P^n\}_{n \in \mathbb{N}}$ are uniformly bounded and equicontinuous in Ω , we can conclude that $\{U^n\}_{n \in \mathbb{N}}$ and $\{P^n\}_{n \in \mathbb{N}}$ converge uniformly to u and p by the well known Arzelà-Ascoli theorem. \square

4. Numerical Experiments

In this section, we apply the DGM to solve the general Stokes problems in both 2D and 3D case. The experimental results show the high efficiency and precision of the DGM. Our numerical experiments are based on Tensorflow [32] and the configuration of the computer is 64-bit Intel Xeon Silver 4116 (2 processors). In the numerical simulation, we utilize six different architectures to train the neural network and set the same number of network layers to solve U and P simultaneously. These architectures include one to six hidden layers respectively, and 16 units on each hidden layer (Denoted as ARCH 1-6). We apply ARCH 1-3 in 2D case and apply ARCH 1-6 in 3D case. The datasets in 2D case contain 1000, 2000, 4000 and 8000 samples respectively (See Figure 2, denoted as 2D-DS 1-4). And in 3D case, the datasets contain 1200, 2400, 4800 and 9600 samples respectively (See Figure 3, denoted as 3D-DS 1-4).

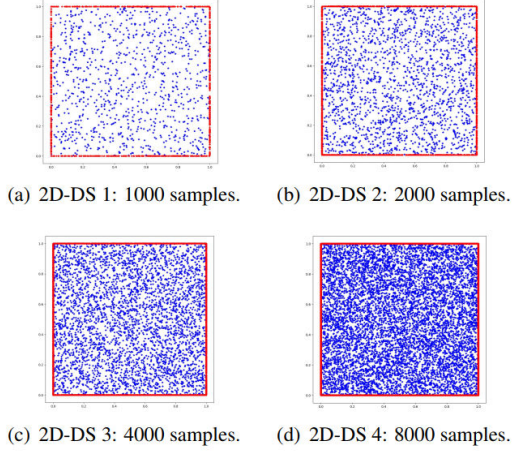


Figure 2: The datasets in 2D case.

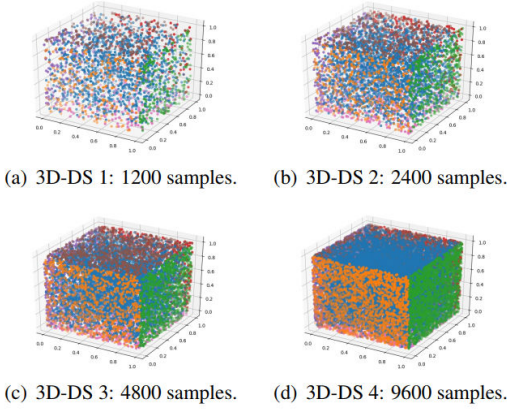


Figure 3: The datasets in 3D case.

4.1. The Stokes equations

In this subsection, we first consider the Stokes equations with homogeneous boundary condition in both 2D and 3D cases. Set $\nu = 0.025$ and $\alpha = 0$ in equations (1) - (3). Use the following 2D and 3D exact solutions,

$$\begin{aligned} u_1(x_1, x_2) &= 2\sin(\pi x_1)^2 \sin(\pi x_2) \cos(\pi x_2) \pi, \\ u_2(x_1, x_2) &= -2\sin(\pi x_1) \sin(\pi x_2)^2 \cos(\pi x_1) \pi, \\ p(x_1, x_2) &= \cos(\pi x_1) \cos(\pi x_2), \end{aligned} \quad (34)$$

in $\Omega = (0, 1)^2$ and

$$\begin{aligned} u_1(x, y, z) &= \sin(\pi x)^2 (\sin(2\pi y) \sin(\pi z)^2 - \sin(\pi y)^2 \sin(2\pi z)), \\ u_2(x, y, z) &= \sin(\pi y)^2 (\sin(2\pi z) \sin(\pi x)^2 - \sin(\pi z)^2 \sin(2\pi x)), \\ u_3(x, y, z) &= \sin(\pi z)^2 (\sin(2\pi x) \sin(\pi y)^2 - \sin(\pi x)^2 \sin(2\pi y)), \\ p(x, y, z) &= \sin(\pi x) \sin(\pi y) \cos(\pi z), \end{aligned} \quad (35)$$

in $\Omega = (0, 1)^3$. Then, the right hands $f(x, y)$ and $f(x, y, z)$ can be determined by equation (1), respectively.

In order to demonstrate the effectiveness and accuracy of the DGM, we put forward the norms as follows

$$errL^1 = \frac{1}{N} \sum_{i=1}^N |U_i - u_i|, \quad (36)$$

$$errL^2 = \frac{1}{N} \sum_{i=1}^N |U_i - u_i|^2, \quad (37)$$

$$J(\bar{U}) = \frac{1}{N} \sum_{i=1}^N [|\mathcal{G}[\bar{U}_i]|^2 + |\nabla \cdot U_i|^2 + |U_i - g_i|^2], \quad (38)$$

where \bar{U}_i and u_i are the neural network and exact solution on each batch $i = 1, 2, \dots, N$ of datasets, respectively. For simplicity, we only calculate the error of velocity. The pressure is similar.

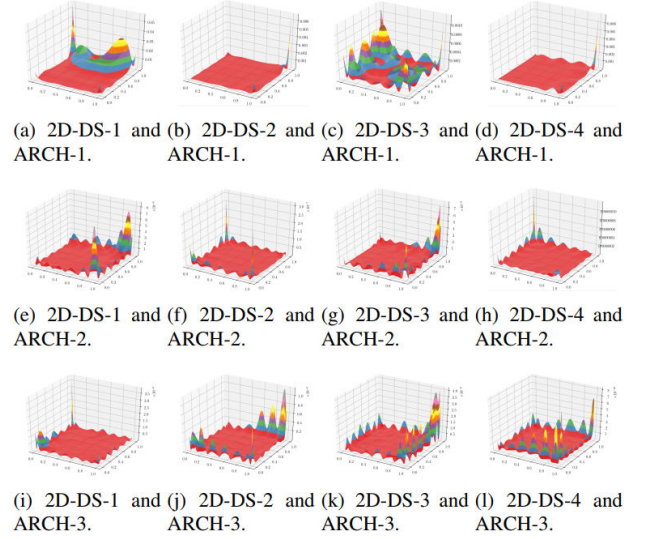


Figure 4: The $errL^2$ norm of the 2D Stokes equations.

Table 1: The performance of the DGM for the 2D Stokes equations.

ARCH 1	2D-DS-1	2D-DS-2	2D-DS-3	2D-DS-4
$errL^1$	7.82×10^{-2}	7.31×10^{-3}	1.41×10^{-2}	9.39×10^{-3}
$errL^2$	5.48×10^{-3}	4.91×10^{-5}	1.58×10^{-4}	8.23×10^{-5}
$J(\bar{U})$	1.52×10^{-2}	2.05×10^{-3}	1.75×10^{-3}	1.79×10^{-3}
ARCH 2	2D-DS-1	2D-DS-2	2D-DS-3	2D-DS-4
$errL^1$	5.45×10^{-5}	5.70×10^{-5}	4.18×10^{-5}	1.30×10^{-4}
$errL^2$	3.18×10^{-9}	3.34×10^{-9}	1.95×10^{-9}	1.60×10^{-8}
$J(\bar{U})$	9.60×10^{-8}	2.71×10^{-7}	1.02×10^{-7}	5.14×10^{-7}
ARCH 3	2D-DS-1	2D-DS-2	2D-DS-3	2D-DS-4
$errL^1$	8.09×10^{-6}	1.53×10^{-5}	1.23×10^{-5}	3.93×10^{-6}
$errL^2$	6.23×10^{-11}	2.58×10^{-10}	1.70×10^{-10}	1.91×10^{-11}
$J(\bar{U})$	1.77×10^{-8}	4.36×10^{-8}	1.15×10^{-8}	2.59×10^{-9}

Figures 4 - 5 demonstrate the $errL^2$ norm in both 2D and 3D cases ($z = 0.5$). Observed from Figures 4 - 5, the more red areas, the better performance of the algorithm. Certainly, we

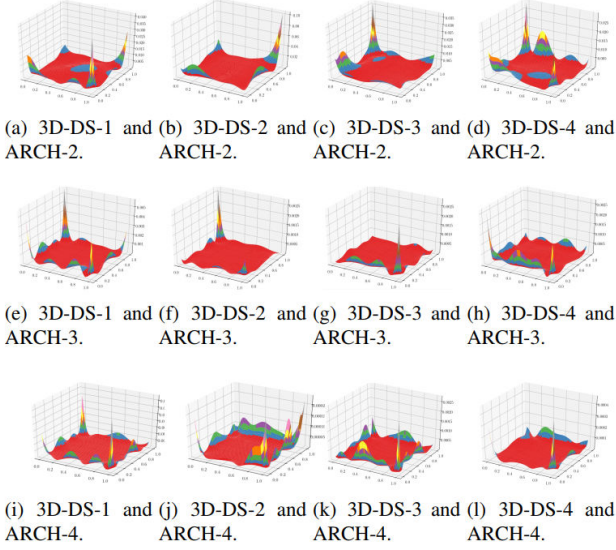


Figure 5: The $errL^2$ norm of the 3D Stokes equations.

find that the subfigures b , d in Figure 4, f , g in Figure 5 are stable than others. The numerical result of subfigure l in Figure 4 is 1.91×10^{-11} , but it does not look stable, which is probably due to the influence of boundary points and corner points. In addition, Figures 6 depict $errL^2$ norm between u and U in 2D case. Tables 1 - 2 display three norms in both 2D and 3D cases ($z = 0.5$), the best results for each ARCH are marked out. From Tables 1 - 2, we can find that numerical results with different datasets are less distinguishable. The precision of the neural network is related to the number of layers and neurons, and does not depend on the size of the datasets. The accuracy of neural networks gets better and better only as the number of hidden layers increases. In 2D case, an interesting phenomenon can be found that the best result obtained by using ARCH-3 and the network with more than three hidden layers will have over-fitting. Obviously, for high-dimensional problems, fewer layers neural network is not enough to achieve the required precision. Therefore, it is indispensable to adopt deeper layers since the non-deep neural network has great limitation for the expression of nonlinear relationship. A particularly significant consideration, there appears over-fitting phenomena by using 3D-DS-3 and ARCH-6, which shows that 6 hidden layers is enough to solve the 3D problem.

4.2. The general Stokes equations

Encouraged by positive results in previous experiments, in this subsection, we mainly consider the effect of the DGM for the general Stokes equations with homogeneous boundary condition in both 2D and 3D cases. Here, we set $\alpha = 1$, $\nu = 1$ in equations (1) - (3), and apply the analytical solutions (34) and (35) for 2D and 3D cases respectively. Consequently, the right hands $f(x, y)$ and $f(x, y, z)$ can be derived by equation (1). The plots of $errL^2$ norm are shown in Figures 7 - 8. In the same manner, the value of three different norms between the neural network and the exact solution are displayed in Tables 3 - 4.

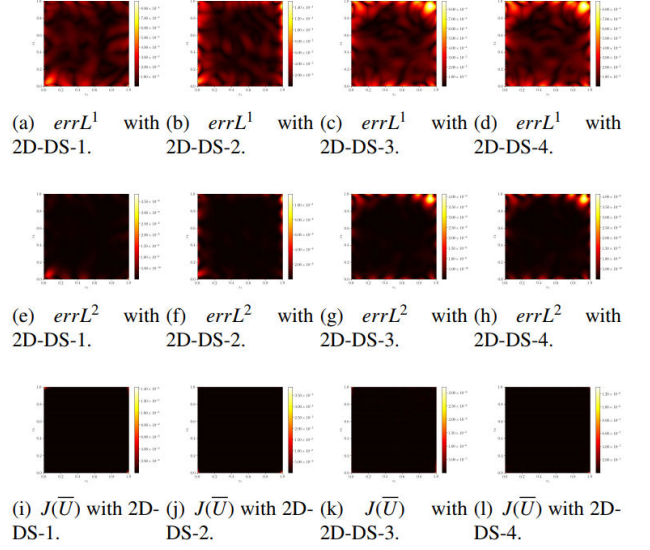


Figure 6: The three norms of the 2D Stokes equations by using ARCH-3.

Evidently, the $errL^2$ norm is going to change by using different ARCHs, as shown in Figures 7 - 8 and Tables 3 - 4. The similar results just as for the 2D case can be derived by Figures 9. In the same way, the error decreases gradually as the number of hidden layers increases, especially by using ARCH-3 and ARCH-6 in 2D and 3D case respectively.

4.3. The driven cavity flow

The driven cavity flows have been extensively applied as test cases for validating the incompressible fluid dynamics algorithm. The corner singularities for the 2D fluid flows are very important since most examples of physical interest have corners. In these two examples, we consider the 2D driven flow in a rectangular cavity when the top surface moves with a constant velocity along its length. The upper corners where the moving surface meets the stationary walls are singular points of the flow at the multi-valued horizontal velocity. The lower corners are also weakly singular points. Moreover, we also consider the 3D driven flow in a cube of unit volume, centered at $x = y = z = 0.5$ (Figure 10). A unit tangential velocity in the x direction is prescribed at the top surface, while zero velocity is prescribed on the remaining bounding surfaces in numerical examples.

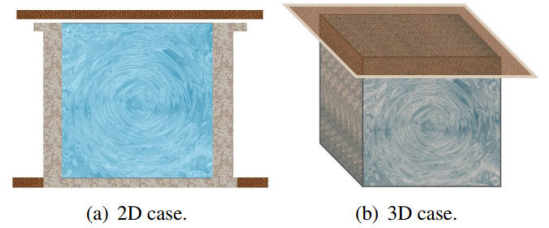


Figure 10: Diagram of 2D and 3D square cavity flow.

In order to explore how many neurons are enough to simulate the driven cavity flow, we apply 4, 8, 12 and 16 neurons for

Table 2: The performance of the DGM for the 3D Stokes equations, when $z = 0.5$.

ARCH 2	3D-DS-1	3D-DS-2	3D-DS-3	3D-DS-4
$errL^1$	6.28×10^{-2}	1.01×10^{-1}	5.91×10^{-2}	5.82×10^{-2}
$errL^2$	2.18×10^{-3}	5.53×10^{-3}	2.01×10^{-3}	1.93×10^{-3}
$J(\bar{U})$	9.59×10^{-2}	9.83×10^{-2}	5.08×10^{-2}	6.26×10^{-2}
ARCH 3	3D-DS-1	3D-DS-2	3D-DS-3	3D-DS-4
$errL^1$	1.75×10^{-2}	7.68×10^{-3}	9.69×10^{-3}	1.32×10^{-2}
$errL^2$	1.99×10^{-4}	4.14×10^{-5}	6.34×10^{-5}	1.00×10^{-4}
$J(\bar{U})$	8.89×10^{-3}	4.07×10^{-3}	2.53×10^{-3}	6.90×10^{-3}
ARCH 4	3D-DS-1	3D-DS-2	3D-DS-3	3D-DS-4
$errL^1$	1.15×10^{-2}	4.44×10^{-3}	9.69×10^{-3}	4.23×10^{-3}
$errL^2$	1.02×10^{-4}	1.31×10^{-5}	6.34×10^{-5}	1.27×10^{-5}
$J(\bar{U})$	5.49×10^{-3}	7.07×10^{-4}	1.74×10^{-3}	6.56×10^{-4}
ARCH 5	3D-DS-1	3D-DS-2	3D-DS-3	3D-DS-4
$errL^1$	2.20×10^{-3}	1.60×10^{-3}	1.63×10^{-3}	2.10×10^{-3}
$errL^2$	3.23×10^{-6}	1.91×10^{-6}	2.38×10^{-6}	3.77×10^{-6}
$J(\bar{U})$	3.49×10^{-4}	8.42×10^{-5}	9.56×10^{-5}	1.01×10^{-4}
ARCH 6	3D-DS-1	3D-DS-2	3D-DS-3	3D-DS-4
$errL^1$	1.23×10^{-3}	1.75×10^{-3}	—	9.75×10^{-4}
$errL^2$	1.33×10^{-6}	2.34×10^{-6}	—	7.10×10^{-7}
$J(\bar{U})$	2.49×10^{-4}	1.38×10^{-4}	—	3.94×10^{-5}

Table 3: The performance of the DGM for the 2D general Stokes equations.

ARCH 1	2D-DS-1	2D-DS-2	2D-DS-3	2D-DS-4
$errL^1$	2.09×10^{-1}	2.12×10^{-1}	2.84×10^{-1}	2.97×10^{-1}
$errL^2$	3.65×10^{-2}	3.05×10^{-2}	7.02×10^{-2}	7.87×10^{-2}
$J(\bar{U})$	1.4×10^{-1}	7.74×10^{-2}	2.38×10^{-1}	2.30×10^{-1}
ARCH 2	2D-DS-1	2D-DS-2	2D-DS-3	2D-DS-4
$errL^1$	4.70×10^{-4}	3.86×10^{-4}	4.98×10^{-4}	1.81×10^{-4}
$errL^2$	2.13×10^{-7}	1.54×10^{-7}	2.43×10^{-7}	3.69×10^{-8}
$J(\bar{U})$	1.68×10^{-5}	1.70×10^{-5}	9.57×10^{-6}	2.06×10^{-6}
ARCH 3	2D-DS-1	2D-DS-2	2D-DS-3	2D-DS-4
$errL^1$	1.25×10^{-4}	8.99×10^{-5}	3.71×10^{-4}	1.88×10^{-4}
$errL^2$	1.78×10^{-8}	8.08×10^{-9}	1.33×10^{-7}	3.79×10^{-8}
$J(\bar{U})$	3.04×10^{-6}	1.94×10^{-6}	2.20×10^{-6}	1.48×10^{-6}

testing. From Figure 11, we can find that the 2D driven cavity flow is stablest when using 16 neurons. Moreover, the optimal models for 2D and 3D cases are obtained by training the existing 2D-DS and 3D-DS respectively. Then, we test the different neural networks by using 1600 data points in 2D case (Figure 12). In like wise, the better results are only related to the number of hidden layers. Especially, the results by using ARCH 3 are in perfect agreement with the physical significance since ARCH 3 has more hidden layers than others. Furthermore, the optimal framework for the 3D case obtained by using 3D-DS-3 and ARCH-2. As shown in Figures 13 - 14, we utilize 1000 and 8000 data points for testing and intercepts the top view along 3 axis respectively. The numerical results indicate that the DGM is efficient and accurate.

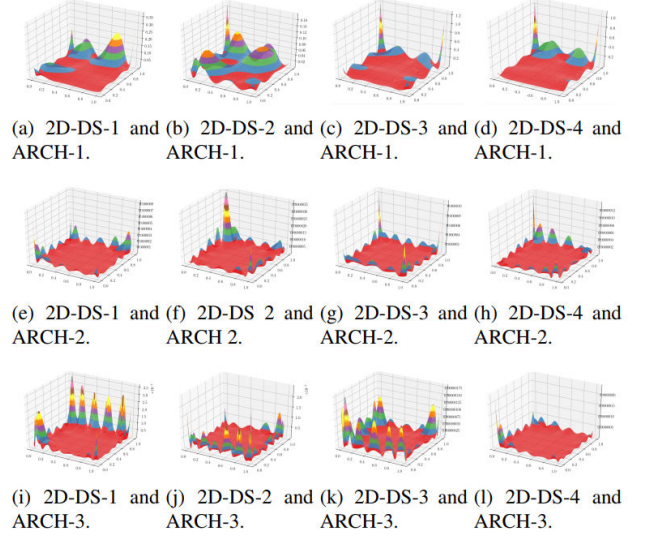


Figure 7: The $errL^2$ norm of the 2D general Stokes equations.

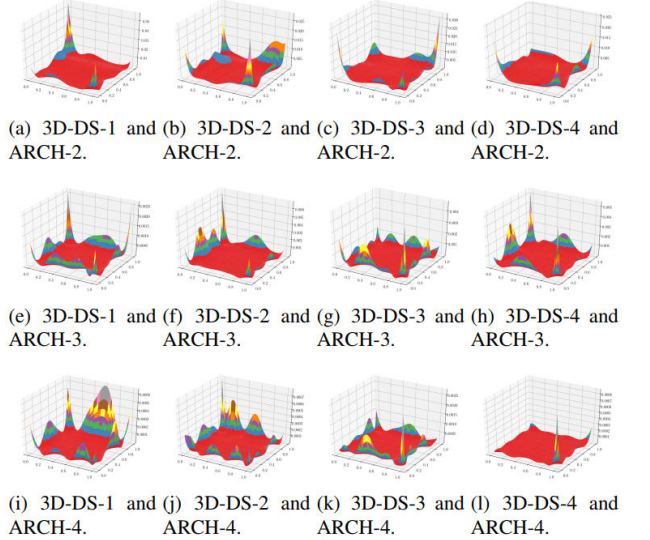


Figure 8: The $errL^2$ norm of the 3D general Stokes equations.

5. Conclusions

This paper applies the DGM to solve the general Stokes problems in both 2D and 3D cases with high efficiency and accuracy, which can transform the traditional grid mesh method into a grid free algorithm by using the random sampled data. Besides, we set the objective function appropriately to convert the constrained problem into an unconstrained problem in the sense and give two theorems to ensure the convergence of the objective function and the convergence of the neural network to the exact solution. In general, this method is based on drawing random sampled points from the domain, which can be readily extended to arbitrary domains; triangulation of the domain is not needed. The numerical results fully demonstrate the convergence properties of the DGM completely. But, we need more deliberation on the elements which have great impact in exper-

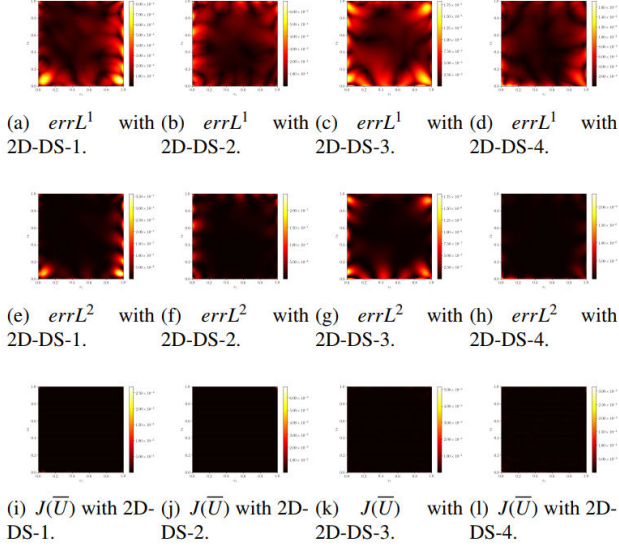


Figure 9: The three norms of the 2D general Stokes equations by using ARCH-3.

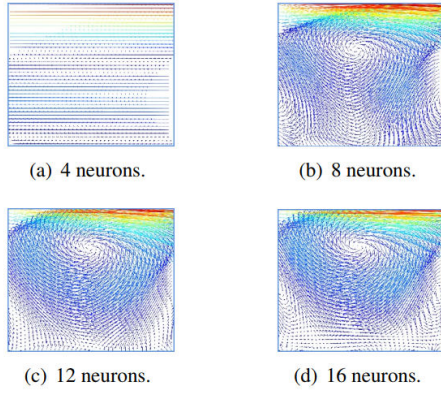


Figure 11: The 2D driven cavity flow with different neurons and ARCH-1.

iments. For example, how to construct the most suitable objective function for measuring loss and applied to optimization, whether the deeper network can get better results and randomness have a positive effect on the algorithm.

References

References

- [1] Q. Wang, D. Cheng, Numerical solution of damped nonlinear Klein-Gordon equations using variational method and finite element approach[J]. Applied Mathematics and Computation. 2005, 162(1): 381-401.
- [2] A. Pels, R. V. Sabariego, S. Schops, Solving multirate partial differential equations using hat finite element basis functions[C]. IEEE Conference on Electromagnetic Field Computation: 2016, 10.1109/2016.7816348.
- [3] G. Zhao, K. Jie, J. Liu, A New Difference Scheme for Hyperbolic Partial Differential Equations[C] International Conference on Computational Intelligence and Security. IEEE, 2018.10.1109/CIS.2017.00102.
- [4] F. Dazheng, B. Zheng, J. Licheng, Distributed parameter neural networks for solving partial differential equations[J]. Journal of Electronics (China). 1997, 14(2): 186-190.

Table 4: The performance of the DGM for the 3D general Stokes equations, when $z = 0.5$.

ARCH 2	3D-DS-1	3D-DS-2	3D-DS-3	3D-DS-4
$errL^1$	6.73×10^{-2}	6.19×10^{-2}	4.67×10^{-2}	4.15×10^{-2}
$errL^2$	2.52×10^{-3}	1.89×10^{-3}	1.26×10^{-3}	1.06×10^{-3}
$J(\bar{U})$	6.36×10^{-2}	4.89×10^{-2}	4.69×10^{-2}	5.16×10^{-2}
ARCH 3	3D-DS-1	3D-DS-2	3D-DS-3	3D-DS-4
$errL^1$	1.38×10^{-2}	1.80×10^{-2}	1.53×10^{-2}	1.52×10^{-2}
$errL^2$	1.30×10^{-4}	2.29×10^{-4}	1.67×10^{-4}	1.81×10^{-4}
$J(\bar{U})$	8.96×10^{-3}	3.17×10^{-3}	4.19×10^{-3}	3.39×10^{-3}
ARCH 4	3D-DS-1	3D-DS-2	3D-DS-3	3D-DS-4
$errL^1$	7.52×10^{-3}	6.48×10^{-3}	1.16×10^{-2}	3.58×10^{-3}
$errL^2$	4.19×10^{-5}	3.08×10^{-5}	9.77×10^{-5}	9.11×10^{-6}
$J(\bar{U})$	1.02×10^{-2}	6.34×10^{-4}	1.63×10^{-3}	4.51×10^{-4}
ARCH 5	3D-DS-1	3D-DS-2	3D-DS-3	3D-DS-4
$errL^1$	1.96×10^{-3}	2.67×10^{-3}	8.78×10^{-3}	—
$errL^2$	3.00×10^{-6}	4.53×10^{-6}	5.84×10^{-5}	—
$J(\bar{U})$	5.15×10^{-4}	2.08×10^{-4}	6.90×10^{-4}	—
ARCH 6	3D-DS-1	3D-DS-2	3D-DS-3	3D-DS-4
$errL^1$	1.66×10^{-3}	1.87×10^{-3}	1.28×10^{-3}	2.11×10^{-3}
$errL^2$	2.37×10^{-6}	2.63×10^{-6}	1.60×10^{-6}	3.31×10^{-6}
$J(\bar{U})$	1.21×10^{-4}	1.20×10^{-4}	7.37×10^{-5}	1.16×10^{-4}

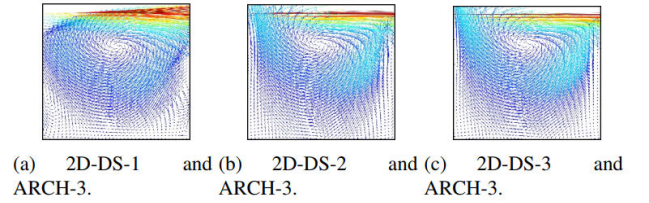


Figure 12: The 2D driven cavity flow.

- [5] I. E. Lagaris, A. Likas, Artificial neural networks for solving ordinary and partial differential equations. IEEE Transactions on Neural Networks. 1998, 9(5): 987-1000.
- [6] L. P. Aarts, P. Veer, Neural network method for solving partial differential equations. Neural processing letters. 2001, 14(3): 261-271.
- [7] S. Mall, S. Chakraverty, Single Layer Chebyshev Neural Network Model for Solving Elliptic Partial Differential Equations[J]. Neural processing letters. 2017, 45: 825. <https://doi.org/10.1007/s11063-016-9551-9>.
- [8] C. Ma, J. Wang, W. E, Model Reduction with Memory and the Machine Learning of Dynamical Systems. arXiv preprint arXiv:1808.04258.2018.
- [9] J. Berg, N. Kaj, A unified deep artificial neural network approach to partial differential equations in complex geometries[J]. Neurocomputing. 2017, 317: 28-41.
- [10] K. Sharmila, T. Rohit, B. Ilias, P. Jitesh, Simulator-free solution of high-dimensional stochastic elliptic partial differential equations using deep neural networks[J]. Journal of Computational Physics. 2020(404): 109120.
- [11] H. Sun, M. Hou, Y. Yang et al, Solving Partial Differential Equation Based on Bernstein Neural Network and Extreme Learning Machine Algorithm. Neural processing letters. 2019, 50: 1153-1172. doi:10.1007/s11063-018-9911-8.
- [12] M. Raissi, Forward-Backward Stochastic Neural Networks: Deep Learning of High-dimensional Partial Differential Equations[J]. arXiv preprint arXiv: 1804.07010.2018.
- [13] M. Raissi, P. Perdikaris, and G. E. Karniadakis, Physics informed deep learning (Part I): Data-driven solutions of nonlinear partial differential equations[J]. 2017, arXiv:1711.10561. [Online]. Available: <https://arxiv.org/abs/1711.10561>
- [14] M. Raissi, P. Perdikaris, and G. E. Karniadakis, Physics informed

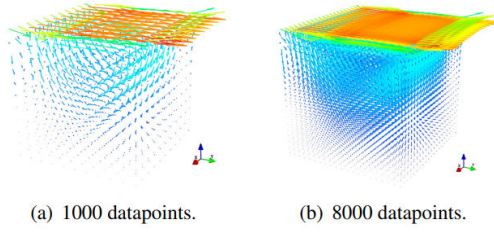


Figure 13: The 3D driven cavity flow testing on two different sampled data points.

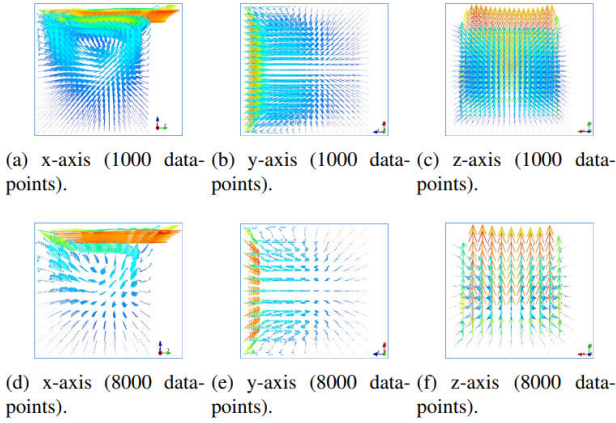


Figure 14: The top view along 3 axis of the 3D driven cavity flow.

deep learning (Part II): Data-driven discovery of nonlinear partial differential equations[J]. 2017, arXiv:1711.10566. [Online]. Available: <https://arxiv.org/abs/1711.10566>

- [15] M. Raissi, P. Perdikaris, and G. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations[J]. *Journal of Computational Physics*, 2019(378): 686-707.
- [16] L. Yang, X. Meng, G. Karniadakis, B-PINNs: Bayesian Physics-Informed Neural Networks for Forward and Inverse PDE Problems with Noisy Data[J]. arXiv: 2003.06097v1. 2020.
- [17] C. Rao, H. Sun, Y. Liu, Physics informed deep learning for computational elastodynamics without labeled data[J]. arXiv: 2006.08472v1. 2020.
- [18] P. Olivier, R. Fablet, PDE-NetGen 1.0: from symbolic PDE representations of physical processes to trainable neural network representations. 2020. <https://doi.org/10.5194/gmd-13-3373-2020>.
- [19] L. Lu, X. Meng, Z. Mao and G. Karniadakis, DEEPXDE: A Deep Learning Library for solving differential equations[J]. arXiv:1907.04502v2. 2020.
- [20] Z. Fang and J. Zhan, A Physics-Informed Neural Network Framework for PDEs on 3D Surfaces: Time Independent Problems[J]. *IEEE Access*. 2020(8): 26328-26335.
- [21] G. Pang, L. Lu and G. Karniadakis, fPINNs: Fractional Physics-Informed Neural Networks[J]. *SIAM Journal on Scientific Computing*. 41(4): A2603-CA2626.
- [22] J. Sirignano, K. Spiliopoulos, Stochastic Gradient Descent in Continuous Time[J]. *Social Science Electronic Publishing*. arXiv preprint arXiv:1611.05545.2017.
- [23] J. Sirignano, K. Spiliopoulos, DGM: A deep learning algorithm for solving partial differential equations[J]. *Journal of Computational Physics*. 2018(375): 1339-1364.
- [24] Y. Zhu and N. Zabaras, Bayesian deep convolutional encoder decoder networks for surrogate modeling and uncertainty quantification[J]. *Journal of Computational Physics*. 2018(366): 415-447.
- [25] Y. Zhu, N. Zabaras, P.-S. Koutsourelakis, and P. Perdikaris, Physics constrained deep learning for high-dimensional surrogate modeling and un-

certainty quantification without labeled data[J]. *Journal of Computational Physics*. 2019(394): 56-81.

- [26] H. Xu, D. Zhang and J. Zeng, Deep-learning of Parametric Partial Differential Equations from Sparse and Noisy Data[J]. *physics.comp-ph*. arXiv preprint arXiv: 2005.07916. 2020.
- [27] H. Xu, H. Chang and D. Zhang, DLGA-PDE: Discovery of PDEs with incomplete candidate library via combination of deep learning and genetic algorithm[J]. *Journal of Computational Physics*. 2020(418): 109584.
- [28] K. Hornik, Approximation capabilities of multilayer feedforward networks[J], *Neural Networks*. 1991(4): 251-257.
- [29] O. A. Ladyzenskaja, V. A. Solonnikov and N. N. Uralceva, *Linear and Quasi-linear Equations of Parabolic Type* (Translations of Mathematical Monographs Reprint)[M]. American Mathematical Society. 1988(23).
- [30] D. Gilbarg, N.S. Trudinger, *Elliptic partial differential equations of second order*, second edition[M]. Springer-Verlang, Berlin Heidelberg, 1983.
- [31] L. Boccardo, A. Dall'Aglio, T. Gallouët and L. Orsina, Nonlinear parabolic equations with measure data[J], *Journal of Functional Analysis*, 1997, 147: 237-258.
- [32] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al., Tensorflow: A system for large-scale machine learning, in: *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 2016, 265-283.