Journal of Scientific Computing (2022) 93:5
https://doi.org/10.1007/s10915-022-01930-8

The Deep Learning Galerkin Method for the General Stokes Equations

Jian Li[1] · Jing Yue[2] · Wen Zhang[2] · Wansuo Duan[3]

Received: 14 January 2021 / Revised: 2 March 2022 / Accepted: 16 April 2022

Abstract
This paper considers the general stokes problems applying the Deep learning Galerkin Method (DGM) and gives the convergence of the DGM which contains two parts. First, guided by data and physical laws, depending on the L^2 error we construct an objective function and control the performance of the approximation solution by minimizing the objective function in which the prior knowledge of PDEs and data are encoded. Then, we prove the convergence of the neural network to the exact solution. In particular, due to it is mesh free, the DGM can reduce the computational complexity and achieve the competitive results especially in face of the high dimensional problems. With this, compared with traditional numerical methods, numerical results verify the theoretical analysis and show the applicability and effectiveness of the proposed method.

Keywords General Stokes equations · Deep Galerkin Method · Convergence · Neural network · Deep learning

Mathematics Subject Classification 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*. Consequently, numerical methods such as finite element method, finite difference method and finite volume method have been flourishing in the past decades for modeling mechanics problems via solving PDEs [1–4]. Alternatively, the other methods, just

✉ Jian Li
jianli@sust.edu.cn

[1] School of Mathematics and Data Science, Shaanxi University of Science and Technology, Xi'an, China

[2] School of Electrical and Control Engineering, Shaanxi University of Science and Technology, Xi'an, China

[3] Institute of Atmospheric Physics, Chinese Academy of Sciences, Beijing, China

like generalized finite element basis functions [5] and construction of multiple difference schemes [6] have broad applications in the same way. Although these methods are well used in PDEs and have achieved good results, the generation of grid often has a certain impact on the computational efficiency especially for high dimensional problems.

There has mathematical guarantee called universal approximation theorem [7] which stating that a single layer neural network could approximate many functions in Sobolev spaces. Neural networks have become an alternative to the numerical solution of differential equations, because they avoid some of the drawbacks of traditional numerical techniques. It has been considered in various forms previously since the 1990s and gains a lot of interests for efficiently solving differential equations [8]. For example, domain discretization usually involves a simple square domain, and does not require special processing for nonlinear differential equations. Such as Cellular Neural Network and Distributed Parameter Neural Network are used for one-dimensional PDEs [9, 10], single layer Chebyshev neural network [11], recurrent neural network and ansatz method [12, 13] can also solve the PDEs. Sun *et al.* [14, 15] used Bernstein neural network and extreme learning machine to solve low order ordinary differential equations and elliptic PDEs.

Due to the limit of the traditional methods, researchers gradually consider the deep learning methods for solving higher-dimensional problems. Inspired by machine learning, the deep learning methods learn the parameters of neural network by minimizing an objective function. These methods combine the random sampled data and the prior knowledge of PDEs and thus can avoid mesh generation to some certain extent. With this, deep learning method has certain adaptability for unknown data, which can guarantee the high accuracy through training the models [16–30]. Despite such remarkable works, the theoretical research still needs to be promoted. Latterly, quiet a few researchers enriched mathematical analysis on the excellent performance of deep neural network in PDEs [31–35].

In this paper, the DGM is applied to solve the general d-dimentional 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 computes variables in parallel and 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 to the exact solution.

The remaining of this paper is organized as follows: In Sect. 2, we provide the preliminaries of methodology of the PDEs. In Sect. 3, we prove the convergence of the objective function and the convergence of the neural network to the exact solution. Section 4 gives numerical examples to demonstrate the efficiency of the proposed framework and justify our theoretical analysis. Finally, Sect. 5 presents a conclusion.

2 Methodology

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

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

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

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

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 $\overline{u} = (u, p)$ and define

$$\mathcal{G}[\overline{u}] = \alpha u - \nu\nabla^2 u + \nabla p - f. \tag{4}$$

Here, we recall the classical Sobolev spaces

$$H^k(\Omega) = \left\{\upsilon \in L^2(\Omega) : D_w^\alpha \upsilon \in L^2(\Omega), \forall\alpha, \mid \alpha \mid \leq k\right\},$$

$$H_0^k(\Omega) = \left\{\upsilon \in H^k(\Omega) : \upsilon \mid_{\partial\Omega} = 0\right\},$$

$$L_0^2(\Omega) = \{q \in L^2(\Omega) : \int_\Omega q dx = 0\},$$

and their norm

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

$$\|q\|_0 = \left(\int_\Omega q^2 dx\right)^{\frac{1}{2}},$$

where $k > 0$ is a positive integer and $\|q\|_0$ denotes the norm on $L_0^2(\Omega)$ or $(L^2(\Omega))^i, i = 1, 2, 3$, $D_w^\alpha \upsilon$ is the generalized derivative of υ, and $(\cdot, \cdot)$ represents the inner product.

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

Lemma 2.1 [38] *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*

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

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

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

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

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

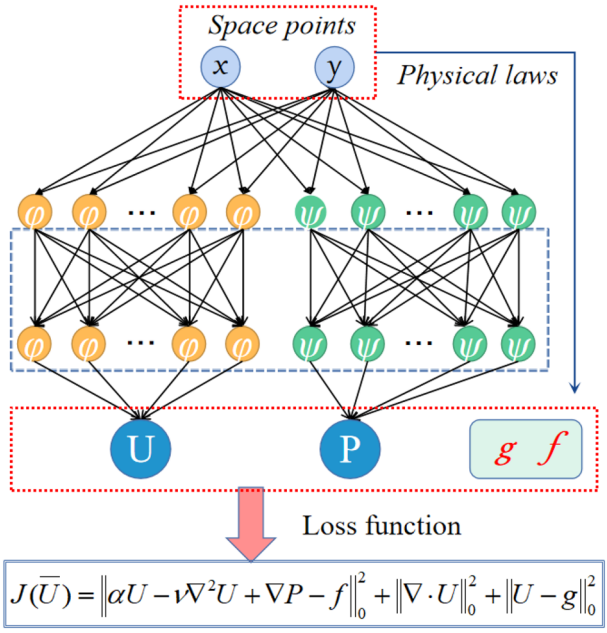


Fig. 1 The neural network architecture of the general Stokes equations

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

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

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

Our goal is to find the parameters θ such that $\overline{U}$ minimizes the objective function $J(\overline{U})$. Especially, if $J(\overline{U}) = 0$ then $\overline{U}$ is the solution to the general Stokes Eqs. (1)-(3). However, it is computationally infeasible to estimate θ by directly minimizing $J(\overline{U})$ 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 main idea of the DGM for the general Stokes equations are displayed in Fig. 1.

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

$$\mathbb{E}\Big[\nabla_\theta G(\theta_n, s_n) \mid \theta_n\Big] = \nabla_\theta J\left(\overline{U}(\cdot\,; \theta_n)\right). \tag{7}$$

In order to illustrate more vividly, the pseudocode is shown in Algorithm 1.

3 Convergence Analysis

Undoubtedly, the objective function $J(\overline{U})$ can measure how well the neural network $\overline{U}$ satisfies the differential operator, boundary condition and divergence condition. We know that the total errors of neural networks-based supervised learning can be decomposed into

Algorithm 1: Deep learning Galerkin Method

Input: Randomly sample points $s_n = (x_n, r_n)$, Max Iterations M, learning rate α_n.
Output: θ_{n+1}.
Initialize the parameters θ;
while *iterations* $\leq M$ **do**
read current;

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

and

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

if $\lim_{n\to\infty} \|\nabla_\theta G(\theta_n, s_n)\| = 0$ **then**
return the parameters θ_{n+1};
else
go back to the beginning of current section;
end
end

three components: (a) approximation error, (b) optimization errors, and (c) estimation error. In this paper, the convergence analysis we discussed is generalization error, which contains approximation error and estimation error.

In more detail, the "convergence" means that we can use the multilayer feed forward networks U to universally approximate the exact solution to the general Stokes equations. Therefore, the neural network U can make the objective function $J(U)$ arbitrarily small. As we known from [7], if there is only one hidden layer and one output, then the set of all functions implemented by such a network with m and n hidden units for velocity and pressure are

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

and

$$\mathfrak{C}_p^{\mathrm{n}}(\psi) = \Big\{\Psi(x) : \mathbb{R}^{\mathrm{d}} \mapsto \mathbb{R} \Big| \Psi(x) = \sum_{i=1}^{n} \beta_i' \psi\big(\sum_{j=1}^{d} \sigma_{ji}' x_j + c_i'\big)\Big\},$$

where $\ell = 1, 2, \ldots, d$, $\Phi(x) = \big(\Phi_1(x), \Phi_2(x), \cdots, \Phi_d(x)\big)$, φ and ψ are the shared activation functions of the hidden units in $\mathcal{C}^2(\Omega)$, 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 such as $[\mathfrak{C}_u(\varphi)]^d \times \mathfrak{C}_p(\psi) = \bigcup_{m\geq 1}[\mathfrak{C}_u^m(\varphi)]^d \times \bigcup_{n\geq 1} \mathfrak{C}_p^n(\psi)$. The parameters can be written as follows

$$\begin{aligned} \theta_1^\ell &= (\beta_1^\ell, \cdots, \beta_m^\ell, \sigma_{11}^\ell, \cdots, \sigma_{dm}^\ell, c_1^\ell, \cdots, c_m^\ell), \\ \theta_2 &= (\beta_1', \cdots, \beta_n', \sigma_{11}', \cdots, \sigma_{dn}', c_1', \cdots, c_n'), \end{aligned} \tag{8}$$

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

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

3.1 Convergence of the Objective Function $J(\overline{U})$

A particularly important processing, we use the multilayer feed forward networks $\overline{U}$ to universally approximate the exact solution to the general Stokes equations. Certainly, the neural network $\overline{U}$ can make the objective function $J(\overline{U})$ arbitrarily small. Thus, using the results of [7] and the following lemma, we obtain the convergence of the objective function $J(\overline{U})$. First, we give the following assumption.

Lemma 3.1 *Assume that $\nabla u(x)$, $\triangle u(x)$ 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 $0 \le q_i \le \infty (i = 1, 2, 3, 4)$ we have*

$$| \triangle U - \triangle u | \le (| \nabla U |^{q_1/2} + | \nabla u |^{q_2/2}) | \nabla U - \nabla u |, \tag{9}$$

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

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

$$J(\overline{U}) \le C\epsilon, \ \forall \epsilon > 0, \tag{11}$$

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

Proof By Theorem 3 of [7], we can conclude that there exists $\overline{U} \in [\mathfrak{C}_u(\varphi)]^d \times \mathfrak{C}_p(\psi)$ which is uniformly 2-dense on compacts of $\mathcal{C}^2(\bar{\Omega}) \times \mathcal{C}^1(\bar{\Omega})$. It means that for $\overline{u} \in \mathcal{C}^2(\bar{\Omega}) \times \mathcal{C}^1(\bar{\Omega})$, $\forall \epsilon > 0$, it follows that

$$\underset{a \le 2}{max} \underset{x \in \Omega}{sup} | \partial_x^a U(x) - \partial_x^a u(x) | < \epsilon, \tag{12}$$

$$\underset{x \in \Omega}{sup} | P(x) - p(x) | < \epsilon. \tag{13}$$

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

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

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

Similarly,

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

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

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

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

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

which implies (11) after rescaling ϵ. □

3.2 Convergence of the Neural Network to the General Stokes Solution

We have discussed the convergence of the objective function $J(\overline{U})$ in the last subsection. Next we give the convergence of the neural network $\overline{U}^n$ to the exact solution $\overline{u}$ 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

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

By Theorem 3.1, we obtain

$$J(\overline{U}^n) \to 0 \ \text{ as } \ n\to\infty.$$

Furthermore, according to the thought of the Galerkin method, each neural network $\overline{U}^n = (U^n, P^n)$ satisfies the following equations

$$\mathcal{G}[\overline{U}^n] = 0, \quad \text{in } \Omega, \tag{21}$$

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

$$U^n = 0, \quad \text{on } \partial\Omega. \tag{23}$$

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 Sect. 4 of Chapter V in [36] or Chapter 8 of [37] for details). For convenience, we provide a theorem to guarantee the convergence of the neural network $\overline{U}^n$ and the exact solution $\overline{u}$ to the Eqs. (18)-(20).

Theorem 3.2 *Under the assumptions of Lemma 3.1, Theorem 3.1, 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 uniformly converge to u and p respectively in Ω.*

Proof The existence and uniqueness for the solution of (18)-(20) are proved by the Saddle point theorem (See Lemma 2.1). We briefly consider the boundness of $(U^n, P^n) \in$

$[\mathfrak{C}_u^n(\varphi)]^d \bigcap [H_0^1(\Omega)]^d \times \mathfrak{C}_p^n(\psi) \bigcap L_0^2(\Omega)$ for the Eqs. (21)-(23). Multiplying the above equations by $(v, q) \in [\mathfrak{C}_u^n(\varphi)]^d \bigcap [H_0^1(\Omega)]^d \times \mathfrak{C}_p^n(\psi) \bigcap L_0^2(\Omega)$, we obtain the variational formulation as follows:

$$\alpha(U^n, v) + \nu(\nabla U^n, \nabla v) - (\nabla P^n, v) + (\nabla \cdot U^n, q) = (f, v). \tag{24}$$

Noting that

$$(\nabla P^n, v) = -(\nabla \cdot v, P^n), \tag{25}$$

taking $(v, q) = (U^n, P^n)$ in (24), and using the definition of the H^1-norm, it follows that

$$\|U^n\|_{1,\Omega} \leq C\|f\|_{-1,\Omega}. \tag{26}$$

By using the uniformly boundedness of U^n, we can extract a subsequence $\{U^n\}_{n\in\mathbb{N}}$ of U^n which can converge weakly in $H^1(\Omega)$. Due to the compact embedding $H^1(\Omega) \hookrightarrow L^2(\Omega)$, we have $\lim\limits_{n\to\infty} \| U^n - u \|_{0,\Omega} = 0$.

In order to study the pressure of the general Stokes problem, let $v \in [\mathfrak{C}_u^n(\varphi)]^d \cap [H_0^1(\Omega)]^d$, we define

$$L(v) =: (f, v) - \alpha(U^n, v) - \nu(\nabla U^n, \nabla v) = 0. \tag{27}$$

Then

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

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

In addition, there exists $P^n \in \mathfrak{C}_p^n(\psi) \cap L_0^2(\Omega)$, for $\forall v \in [\mathfrak{C}_u^n(\varphi)]^d \cap [H_0^1(\Omega)]^d$ such that

$$< L, v >= \int_\Omega P^n \mathrm{div} v dx = -(P^n, \mathrm{div} v).$$

Namely,

$$(P^n, \mathrm{div} v) = \alpha(U^n, v) + \nu(\nabla U^n, \nabla v) - (f, v). \tag{28}$$

What's more, as in Theorem 3.3 of [40], we find that

$$\nabla U^n \to \nabla u \ \ almost \ \ everywhere \ \ in \ \ \Omega,$$

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

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

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 Arzel$\grave{a}$-Ascoli theorem. □

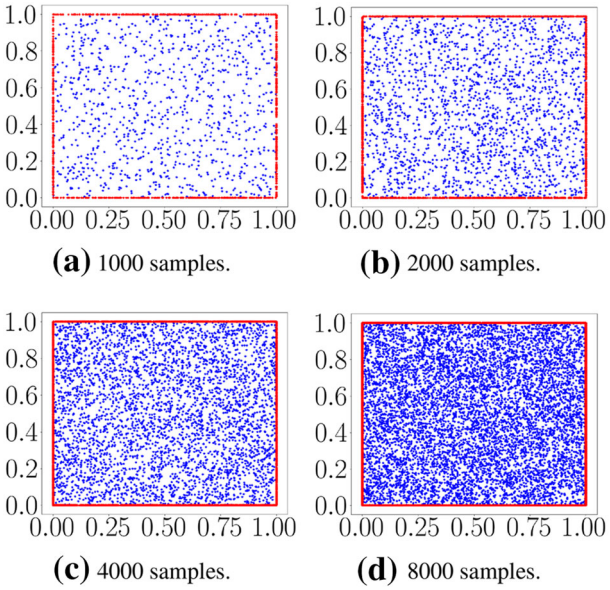


Fig. 2 The datasets in 2D case

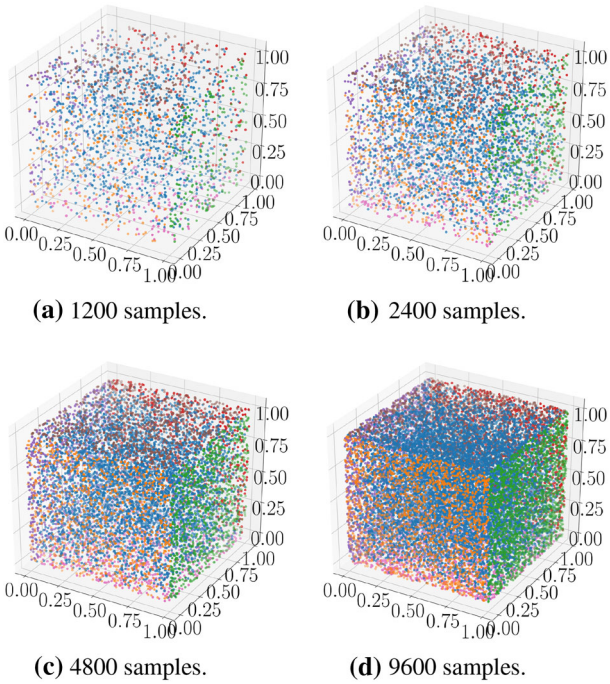
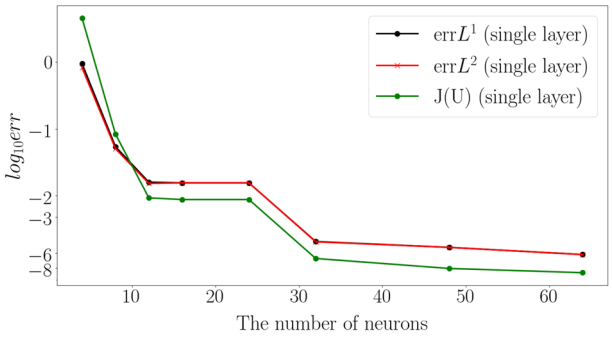
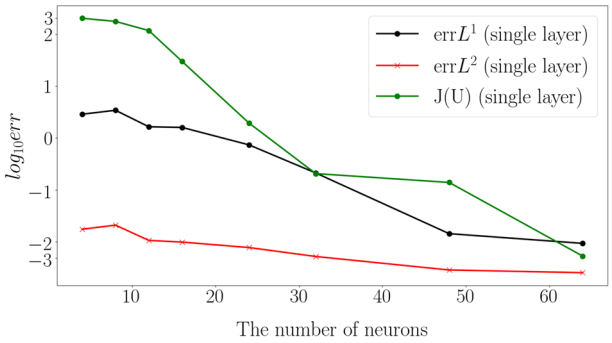


Fig. 3 The datasets in 3D case



(a) 2D case.



(b) 3D case.

Fig. 4 The relative error of different neurons

4 Numerical Experiments

The general Stokes problems are considered in this section especially with the high dimentional problems. Our numerical experiments are based on Tensorflow [41] and the configuration of the computer is 64-bit Intel Xeon Silver 4116 (2 processors). In order to demonstrate the effectiveness and accuracy of the DGM, we compute the L^1, L^2 relative errors and $J(\overline{U})$ as follows

$$errL^1 = \frac{\| U - u \|_{L^1}}{\| u \|_{L^1}}, \tag{29}$$

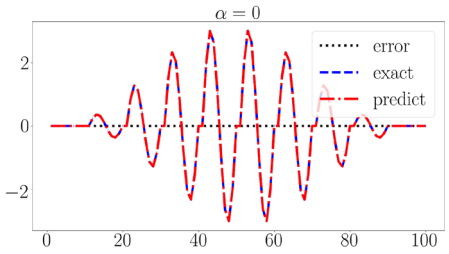
$$errL^2 = \frac{\| U - u \|_{L^2}}{\| u \|_{L^2}}, \tag{30}$$

$$J(\overline{U}) = \frac{1}{N}\sum_{i=1}^{N}\left[\left\|\mathcal{G}[\overline{U_i}]\right\|_0^2 + \left\|\nabla \cdot U_i\right\|_0^2 + \left\|U_i - g_i\right\|_0^2\right], \tag{31}$$

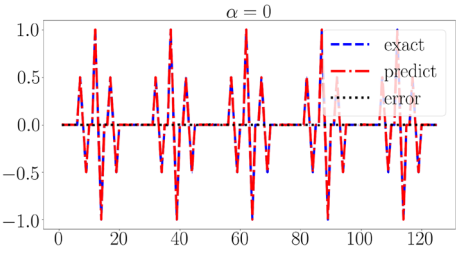
where $\overline{U}_i$ and u_i are the neural network and exact solution on each point $i = 1, 2, \cdots N$ of datasets, respectively. Here, we only calculate the error of velocity and the pressure is similar. For conveniently, we set the same number of network layers to solve U and P simultaneously. The datasets in 2D case contain 1000, 2000, 4000 and 8000 samples respectively (See Fig. 2.

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

ARCH a1	1000	2000	4000	8000
$errL^1$	9.87×10^{-3}	1.08×10^{-2}	1.18×10^{-2}	1.14×10^{-2}
$errL^2$	9.48×10^{-3}	1.05×10^{-2}	1.13×10^{-2}	1.09×10^{-2}
$J(\overline{U})$	1.52×10^{-2}	2.05×10^{-3}	1.75×10^{-3}	1.79×10^{-3}
ARCHa2	1000	2000	3000	4000
$errL^1$	3.79×10^{-5}	4.81×10^{-5}	2.95×10^{-5}	4.48×10^{-5}
$errL^2$	3.99×10^{-5}	5.11×10^{-5}	3.15×10^{-5}	4.75×10^{-5}
$J(\overline{U})$	9.60×10^{-8}	2.71×10^{-7}	1.02×10^{-7}	5.14×10^{-7}
ARCH a3	1000	2000	3000	4000
$errL^1$	1.65×10^{-5}	9.02×10^{-6}	7.71×10^{-6}	8.59×10^{-6}
$errL^2$	1.75×10^{-5}	9.83×10^{-6}	8.49×10^{-6}	9.47×10^{-6}
$J(\overline{U})$	1.77×10^{-8}	4.36×10^{-8}	1.15×10^{-8}	2.59×10^{-9}



(a) 2D: ARCH a3 and 1000 training data.



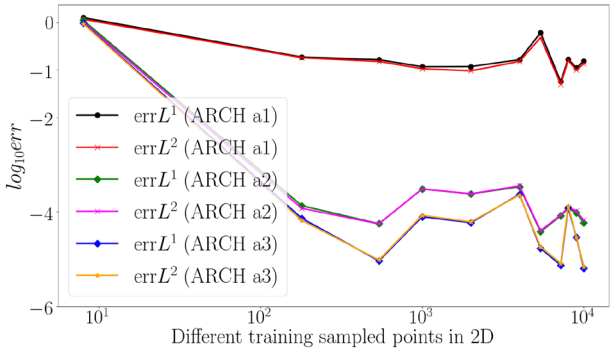
(b) 3D: ARCH b3 and 1200 training data.

Fig. 5 The absolute error with points in 2D and 3D case

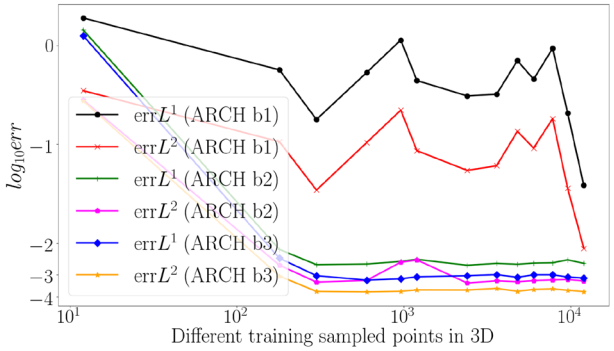
And in 3D case, the datasets contain 1200, 2400, 4800 and 9600 samples respectively (See Fig. 3).

4.1 The Stokes Equations

In this subsection, we consider the Stokes equations with homogeneous boundary condition both in 2D and 3D cases. Set $\nu = 1$ and $\nu = 0.025$ for 2D and 3D cases respectively. Use



(a) 2D: Different training data and ARCH a1-a3.



(b) 3D: Different training data and ARCH b1-b3.

Fig. 6 The relative errors of different training data both in 2D and 3D case ($\alpha = 1$)

the following exact solutions,

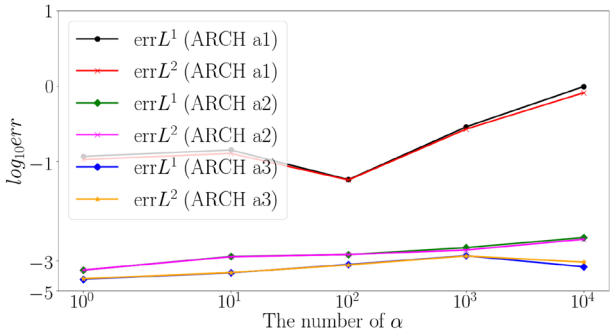
$$\begin{aligned} u_1(x_1, x_2) &= 2sin(\pi x_1)^2 sin(\pi x_2) cos(\pi x_2)\pi, \\ u_2(x_1, x_2) &= -2sin(\pi x_1) sin(\pi x_2)^2 cos(\pi x_1)\pi, \\ p(x_1, x_2) &= cos(\pi x_1) cos(\pi x_2), \end{aligned} \tag{32}$$

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

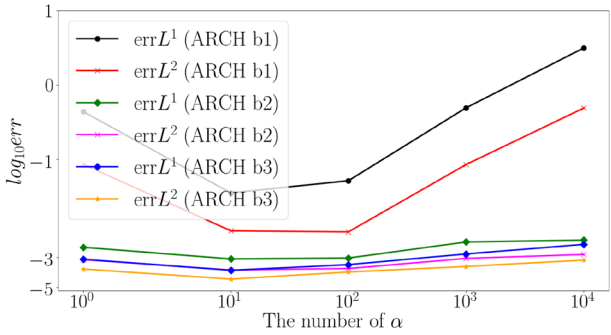
$$\begin{aligned} u_1(x, y, z) &= sin(\pi x)^2 (sin(2\pi y) sin(\pi z)^2 - sin(\pi y)^2 sin(2\pi z)), \\ u_2(x, y, z) &= sin(\pi y)^2 (sin(2\pi z) sin(\pi x)^2 - sin(\pi z)^2 sin(2\pi x)), \\ u_3(x, y, z) &= sin(\pi z)^2 (sin(2\pi x) sin(\pi y)^2 - sin(\pi x)^2 sin(2\pi y)), \\ p(x, y, z) &= sin(\pi x) sin(\pi y) cos(\pi z), \end{aligned} \tag{33}$$

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

For the stokes equations, the results of relative errors and $J(U)$ are plotted in Fig. 4. It can be observed that the results produced by the DGM converge as the number of neurons increases. Obviously, from Fig. 4(b), for 3D problems, fewer neurons is not enough to achieve



(a) 2D: Different α and ARCH a1-a3.



(b) 3D: Different α and ARCH b1-b3.

Fig. 7 The relative errors of different α and ARCH both in 2D and 3D case ($\alpha = 1$)

the required precision. Therefore, it is indispensable to adopt more neurons since the non-wide and non-deep neural network has great limitation for the expression of nonlinear relationship. Here, we choose 16 units each layer for ARCH a1-a3(1-3 hidden layers) in 2D case, 32 units each layer for ARCH b1-b3(1-3 hidden layers) in 3D case.

Table 1 displays the value of relative error and $J(\overline{U})$ in 2D case, the best results for each ARCH are marked out. From Table 1 we can find that numerical results with different datasets are less distinguishable. Additionally, Fig. 5 shows that the absolute error with points both in 2D and 3D case. From the above results, we can find that with the decrease of $J(\overline{U})$, the relative errors between the exact solution and the approximate solution tends to converge, which indicate that the DGM can solve the stoke problem accurately and effectively.

4.2 The General Stokes Equations

For the general stokes problems, [42] proposed the problem of topology optimization of fluids in Stokes flow and [43] applied the physics informed neural network for inverse design. Consider the focus of this article is not the inverse design, we utilize a forward thought to solve the general stokes problems with different α and verify the performance of the DGM. Similarly, we utilize same ARCH, ν and the analytical solutions for 2D and 3D cases as before. Consequently, the right hands $f(x, y)$ and $f(x, y, z)$ can be derived by Eq. (1).

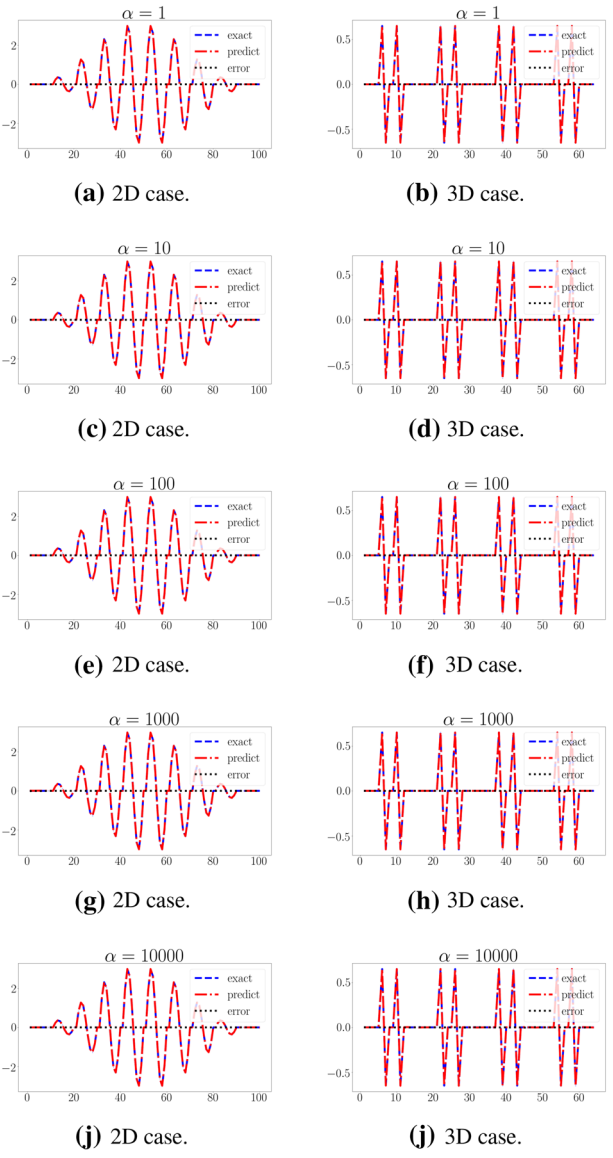
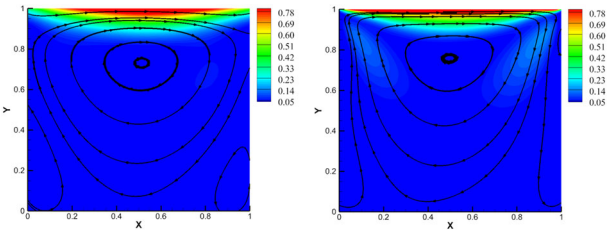
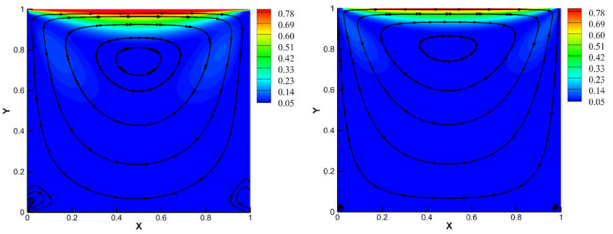


Fig. 8 The absolute error with points in 2D and 3D case for different α

Here, we firstly consider whether the precision of the neural network is related to the number of hidden layers and the size of the datasets. Figure 6 shows that the accuracy of the neural networks tends to be small and stable only as the hidden layer increases. A particularly significant consideration, the relative error drops rapidly once the training data more than 100 whether in 2D or 3D case. Figures 7 and 8 display that the relative and absolute errors with different α and ARCH both in 2D and 3D case. Seen from Fig. 7, the approximation solution to the general Stokes problem can still maintain a stable effect with the increase of α especially in higher hidden layers.

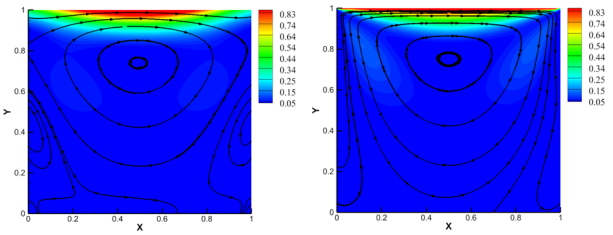


(a) ARCH a1. **(b)** ARCH a2.

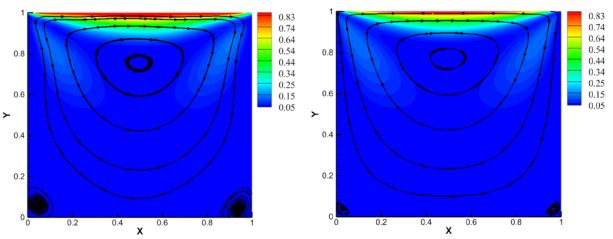


(c) ARCH a3 **(d)** FreeFem.

Fig. 9 The contrast of the 2D driven cavity flow ($\nu = 0.025$)



(a) ARCH a1. **(b)** ARCH a2.

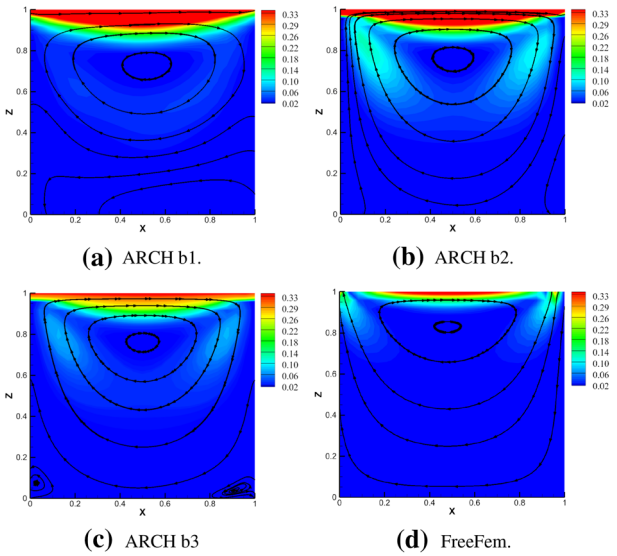


(c) ARCH a3 **(d)** FreeFem.

Fig. 10 The contrast of the 2D driven cavity flow ($\nu = 0.5$)

4.3 The Driven Cavity Flow

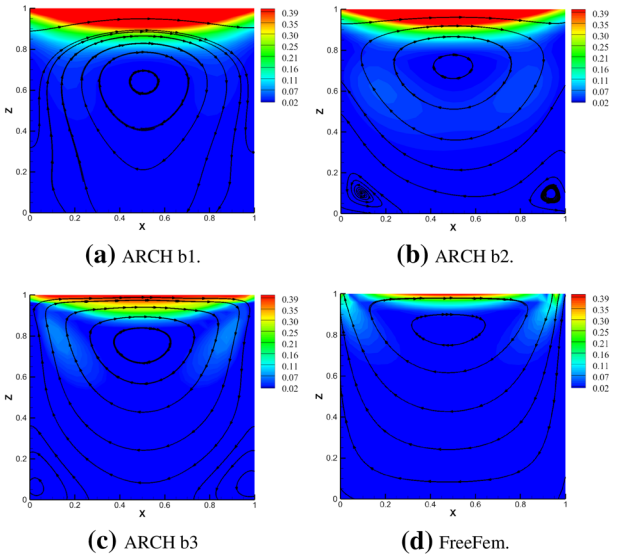
The driven cavity flows have been extensively applied as test cases for validating the incompressible fluid dynamics algorithm. The corner singularities for the 2D fluid flows are very important since most examples of physical interest have corners. In these two examples, we consider the 2D driven flow 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



(a) ARCH b1. **(b)** ARCH b2.

(c) ARCH b3 **(d)** FreeFem.

Fig. 11 The contrast of the 3D driven cavity flow ($y = 0.5$ and $\nu = 0.025$)



(a) ARCH b1. **(b)** ARCH b2.

(c) ARCH b3 **(d)** FreeFem.

Fig. 12 The contrast of the 3D driven cavity flow ($y = 0.5$ and $\nu = 0.5$)

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 $x = y = z = 0.5$. A unit tangential velocity in the x direction is prescribed 1 at the top surface, while zero velocity is prescribed on the remaining bounding surfaces in numerical examples. Here, we set $\nu = 0.025$ and $\nu = 0.5$ for 2D and 3D cases.

Table 2 The relative errors between the results of the DGM and FreeFem

2D ($\nu = 0.025$)	ARCH a1	ARCH a2	ARCH a3
$errL^1$	8.81×10^{-2}	4.28×10^{-2}	3.76×10^{-2}
$errL^2$	4.50×10^{-3}	7.90×10^{-3}	7.10×10^{-3}
2D ($\nu = 0.5$)	ARCH a1	ARCH a2	ARCH a3
$errL^1$	5.71×10^{-2}	7.00×10^{-3}	3.68×10^{-4}
$errL^2$	4.70×10^{-3}	4.60×10^{-3}	3.90×10^{-3}
3D ($\nu = 0.025$)	ARCH b1	ARCH b2	ARCHb3
$errL^1$	3.07×10^{-1}	2.02×10^{-1}	1.81×10^{-1}
$errL^2$	6.90×10^{-3}	5.40×10^{-3}	5.30×10^{-3}
3D ($\nu = 0.5$)	ARCH b1	ARCH b2	ARCH b3
$errL^1$	2.68×10^{-1}	2.02×10^{-1}	1.75×10^{-1}
$errL^2$	1.00×10^{-2}	7.50×10^{-3}	6.50×10^{-3}

We train the DGM in 2D case (1000 training data points, $\nu = 0.025, 0.5$ and ARCH a1-a3) and compare the results with the FreeFem in Figures 9, 10. Similarly, by using 1200 training data points and ARCH b3, the results in 3D case are depicted in Figures 11, 12. Table 2 provides relative errors to reflect the closeness between the results of the DGM and the FreeFem. The above numerical results indicate that the DGM has competitiveness to model physical phenomena when ν changed.

5 Conclusions

This paper applies the DGM to solve the general Stokes problems in 2D and 3D. This method 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. The numerical results fully demonstrate the convergence properties of the DGM completely. But, we need more deliberations on the elements which have great impact in experiments. 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 has a positive effect on the algorithm.

Funding This work is supported in part by NSF of China (No. 11771259), special support program to develop innovative talents in the region of Shaanxi province, innovation team on computationally efficient numerical methods based on new energy problems in Shaanxi province, and innovative team project of Shaanxi Provincial Department of Education (No. 21JP013).

Data Availability Enquiries about data availability should be directed to the authors.

Declarations

Competing interests The authors declare that there is no conflict of interest.

References

1. Wang, Q., Cheng, D.: Numerical solution of damped nonlinear Klein-Gordon equations using variational method and finite element approach[J]. Appl. Math. Comput. **162**(1), 381–401 (2005)
2. Li, J.: Numerical method of Navier-Stokes equations for the incompressible flows[M]. Science Press, Beijing (2019). (in Chinese)
3. Li, J., Lin, X., Chen, Z.: Finite VolumeMethods for the Incompressible Navier-Stokes Equations[M], Springer Verlag, Berlin, Heidelberg. (2022). https://doi.org/10.1007/978-3-030-94636-4
4. Li, J., Bai, Y., Zhao, X.: Modern numerical methods for mathematical physics equations[M]. Science Press, Beijing. (in Chinese). Acceptted
5. Pels, A., Sabariego, R.V., Schops, S.: Solving multirate partial differential equations using hat finite element basis functions[C]. IEEE Conference on Electromagnetic Field Computation (2016). https://doi.org/10.1109/2016.7816348
6. Zhao, G., Jie, K., Liu, J.: A New Difference Scheme for Hyperbolic Partial Differential Equations[C] International Conference on Computational Intelligence and Security. IEEE, (2018). https://doi.org/10.1109/CIS.2017.00102
7. Hornik, K.: Approximation capabilities of multilayer feedforward networks[J]. Neural Netw. **4**, 251–257 (1991)
8. Lagaris, I.E., Likas, A.: Artificial neural networks for solving ordinary and partial differential equations[J]. IEEE Trans. Neural Networks **9**(5), 987–1000 (1998)
9. Dazheng, F., Zheng, B., Licheng, J.: Distributed parameter neural networks for solving partial differential equations[J]. J. Electron. **14**(2), 186–190 (1997)
10. Aarts, L.P., Veer, P.: Neural network method for solving partial differential equations[J]. Neural Process. Lett. **14**(3), 261–271 (2001)
11. Mall, S., Chakraverty, S.: Single Layer Chebyshev Neural Network Model for Solving Elliptic Partial Differential Equations[J]. Neural Process. Lett. **45**, 825 (2017). https://doi.org/10.1007/s11063-016-9551-9
12. Ma, C., Wang, J., E, W.: Model Reduction with Memory and the Machine Learning of Dynamical Systems[J]. arXiv preprint arXiv:1808.04258
13. Berg, J., Kaj, N.: A unified deep artificial neural network approach to partial differential equations in complex geometries[J]. Neurocomputing. **317**, 28–41 (2017)
14. Sharmila, K., Rohit, T., Ilias, B., Jitesh, P.: Simulator-free solution of high-dimensional stochastic elliptic partial differential equations using deep neural networks[J]. J. Comput. Phys. **404**, 109120 (2020)
15. Sun, H., Hou, M., Yang, Y., et al.: Solving Partial Differential Equation Based on Bernstein Neural Network and Extreme Learning Machine Algorithm[J]. Lett. **50**, 1153–1172 (2019). https://doi.org/10.1007/s11063-018-9911-8
16. Raissi, M.: Forward-Backward Stochastic Neural Networks: Deep Learning of High-dimensional Partial Differential Equations[J]. arXiv preprint arXiv:1804.07010
17. Raissi, M., Perdikaris, P., Karniadakis, G.E.: Physics informed deep learning (Part I): Data-driven solutions of nonlinear partial differential equations[J]. 2017, arXiv:1711.10561
18. Raissi, M., Perdikaris, P., Karniadakis, G.E.: Physics informed deep learning (Part II): Data-driven discovery of nonlinear partial differential equations[J]. (2017), arXiv:1711.10566
19. Raissi, M., Perdikaris, P., Karniadakis, G.: Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations[J]. J. Comput. Phys. **378**, 686–707 (2019)
20. Yang, L., Meng, X., Karniadakis, G.: B-PINNs: Bayesian Physics-Informed Neural Networks for Forward and Inverse PDE Problems with Noisy Data[J]. arXiv:2003.06097v1. (2020)
21. Rao, C., Sun, H., Liu, Y.: Physics informed deep learning for computational elastodynamics without labeled data[J]. arXiv:2006.08472v1. (2020)
22. Olivier, P., Fablet, R.: PDE-NetGen 1.0: from symbolic PDE representations of physical processes to trainable neural network representations[J]. (2020). https://doi.org/10.5194/gmd-13-3373-2020
23. Lu, L., Meng, X., Mao, Z., Karniadakis, G.: DEEPXDE: A Deep Learning Library for solving differencial equations[J]. arXiv:1907.04502v2. (2020)

24. Fang, Z., Zhan, J.: A Physics-Informed Neural Network Framework for PDEs on 3D Surfaces: Time Independent Problems[J]. IEEE Access. **8**, 26328–26335 (2020)
25. Pang, G., Lu, L., Karniadakis, G.: fPINNs: Fractional Physics-Informed Neural Networks[J]. SIAM J. Sci. Comput. **41**(4), A2603-CA2626 (2019)
26. Raissi, M., Perdikaris, P., Karniadakis, G.E.: Multistep Neural Networks for Data-driven Discovery of Nonlinear Dynamical Systems[J]. arXiv:1801.01236v1 [math.DS] 4 Jan (2018)
27. Zhu, Y., Zabaras, N.: Bayesian deep convolutional encoder decoder networks for surrogate modeling and uncertainty quantification[J]. J. Comput. Phys. **366**, 415–447 (2018)
28. Zhu, Y., Zabaras, N., Koutsourelakis, P.-S., Perdikaris, P.: Physics constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data[J]. J. Comput. Phys. **394**, 56–81 (2019)
29. Xu, H., Zhang, D., Zeng, J.: Deep-learning of Parametric Partial Differential Equations from Sparse and Noisy Data[J]. physics.comp-ph. arXiv preprint arXiv:2005.07916. (2020)
30. Xu, H., Chang, H., Zhang, D.: DLGA-PDE: Discovery of PDEs with incomplete candidate library via combination of deep learning and genetic algorithm[J]. J. Comput. Phys. **418**, 109584 (2020)
31. Shin, Y., Darbon, J., Karniadakis, G.E.: On the Convergence of Physics Informed Neural Networks for Linear Second-Order Elliptic and Parabolic Type PDEs[J]. Commun. Comput. Phys. **28**(5), 2042–2074 (2020)
32. Sirignano, J., Spiliopoulos, K.: Stochastic Gradient Descent in Continuous Time[J]. Social Science Electronic Publishing. arXiv preprint arXiv:1611.05545
33. Sirignano, J., Spiliopoulos, K.: DGM: A deep learning algorithm for solving partial differential equations[J]. J. Comput. Phys. **375**, 1339–1364 (2018)
34. Yue, J., Li, J.: The Physics Informed Neural Networks for the unsteady Stokes problems[J]. Int. J. for Numerical Methods in Fluids (2022). https://onlinelibrary.wiley.com/doi/abs/10.1002/fld.5095
35. Li, J., Zhang, W., Yue, J.: A Deep Learnign Galerkin Method for the Second-order linear elliptic equations[J]. Int. J. Numer. Anal. Model. **18**(4), 427–441 (2021)
36. Ladyzenskaja, O.A., Solonnikov, V. A., Uralceva, N.N.: Linear and Quasi-linear Equations of Parabolic Type (Translations of Mathematical Monographs Reprint)[M]. American Mathematical Society. 1988(23)
37. Gilbarg, D., Trudinger, N.S.: Elliptic Partial Differential Equations of Second Order[M]. 2nd edn. Springer-Verlang, Berlin Heidelberg (1983)
38. Temam, R.: Navier-Stokes Equations, Theory and Numerical Analysis[M], 3rd edn. North-Holland, Amsterdam (1984)
39. Li, J., He, Y.: Superconvergence of discontinuous Galerkin finite element method for the stationary Navier-Stokes equations[J]. Numerical Methods for Partial Differential Equations **23**(2), 421–436 (2007)
40. Boccardo, L., Dall'Aglio, A., Gallouët, T., Orsina, L.: Nonlinear parabolic equations with measure data[J]. J. Funct. Anal. **147**, 237–258 (1997)
41. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al.: Tensorflow: A system for large-scale machine learning[C] in: 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), 265-283 (2016)
42. Borrvall, T., Petersson, J.: Topology optimization of fluids in Stokes flow[J]. Int. J. for Numerical Methods in Fluids **41**, 77–107 (2003)
43. Lu, L., Pestourie, R., Yao, W., Wang, Z., Verdugo, F., Johnson, S.G.: Physics-Informed Neural Networks with hard constraints for inverse design[J]. physics. comp-ph. arXiv:2102.04626v1