VPVnet: a velocity-pressure-vorticity neural network method for the Stokes’ equations under reduced regularity

Yujie Liu¹ and Chao Yang²,*

¹ Artificial Intelligence Research Center, Peng Cheng Laboratory, Shenzhen 518005, China.
² School of Mathematical Sciences, Peking University, Beijing 100871, China.

Abstract. We present VPVnet, a deep neural network method for the Stokes’ equations under reduced regularity. Different with recently proposed deep learning methods [40,51] which are based on the original form of PDEs, VPVnet uses the least square functional of the first-order velocity-pressure-vorticity (VPV) formulation (30) as loss functions. As such, only first-order derivative is required in the loss functions, hence the method is applicable to a much larger class of problems, e.g. problems with non-smooth solutions. Despite that several methods have been proposed recently to reduce the regularity requirement by transforming the original problem into a corresponding variational form, while for the Stokes’ equations, the choice of approximating spaces for the velocity and the pressure has to satisfy the LBB condition additionally. Here by making use of the VPV formulation, lower regularity requirement is achieved with no need for considering the LBB condition. Convergence and error estimates have been established for the proposed method. It is worth emphasizing that the VPVnet method is divergence-free and pressure-robust, while classical inf-sup stable mixed finite elements for the Stokes’ equations are not pressure-robust. Various numerical experiments including 2D and 3D lid-driven cavity test cases are conducted to demonstrate its efficiency and accuracy.

AMS subject classifications: 76D07, 76M25, 65N12, 65N15, 35B45.

Key words: Stokes’ equations, deep neural network method, first-order velocity-pressure-vorticity formulation.

1 Stokes’ equations

Recently, the deep neural network (DNN) methods have attracted remarkable attention in the field of computational fluid dynamics [9,33,57,61]. In contrast to classical methods such as finite element, finite difference, and finite volume, the DNN approach does not...
require a mesh topology and can achieve good accuracy even when the domain of interest is presented by scattered discrete points \[50\]. Moreover, it requires less number of parameters to achieve the same accuracy as with finite element method on the grid points \[37\]. And it can lessen or even overcome the curse of dimensionality for high-dimensional problems \[20\], it also has great potential in nonlinear approximations \[51, 55\]. However, the performance of existing DNN methods may degrade for the Navier-Stokes equations with low regularity, e.g. when sharp local gradients present in a broad computational domain \[39, 48\]. Noting that most DNN methods such as the recently developed physics informed neural network (PINN) method \[33\] employs the residual of equations as the loss function, which requires derivatives of variables that is only applicable to problems whose solutions are sufficiently smooth, for example \((u, p)\) at least in \([H^2(\Omega)]^2 \times H^1(\Omega)\) for the Stokes equations, and is not able to cope properly with problems that have non-smooth solutions. As exactly solving the Stokes’ equations with low regularity is the preliminary for many Navier-Stokes applications such as the lid driven cavity problems and other problems with singular sources and sharp interfaces \[2, 15, 30\], etc. In this paper, we propose a deep neural network method for the Stokes’ equations with reduced regularity, i.e. \((u, p) \in [H^1(\Omega)]^2 \times H^1(\Omega)\).

For simplicity, consider the following Stokes’ problem which seeks a velocity field \(u\) and a pressure unknown \(p\) satisfying

\[
\begin{cases}
-\nu \Delta u + \nabla p = f, & \text{in } \Omega, \\
\nabla \cdot u = 0, & \text{in } \Omega, \\
u = g, & \text{on } \Gamma, \\
\end{cases}
\] (1.1)

where \(\Omega\) is a bounded polygonal domain with boundary \(\Gamma = \partial \Omega\) in \(d (d = 2 \text{ or } 3)\) dimension, \(\nu > 0\) is a constant viscosity parameter, \(f = (f^{(1)}, \ldots, f^{(d)})\) represents external source, \(g\) is a given Dirichlet boundary condition, and the pressure \(p\) is assumed to have mean value zero; i.e., \(\int_{\Omega} p d\Omega = 0\). Noting that the classical solution of (1.1) satisfies \(u \in [C^2(\Omega) \cap C^0(\Omega)]^d\) and \(p \in C^1(\Omega)\).

Various deep learning based methods have been proposed and explored recently for the simulation of partial differential equations, such as the least square methods \[10, 17\], the deep Ritz methods \[19\], the physics-informed neural network (PINN) methods \[33, 51, 52\] and the variational formulation based methods \[34, 35\] among many others \[11, 28, 29, 42, 55\]. Here we roughly classify these methods according to their formulation of the loss functions; the reader is referred to \[4, 35\] and the references therein for a more detailed review. The least square methods may trace back to the 1990s., Dissanayake and Phan-Thhien \[17\] have proposed training neural networks via a least square functional that based on the original formulation of the PDEs. Thereafter the method have been further developed and it has been shown that the sampling points can be obtained by a random sampling \[26\], which is beneficial for high dimensional problems. Despite these advantages, for a second-order PDE, the minimization of the loss function over admissible functions leads to a fourth-order PDE, which is a more difficult
problem than the original one \cite{10}. Moreover, the method is only applicable to problems whose solutions are sufficiently smooth, more precisely, at least in $H^2(\Omega)$. Several methods have been recently proposed by transforming the original PDEs to the corresponding weak form so as to lower the regularity requirement. For example, the deep Ritz method \cite{19} uses the energy functional of the underlying PDEs as loss functions, hence only first-order spatial derivatives is required for the second-order elliptic PDEs. However, the method only applies to problems that have a underlying minimization principle. In addition to that, variational type neural network methods \cite{34,35} have been proposed. These methods train the neural network by minimizing variational residuals of PDEs over a set of test functions with their compact supports located at different regions in space and time. The loss function is expected to be discretization free and highly parallelizable \cite{35} and the methods also employ lower order derivatives. For instance, numerical experiments have shown good performance of the variational type methods for advection-diffusion problems with high Peclet numbers.

As for the Stokes problems, which is the main concern of this paper, different deep learning methods based on the least square functional of the original PDEs have been proposed and verified in \cite{11,40,51}. As mentioned previously, these methods are only applicable to problems whose solutions are sufficiently smooth, for $(u, p)$ at least in $[H^2(\Omega)]^2 \times H^1(\Omega)$. Since the Stokes’ problem does not have an underlying minimization principle, methods such as the deep Ritz method do not apply. As for the variational type methods, it is well known that the variational formulation of the Stokes’ problem leads to a saddle-point problem. Consequently, the combination of velocity and pressure interpolations is required to satisfy the famous Ladyzhenskaya-Babuška-Brezzi (LBB) condition \cite{22,41}. On the other side, by using different activation functions such as Sigmoid, deep neural network can give rise to a very wide range of functional classes that can be drastically different from the piecewise polynomial function classes used in classical finite element methods. These function classes, however, do not usually form a linear vector space \cite{64} which makes it tricky to apply the variational type neural network methods to Stokes’ equations.

In this paper, we propose a neural network method that is based on the first-order velocity-pressure-vorticity (VPV) system \cite{30,32}, namely VPVnet, to approximate the solution of the Stokes’ equations. The least square functional of the VPV system leads to a minimization problem rather than a saddle point problem, thus the combination of velocity, vorticity and pressure interpolations is not subject to the restriction of the LBB conditions. And no artificial boundary conditions need to be devised for the vorticity. Moreover, the VPV least square functional only requires first-order spatial derivatives, hence it has less regularity requirements on the solutions and is applicable to a much larger class of problems compared with least square methods that based on the original form of the Stokes’ equations. Note that similar ideas of making use of first order systems has also been used for the second-order elliptic PDEs and verified with one dimensional test cases in \cite{10}, but have not been studied for more complex systems such as the Stokes’ equations. Wang et. al. \cite{61} developed a multi-scale deep neural network
method for computing oscillatory Stokes flows in complex domains by using first order systems. We have established an error estimate for the proposed VPVnet method, which guarantees the convergence of the method for a small value of the loss function and vice versa, i.e. when the approximate solution is close to the exact solution, the loss function is of small value. It’s also worth emphasizing that the VPVnet method is divergence-free and pressure-robust, while classical inf-sup stable mixed finite elements for the Stokes’ equations are not pressure-robust. We use deep neural network comprised of ResNet blocks to approximate the solutions of the Stokes’ equations, which is shown to have stronger predictive ability than traditional feedforward neural networks. The loss function is computed using Gauss-Legendre quadrature rules and the optimization algorithm is composed of two steps: an Adam optimizer with self-adaptive learning rates followed by a limited-memory Broyden Fletcher Goldfarb Shanno algorithm with bound constraints (L-BFGS-B) [44] to finetune the results, which improves and accelerates the optimization process. Furthermore, the approximate and expressive ability of the VPVnet method is illustrated with non-smooth test cases in both 2D and 3D.

The paper is organized as follows. In Section 2, we present the first order velocity-pressure-vorticity formulation for the Stokes’ problem (1.1) and its relative least square functionals. In Section 3, we describe the structure of the VPV neural networks for approximating the solutions of the Stokes’ problem. The convergence and error estimates of the method is established in Section 4. In Section 5, several numerical results will be presented to demonstrate the efficiency and accuracy of the method. Finally, conclusions and discussions are presented in Section 6.

Throughout this paper, we use the standard notation and definition for the Sobolev spaces $H^s(\Omega)$ and $H^s(\Gamma)$ with inner products and norms denoted by $(\cdot, \cdot)_{s,\Omega}$ and $(\cdot, \cdot)_{s,\Gamma}$ and $\| \cdot \|_{s,\Omega}$ and $\| \cdot \|_{s,\Gamma}$, respectively. When there is no confusion, $\Omega$ (and $\Gamma$) is often omit in the indices for simplicity. $H^s_0(\Omega)$ is defined as

$$H^s_0(\Omega) = \{ v \in H^s(\Omega) : v|_{\partial\Omega} = 0 \},$$

and $L^2_0(\Omega)$ denotes the subspace of square integrable functions with zero mean. We also set $H^{s}(\Omega) = H^s(\Omega) \cap L^2_0(\Omega)$. And by $(\cdot, \cdot)_{(s_1, \ldots, s_n)}$ and $\| \cdot \|_{(s_1, \ldots, s_n)}$ we denote inner products and norms, respectively, on the product space $H^{s_1} \times \cdots \times H^{s_n}$; when all $s_i$ are equal, we shall simply write $(\cdot, \cdot)_{s,\Omega}$ and $\| \cdot \|_{s,\Omega}$.

## 2 Velocity-pressure-vorticity formulation of Stokes’ equations and least square functionals

For the sake of clarity, we consider the VPV formulation in 2D in this section. The 3D formulation can be found in [6,32] and will not be detailed here, while numerical experiments will be conducted in both 2D and 3D in Section 5. By introducing the vorticity $w = \nabla \times u$ as an auxiliary variable, and using the identity $\nabla \times (\nabla \times u) = -\Delta u + \nabla (\nabla \cdot u)$, in
addition with the incompressibility constraint \( \nabla \cdot u = 0 \), the Stokes equation (1.1) can be written as

\[
\begin{align*}
\nu \nabla \times w + \nabla p &= f, & \text{in } \Omega, \\
\nu (w - \nabla \times u) &= 0, & \text{in } \Omega, \\
\nabla \cdot u &= 0, & \text{in } \Omega,
\end{align*}
\] (2.1)

or

\[
\begin{align*}
p_x + \nu w_y - f^{(1)} &= 0, & \text{in } \Omega, \\
p_y - \nu w_x - f^{(2)} &= 0, & \text{in } \Omega, \\
\nu (w + u_y - v_x) &= 0, & \text{in } \Omega, \\
u u_x + v_y &= 0, & \text{in } \Omega,
\end{align*}
\] (2.2)

with a velocity boundary condition

\[
u = g \text{ on } \Gamma, \tag{2.3}
\]

and a constraint condition on the pressure

\[
\int_\Omega pd\Omega = 0. \tag{2.4}
\]

We have the following proposition [6]:

**Proposition 2.1.** The boundary value problem (2.1) (or (2.2)), (2.3) and (2.4) is equivalent to the Stokes’ problem (1.1) in primitive variable form. The problem has a unique solution for all smooth data \( f^{(1)}, f^{(2)} \) and \( g \).

Besides the boundary condition (2.3), various boundary conditions are possible for system (2.1), and the solvability of the boundary-value problem depends on the combination of the boundary conditions [6]. More generally, let \( U = (u, w, p) \), where \( u = (u, v)^T \), the VPV formulation of the Stokes’ equations can be written in general form of a first-order system

\[
\begin{align*}
\mathcal{L} U &= \begin{pmatrix} 0 & 0 & \nu \frac{\partial}{\partial y} & \frac{\partial}{\partial x} \\
0 & 0 & -\nu \frac{\partial}{\partial x} & \frac{\partial}{\partial y} \\
\nu \frac{\partial}{\partial y} & -\nu \frac{\partial}{\partial x} & v & 0 \\
\frac{\partial}{\partial y} & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} u \\
v \\
w \\
p \end{pmatrix} = \begin{pmatrix} f^{(1)} \\
f^{(2)} \\
0 \\
0 \end{pmatrix} = F, \text{ in } \Omega, \tag{2.5}
\end{align*}
\]

\[\mathcal{R} U = G, \text{ on } \Gamma, \tag{2.6}\]

where \( \mathcal{L} \) is an elliptic operator in the sense of Dougis and Nirenberg [1], and \( \mathcal{R} \) represents the boundary operators. Following the study in [6], we consider two choices for the boundary operators. The first one is the boundary condition (2.3) that imposes the velocity on the boundary, which can be rewritten in the following form

\[
\mathcal{R}_1 U = \begin{pmatrix} u^0 \\
v^0 \end{pmatrix} \text{ on } \Gamma, \tag{2.7}
\]
where $u^0$ is a given function defined on $\Gamma$. The second boundary operator imposes the pressure and the normal component of velocity,

$$\mathcal{R}_1 U = \begin{pmatrix} u_n^0 \\ p^0 \end{pmatrix} \text{ on } \Gamma,$$

(2.8)

where $u_n$ represents the normal component of velocity, $u_n^0$ and $p^0$ are given functions defined on $\Gamma$. Recall that one also has to impose the additional constraint on the pressure (2.4) to guarantee the uniqueness of solutions.

2.1 Least square functional

We consider the least square functional for the VPV system (2.5)-(2.6) that is defined in terms of the norms indicated by the Agmon-Douglis-Nirenberg theory [1]. The reader is referred to [6, 7, 16, 30] for more details concerning its derivation, only the main results are recalled here. Let $\mathcal{L}$ be characterized by two systems of integers $\{s_i\} = \{s_1, s_2, s_3, s_4\}$, $s_i \leq 0$, and $\{t_j\} = \{t_1, t_2, t_3, t_4\}$, $t_j \geq 0$, which are attached to the equations and the unknowns, respectively. Here variable $s_i$ corresponds to the $i$th equation and $t_j$ to the $j$th dependent variable. The least square functional for system (2.5)-(2.6) is defined as

$$J(U) = \|\mathcal{L} U - F\|_{(-s_1, -s_2, -s_3, -s_4)}^2$$

$$= \|\nu \nabla \times w + \nabla p - f\|_{(-s_1, -s_2)}^2 + \|\nu (w - \nabla \times u)\|_{-s_3}^2 + \|\nabla \cdot u\|_{-s_4}^2,$$

(2.9)

where $\{s_i\}$ and $\{t_j\}$ are indices for which the operators $\mathcal{L}$ and $\mathcal{R}$ satisfy the ellipticity supplementary and complementing conditions [6,13] and that the VPV system has a unique solution. The minimization of (2.9) is well-posed over a suitable subspace $U$ of $H^1_\mathcal{L}(\Omega) \times H^2_\mathcal{P}(\Omega) \times H^3_\mathcal{P}(\Omega) \times H^4_\mathcal{P}(\Omega)$. The values of $\{s_i\}$ and $\{t_j\}$ depend on particular boundary operators, which will be specified later.

The least squares principle is then given by: seek $U = (u, w, p) \in U$ such that

$$J(U) \leq J(U), \quad \forall \ U = (u, w, p) \in U.$$  

(2.10)

And we have the following results.

**Proposition 2.2.** The problem (2.10) has a unique minimizer $U \in U$, and there exists a constant $C > 0$ such that

$$\|u\|_{(t_1, t_2)}^2 + \|w\|_{t_3}^2 + \|p\|_{t_4}^2 \leq C \|f\|_{(-s_1, -s_2)}^2,$$

(2.11)

where $\{s_i\}$ and $\{t_i\}$ are indices such that the operators $\mathcal{L}$ and $\mathcal{R}$ satisfy the ellipticity supplementary and complementing conditions and that the VPV system has a unique solution [6,13].
Then we specify the above results to different boundary operators. For homogenous pressure normal velocity boundary condition (2.8), we can choose the indices \( s_i = 0 \) and \( t_j = 1 \), then the least square functional involves only \( L^2(\Omega) \)-norms of the residuals of all the equations, i.e. we have that

\[
J(U) = \| \mathcal{L} U - F \|_0^2
\]

(2.12)

and \( U = H^1_n(\Omega) \times H^1(\Omega) \times H^1(\Omega) \), where \( H^1_n(\Omega) \) denotes the subspace of \( H^1(\Omega) \times H^1(\Omega) \times H^1(\Omega) \) whose members have normal components equal to zero on the boundary.

For the case when system (2.5) is supplemented with the homogenous velocity boundary condition (2.7), we have \( s_1 = s_2 = 0, s_3 = s_4 = -1 \) and \( t_1 = t_2 = 2, t_3 = t_4 = 1 \), thus the least square functional now involves \( H^1(\Omega) \)-norms of the residuals of some of the equations, we have

\[
J(U) = \| \mathcal{L} U - F \|_{(0,0,1,1)}^2
\]

(2.13)

and \( U = (H^2(\Omega) \cap H^1_0(\Omega))^2 \times H^1(\Omega) \times \tilde{H}^1(\Omega) \). This implies that the velocity field should be approximated in finite-dimensional subspace of \( H^2(\Omega) \), i.e. continuous differentiable functions. Hence the straightforward application of the LS functional (2.13) bears no direct advantage over least square principle based on the primitive variable stokes equations. In order to use merely continuous functions, we shall consider a partition-dependent functional [6] which will involve only weighted \( L^2 \)-norms of the residuals, so the least square functional becomes

\[
J(U) = \| \mathcal{L} U - F \|_{(0,0,1,1)}^2 + h^{-1} \| \mathcal{L} U - F \|_{0,1}^2 + \| \nabla \cdot u \|_{0,1}^2,
\]

(2.14)

where \( h \) is the meshsize of the elements of a regular partition of the domain \( \Omega \), which will be detailed later in Section 3.3. We can now choose to approximate each unknown in a subspace of \( H^1 \), i.e \( U = [H^1_0(\Omega)]^2 \times H^1(\Omega) \times \tilde{H}^1(\Omega) \).

### 3 Deep least square neural network method

This section describes the VPVnet method for the Stokes’ equations. Let \( (u_\theta, w_\theta, p_\theta) \) be an approximation of the solutions of the first-order velocity-pressure-vorticity formulation (2.5) of the Stokes’ equations obtained with a neural networks \( U_\theta \). We hope to learn the parameters \( \theta \) of the neural network such that \( (u_\theta, w_\theta, p_\theta) \) approximate the solutions \( (u, w, p) \) as well as possible. To this end, we define a loss function based on the least square functional (2.9) of the problem, which will be presented in subsection 3.2. The structure of the deep neural network \( U_\theta \) is first described as follows:
3.1 DNN comprised of ResNet blocks

The solutions of the Stokes’ equations are approximated by a deep neural network comprised of ResNet blocks, which takes spatial coordinates as input and predicts the corresponding velocity, vorticity and pressure fields, i.e., $U_{\theta} : (x, y) \mapsto (u_{\theta}, \omega_{\theta}, p_{\theta})$. The construction of the DNN contains mainly the following two steps:

First, we construct a neural network which takes $x = [x, y]$ as input and neural networks $Y = [\psi_{\theta}(x, y), \omega_{\theta}(x, y), p_{\theta}(x, y)]$ as output. It is comprised of 4 ResNet blocks, each block consists of two linear transformations, two activation functions and a residual connection, and both the inputs $s$ and the output $t$ of the block are vectors in $\mathbb{R}^m$ (see Figure 1 a schematic illustration of the Resnet networks). The $i$th block can be expressed as:

$$t = \mathcal{F}_i(s) = \sigma(W_{i,2} \cdot \sigma(W_{i,1}s + b_{i,1}) + b_{i,2}) + s \tag{3.1}$$

where $W_{i,1}, W_{i,2} \in \mathbb{R}^{m \times m}$, $b_{i,1}, b_{i,2} \in \mathbb{R}^m$ are parameters associated with the block. $\sigma$ is the activation function. The dimension of $s$ and $t$ must be equal in (3.1), if it is not the case, a linear projection $W_s$ by the shortcut connections can be performed to match the dimensions, i.e.:

$$t = \mathcal{F}_i(s) = \sigma(W_{i,2} \cdot \sigma(W_{i,1}s + b_{i,1}) + b_{i,2}) + W_s s. \tag{3.2}$$

It has been shown by experiments in [25] that the identity mapping is sufficient for addressing the degradation problem and is economical, and thus $W_s$ is only used when matching dimensions.

Second, based on the neural networks $Y = [\psi_{\theta}(x, y), \omega_{\theta}(x, y), p_{\theta}(x, y)]$, we derive the neural networks for $u_{\theta}(x, y)$, $v_{\theta}(x, y)$ by using the automatic differentiation. We define

$$u_{\theta} := \frac{\partial \psi_{\theta}}{\partial y}, \quad v_{\theta} := -\frac{\partial \psi_{\theta}}{\partial x}. \tag{3.3}$$

Hence we have $\frac{\partial u_{\theta}}{\partial x} + \frac{\partial v_{\theta}}{\partial y} = \frac{\partial^2 \psi_{\theta}}{\partial y \partial x} - \frac{\partial^2 \psi_{\theta}}{\partial x \partial y} = 0$, here we define $u_{\theta}$ and $v_{\theta}$ in this way such that the conservation law (i.e. $\nabla \cdot u_{\theta} = 0$) is satisfied directly by its definition. The networks $u_{\theta}$, $v_{\theta}$ can be derived in a variety of ways, here we use automatic differentiation method [3] available in Tensorflow during the programming, e.g. $u_{\theta} = \text{tf.gradients}(\psi_{\theta}, y)[0]$. Note that Tensorflow uses reverse mode automatic differentiation, also known as backpropagation, for its gradient operation.
Figure 1: A schematic illustration of the ResNet networks for the construction of $\mathcal{F}_{\theta_i}$.

**Remark 3.1.** The form of the ResNet block is flexible, which may contain two or three layers, or even more layers are possible. But if the block has only a single layer, Eq (5.1) is in fact a linear layer, for which no advantages have been observed in [25]. The parameters $W$ and $b$ can be randomly initialized using the Xavier scheme (which returns truncated normal distribution with the standard deviation of $1/m$). As the choice of the activation function $\sigma$, $\tanh$, $\text{sinusoid}$, $\text{sigmoid}$, etc. are possible, which will be verified and tested later in Section 5. Note that in [35], the authors remarked that for variational type method $\tanh$ might be incapable of capturing some solutions.

### 3.2 Loss function

Let $(u_\theta, w_\theta, p_\theta)$ be an approximation of the solutions of system (2.5)–(2.6) obtained with the neural networks $U_\theta(x)$. We define a loss function based on the least square functional (2.9) which is written in the following form:

$$J(U_\theta) = \| \mathcal{L}U_\theta - F \|_{2}^{2} + \alpha \| \mathcal{R}U_\theta - G \|_{1/2,\Gamma}^{2}$$  \hspace{1cm} (3.4)

where we seek $U_\theta \in H^{1}(\Omega) \times H^{2}(\Omega) \times H^{3}(\Omega) \times H^{4}(\Omega)$, $\{s_i\}$ and $\{t_i\}$ are admissible values for specific boundary operators, $\alpha$ is a positive constant, and the sobolev norm $\| \cdot \|_{1/2}$ is defined as

$$\| \ell \|_{1/2,\Gamma}^{2} := \int_{\Gamma} |\ell|^{2} d\sigma + \int_{\Gamma \times \Gamma} \frac{|\ell(x) - \ell(y)|^{2}}{|x - y|^{d}} d\Gamma(x) d\Gamma(y).$$  \hspace{1cm} (3.5)

The treatment of boundary conditions is of great importance for numerical simulations. For the sake of clarity, here we take into account the boundary conditions (2.6) directly in the least square functional. By the trace theorem, the boundary norm is of $1/2$ order weaker than the interior norm, hence the norm $\| \cdot \|_{1/2}$ is employed. Note that the extra
penalty terms due to the boundary conditions could have negative effects on the training process and the final accuracy, several efforts have been made recently for the deep neural network methods to deal with boundary conditions. For example, Liao et al. proposed a Nitsche method [43] to deal with the essential boundary conditions without significant extra computational cost. Sheng et al. [54] developed a penalty free neural network methods, where they employ two networks, rather than just one, to construct the approximate solutions of the PDEs. With one networks satisfying the essential boundary conditions and the other handling the rest part of the domain, an unconstrained optimization problem is to be solved instead of a constrained one. We refer to the references above and herein [38, 49] for possible improvement treatment of the boundary conditions.

Thus the VPVnet method is: find $U_\theta: (x, y) \mapsto (u_\theta, w_\theta, p_\theta)$ such that

$$J(U_\theta) = \min_{\hat{\theta} \in \mathbb{R}^n} J(U_{\hat{\theta}}).$$

(3.6)

And we have the following remark:

**Remark 3.2.** Since $\nabla \cdot u_\theta$ vanish by construction of the neural network (see section 3.1), the neural network $U_\theta$ can be in fact trained by minimizing the loss function:

$$J(U_\theta) = \|v \nabla \times w_\theta + \nabla p_\theta - f\|_{(-s_1, -s_2)}^2 + \|v(w_\theta - \nabla \times u_\theta)\|_{2}^2 + \alpha \|\mathcal{R}U_\theta - G\|_{2/2, \Gamma}^2.$$  

(3.7)

More precisely, for the pressure normal velocity boundary condition (2.8), we have

$$J(U_\theta) = \|v \nabla \times w_\theta + \nabla p_\theta - f\|_{0}^2 + \|v(w_\theta - \nabla \times u_\theta)\|_{0}^2 + \alpha \|(u_{n, \theta}, p_\theta)^T - (u_{n, 0}, p^0)^T\|_{1/2, \Gamma}^2.$$  

(3.8)

For the velocity boundary condition (2.7), i.e. (2.3), we have

$$J(U_\theta) = \|v \nabla \times w_\theta + \nabla p_\theta - f\|_{0}^2 + h^{-2}\|v(w_\theta - \nabla \times u_\theta)\|_{0}^2 + \alpha \|u_\theta - g\|_{1/2, \Gamma}^2.$$  

(3.9)

### 3.3 Numerical computation of loss function

We consider evaluating the loss functions (3.7) by numerical approximations, i.e. we use Gauss-Legendre quadrature rules to compute the integration of the loss functions. Once the loss function is computed, the gradient of the numerical approximation can be naturally calculated using automatic differentiation algorithm available in Tensorflow for the optimization process thereafter. Similarly as in [10], the computation make use of a partition of the domain. Let $\mathcal{D}_h$ be an uniform square grid partition of the polygonal domain $\Omega \subset \mathbb{R}^2$ (See figure 2). Denote by $h := \max_{D \in \mathcal{D}_h} h_D$ the meshsize of $\mathcal{D}_h$, where $h_D = \text{diam}(D)$ is the diameter of the element $D \in \mathcal{D}_h$. Denote by $\mathcal{E}_h$ the edge set of $\mathcal{D}_h$, and $\mathcal{E}_h^0 \subset \mathcal{E}_h$ the set of all interior edges. The set of boundary edges is denoted as $\mathcal{E}_h^B :=$.
\[ E_h \setminus E^0_h \]. The diameter of the edge \( e \in E_h \) is denoted as \( h_e = \text{diam}(e) \). Throughout the paper, we employ one-point Gaussian quadrature, i.e. only the element center of each grid is considered as the quadrature point. We use \( x_D \) and \( x_E \) to denote the quadrature points in element \( D \) and on boundary \( e \), respectively. Since the Sobolev norm \( \| \cdot \|_{1/2} \) is not computationally feasible, we approximate it by weighted \( L^2 \) norms with local weights \( h^{-1/2}_e \). By (3.8), we have the following discrete loss function

\[
\hat{J}(\hat{U}_\theta(x)) \approx \sum_{D \in D_h} \left[ (p_x + v w_y - f^{(1)}_{(x_D, \theta)}) + (p_y - v w_x - f^{(2)}_{(x_D, \theta)}) \right] |D| + \sum_{D \in D_h} v^2 h^{-2} (w + u_y - v_x)^2_{(x_D, \theta)} |D| + \alpha \sum_{e \in E_h} h_e^{-1} (u - g_D)^2_{(x_E, \theta)} |e|,
\]

for the velocity boundary condition (2.7), where \( \alpha \) is positive constants, \( |D| \) and \( |e| \) are the measures of element \( D \) and boundary \( e \). The discrete loss function for the pressure normal velocity boundary condition (2.8) can be derived similarly using (3.9) and will not be detailed here. The discrete VPVnet method is to find \( \hat{U}_\theta : (x, y) \mapsto (\hat{u}_\theta, \hat{w}_\theta, \hat{p}_\theta) \) such that

\[
\hat{J}(\hat{U}_\theta) = \min_{\hat{\theta} \in \mathbb{R}^n} \hat{J}(\hat{U}_{\hat{\theta}}).
\]

The discrete loss function is lower bounded by zero, since this bound is attainable for the exact solution of the Stokes’ equations, the values of the loss function reflects how well the functions \( (\hat{u}_\theta, \hat{w}_\theta, \hat{p}_\theta) \) approximate the solution of the Stokes’ equations. In practice, the problem (3.11) is solved by an iterative method, such as the stochastic gradient decent method, which will be detailed in the section of numerical experiments.

**Remark 3.3.** Note that here we use Gauss-Legendre quadrature rules relying on a partition of the domain to compute the integration of the loss functions. In this way, the numerical error of evaluating the integral can be computed to a desired accuracy through

![Figure 2: A uniform partition of domain \( \Omega \) and related quadrature points (QPs) for the computation of the LS functional.](image-url)
either uniform or adaptive partition of the domain, which can be easily estimated and controlled. Otherwise, we can sample some points in the domain and use those sampled points to define the discrete loss function, then the method become a meshless method. For example, we sample \( M_i \) points inside \( \Omega \), \( M_{\partial \Omega} \) points on \( \Gamma \), and define the discrete loss function by the following weighted mean squared error,

\[
\hat{J} \left( \hat{U}_{\theta}(x) \right) \approx \sum_{M_i} \left[ (p_x + \nu w_y - f^{(1)}(x_{Mi}, \theta))^2 \right] + \alpha_2 \sum_{M_{\partial \Omega}} \left( u - g_{M_{\partial \Omega}} \right)^2 \left( x_{\partial \Omega}, \theta \right),
\]

where \( \alpha_2 \) is a parameter to be determined and we adopt \( \alpha_2 = 2500 \) in the computation of test cases (5.1a) and (5.1b) on irregular domain (cf. Figure 7). The derivative of the discrete loss function is calculated using automatic differentiation algorithm available in Tensorflow.

4 Convergence analysis of VPVnet method

According to the universal approximation property [27], the neural networks can approximate any continuous function with small errors. Hence approximation errors of the neural network \( U_{\theta} \) can be small for any \( U = (u, w, p) \in [C^0(\Omega)]^4 \). In this section, we will analyze the convergence of the VPVnet method. The following is a relevant theorem [5, 63] that will be used in the demonstration:

**Theorem 4.1.** Suppose that \( \sigma \in C^\infty(\mathbb{R}) \), \( \sigma^{(v)}(0) \neq 0 \) for \( v = 0, 1, \cdots \), and \( K \subset \mathbb{R}^d \) is any compact set. If \( f \in C^k(K) \), then a function \( \varphi_n \) represented by a DNN with complexity \( n \in \mathbb{N} \) exists such that

\[
\| \mathbf{D}^\alpha \varphi_n \|_{C(K)} = O \left( \frac{1}{n^{(k-|\alpha|)/d}} \right),
\]

where \( \alpha, \beta \) with \( |\alpha| \leq k, |\beta| = k \), where

\[
\omega(g, \delta) = \sup_{x, y \in K, |x - y| \leq \delta} |g(x) - g(y)|.
\]

Recall that the VPV system (2.5)-(2.6) is a uniformly bounded elliptic system in the sense of Douglis and Nirenberg. The system is well posed and has a unique solution. Hence, the minimization problem (3.6) has a unique solution, with the value of the infimum being 0, and the infimum is attained at the solution \( U \). In this study, for simplicity, we consider only the case \( s_i = 0 \) and \( t_j = 1 \), and \( G = 0 \), although the general case is similar. According to proposition 2.2, we have the following a priori estimate:

\[
\| u \|^2 + \| w \|^2 + \| p \|^2 \leq C \| f \|^2.
\]
First, we show that when the neural network $U_\theta$ and the actual solution $U$ are close to each other, we can control the loss function \((3.4)\).

**Theorem 4.2.** Let $U = (u, w, p) \in [H^1(\Omega)]^4$ be the solution of system \((2.5)-(2.6)\), then there exists $U_\theta : (x, y) \mapsto (u_\theta, w_\theta, p_\theta)$ with $\|U - U_\theta\|_1 \leq \varepsilon$ such that

$$J(U_\theta) = \|\mathcal{L}U_\theta - F\|_0^2 + \|\mathcal{R}U_\theta\|_{1/2, \Gamma}^2 \leq c\varepsilon^2 \text{ and } \|U_\theta\|_1 \leq \tilde{M}, \quad (4.4)$$

where $c$ is a positive constant.

**Proof.** We remark first that given any $\varepsilon > 0$, by Theorem 4.1, there exists a DNN $U_\theta$ such that $\|U - U_\theta\|_1 \leq \varepsilon$. It follows that

$$\|\mathcal{L}U_\theta - F\|_0^2 = \|\mathcal{L}U_\theta - \mathcal{L}U\|_0^2 \leq C_L^2 \|U_\theta - U\|_1^2 \leq C_L^2 \varepsilon^2, \quad (4.5)$$

where $C_L$ is the operator norm bound of $\mathcal{L}$. Since $G = 0$, we also have

$$\|\mathcal{R}U_\theta\|_{1/2, \Gamma} = \|\mathcal{R}U_\theta - G\|_{1/2, \Gamma} = \|\mathcal{R}U_\theta - \mathcal{R}U\|_{1/2, \Gamma} \leq C_R \|U_\theta - U\|_{H^1(\Omega)} \leq C_R \varepsilon^2, \quad (4.6)$$

where $C_R$ is the norm bound of the boundary operator. Combining \((4.5)\) and \((4.6)\), we have

$$J(U_\theta) = \|\mathcal{L}(\mathcal{R}U_\theta) - F\|_0^2 + \|\mathcal{R}U_\theta\|_{1/2, \Gamma}^2 \leq c\varepsilon^2. \quad (4.7)$$

Moreover, by \((4.3)\), we have

$$\|U_\theta\|_1 \leq \|U_\theta - U\|_1 + \|U\|_1 \leq \varepsilon + M = \tilde{M}, \quad (4.8)$$

where $M$ is the norm bound of $f$. 

Second, we show that by controlling the loss function \((3.4)\), we can have an approximation $U_\theta$ to the solution $U$ with small errors. The error estimate is given in the theorem below.

**Theorem 4.3.** Let $U = (u, w, p) \in [H^1(\Omega)]^4$ be the solution of system \((2.5)-(2.6)\) and let $U_\theta : (x, y) \mapsto (u_\theta, w_\theta, p_\theta)$ be such that

$$J(U_\theta) = \|\mathcal{L}U_\theta - F\|_0^2 + \|\mathcal{R}U_\theta\|_{1/2, \Gamma}^2 \leq \varepsilon^2. \quad (4.9)$$

Then there exists a constant $c > 0$ such that

$$\|U - U_\theta\|_0 \leq \|U - U_\theta\|_1 \leq c\varepsilon. \quad (4.10)$$
Proof. Denote $\mathcal{L}U_\theta = F_\varepsilon$ and $\mathcal{R}U_\theta = G_\varepsilon$. From (4.9), we have $\|F_\varepsilon - F\|_0 \leq \varepsilon$ and $\|G_\varepsilon\|_{1/2, \Gamma} \leq \varepsilon$. Consider the inverse boundary operator $\mathcal{R}^{-1} : [H^{1/2}(\Gamma)]^2 \rightarrow [H^1(\Omega)]^4$, which is linear and bounded such that $\mathcal{R} \mathcal{R}^{-1} = I$, where $I$ is the identity operator. Let $\tilde{U}_\theta = U_\theta - \mathcal{R}^{-1}G_\varepsilon$, we have

$$\begin{cases}
\mathcal{L}\tilde{U}_\theta = F_\varepsilon - \mathcal{L}\mathcal{R}^{-1}G_\varepsilon, & \text{in } \Omega, \\
\mathcal{R}\tilde{U}_\theta = 0, & \text{on } \Gamma.
\end{cases} \tag{4.11}$$

Subtracting system (4.11) into system (2.5)-(2.6) leads to

$$\begin{cases}
\mathcal{L}\tilde{U}_\theta - \mathcal{L}U = F_\varepsilon - \mathcal{L}\mathcal{R}^{-1}G_\varepsilon - F, & \text{in } \Omega, \\
\mathcal{R}\tilde{U}_\theta - \mathcal{R}U = 0, & \text{on } \Gamma,
\end{cases} \tag{4.12}$$

Note that $\mathcal{R}^{-1} \in [H^1(\Omega)]^4$, since $\mathcal{L}$ is a fourth order elliptic operator, we have $\mathcal{L}\mathcal{R}^{-1}G_\varepsilon \in [H^{-3}(\Omega)]^4$. Then by the Lax-Milgram, we have

$$\|\tilde{U}_\theta - U\|_1 \leq C_{\mathcal{L}^{-1}} \|(F_\varepsilon - F) - \mathcal{L}\mathcal{R}^{-1}G_\varepsilon\|_{-3}, \tag{4.13}$$

Therefore, we have

$$\|U_\theta - U\|_1 = \|U - (U_\theta - \mathcal{R}^{-1}G_\varepsilon) - \mathcal{R}^{-1}G_\varepsilon\|_1 \tag{4.14}$$

$$\leq \|U - \tilde{U}_\theta\|_1 + \|\mathcal{R}^{-1}G_\varepsilon\|_1$$

$$\leq C_{\mathcal{L}^{-1}} \|(F_\varepsilon - F) - \mathcal{L}\mathcal{R}^{-1}G_\varepsilon\|_{-3} + C_{\mathcal{R}^{-1}} \|G_\varepsilon\|_{1/2, \Gamma}.$$ 

Since

$$\|F_\varepsilon - F\|_{-3} \leq \|F_\varepsilon - F\|_0 \leq \varepsilon, \tag{4.15}$$

and

$$\|\mathcal{L}\mathcal{R}^{-1}G_\varepsilon\|_{-3} \leq C_{\mathcal{L}} \|\mathcal{R}^{-1}G_\varepsilon\|_1 \leq C_{\mathcal{L}} C_{\mathcal{R}^{-1}} \|G_\varepsilon\|_{1/2, \Gamma}. \tag{4.16}$$

Thus

$$\|U - U_\theta\|_0 \leq \|U - U_\theta\|_1 \leq c \varepsilon, \tag{4.17}$$

The numerical error of the VPVnet method mainly comes from three parts, the approximation/expressive error caused by the deep neural network $U_\theta$, the numerical error of evaluating the least square functional/neural network through numerical quadratures, and the algebraic error caused by the optimization algorithms. As shown in Theorems 4.2 and 4.3, we can obtain a neural network such that $\|U_\theta - U\| \leq \varepsilon$ by minimizing the loss function (3.6). On the other side, the Gauss-Legendre quadrature is a classical method that is often used for approximating the value of integrals. The numerical error of evaluating the least square functional can be computed to a desired accuracy through either
uniform or adaptive partition of the domain $\Omega$ and $\Gamma$ \cite{10}. Note that common methods of optimization such as stochastic gradient descent (SGD) has been employed in this paper. Although the convergence of the optimization methods depends on many computation factors and is still a hot topic of research, as we observed in section 5, the optimization algorithm employed works well for the test cases considered.

5 Numerical experiments

The purpose of this section is two-fold. First, we shall verify the performance and accuracy of the VPVnet method on some synthetic test cases. Second, we will demonstrate the capability of the VPVnet method to approximate non-smooth/singular solutions of the Stokes' equations, using test cases with singular solutions on L-shape domains, and a lid-driven cavity problem will be considered.

All the experiments are replicated three times to reduce variability of random initialization of the method of gradient decent and the medians of three training results are reported. The optimization algorithm is composed of two steps: we first use the Adam optimizer \cite{36} for e.g. 5,000 iterations with self-adaptive learning rates as proposed in Tensorflow, then apply the limited-memory BroydenFletcherGoldfarbShanno algorithm with bound constraints (L-BFGS-B) \cite{44} to finetune the results, so that errors from inexact optimization are suppressed. The training process of L-BFGS-B is terminated automatically based on the increment tolerance. Note that although both the loss functions for the pressure normal velocity boundary condition (3.8) and for the velocity boundary condition (3.9) provide satisfying numerical approximations, only numerical results obtained with the loss function (3.9) (or (3.12) in discrete form) is presented in the following, as the velocity boundary condition is most commonly used in practice. The results for the pressure normal velocity boundary condition will not be present here for simplicity.

5.1 Smooth test case

Consider the following test case defined in domains $\Omega = (0,1)^d, d=2,3$ such that (5.1a) and (5.1b) are the exact solutions. The parameter $\nu = 1$ in these test cases. The external source $f$ and the Dirichlet boundary data $g$ is chosen such that the above functions $u$ and $p$ are solutions to the system (1.1).

\[
\begin{align*}
    u(x,y) &= \begin{pmatrix} \sin(x)\sin(x)\cos(y)\sin(y) \\ -\cos(x)\sin(x)\sin(y)\sin(y) \end{pmatrix}, & p(x,y) &= \cos(x)\cos(y), \\
    u(x,y,z) &= \begin{pmatrix} x+x^2+xy+x^2y \\ y+xy+y^2+x^2y^2 \\ -2z-3xz-3yz-5x^2yz \end{pmatrix}, & p(x,y,z) &= xyz+x^3y^2z - \frac{5}{32}.
\end{align*}
\]
We first examine the accuracy of the VPVnet method by varying the number of quadrature points and the number of layers and neurons. We use neural networks of different structures to approximate the solution of the Stokes’ problem (5.1a). For example, a neural network comprised of 2 Resnet blocks, i.e. 4 hidden layers, and each hidden layer admits 8 neurons, with total 267 trainable parameters, is one of the neural networks considered. For simplicity, we note the neural networks by its number of hidden layers $N_L$ and the number of neurons $N_N$ within each hidden layers, hence the above neural network is noted as a neural network of $4 \times 8$. We also use neural networks of $8 \times 8$, $4 \times 16$, $8 \times 16$ hidden layers and neurons, with total 555, 915, 2003 trainable parameters, respectively, to approximates the solutions of the model problem (5.1a).

The discrete loss functions (3.12) is considered for the VPVnet method. The quadrature points (QPs) for the evaluation of the loss function is uniform grid data on the domain, one point quadrature rules is considered, which consist of $50 \times 50$ (or $20 \times 20$) points in total. Hence we have $h=1/50$ (or $h=1/20$). And we choose $\alpha=1$. QPs for the boundary...
condition of \( u, v \) is uniformly distributed points on the boundary of \( \Omega \), which contains 50 (or 20) points on each edge, i.e. 200 (or 80) points in total. An illustration of these QPs is shown in Fig 2. As mentioned previously, the optimization problem is first computed with 2000 Adam iterations, and followed by at most 5000 L-BFGS-B iterations, the learning rates are determined by existing self-adaption algorithms in Tensorflow.

Figure 5: Convergence of loss function over optimization iteration with different optimization processes for test case 5.1(a) and 5.1(b), the curves are: (1) Two step optimization : Adam part; (2) Two step optimization : L-BFGS-B part; (3) Adam optimizer only; (4) Adam optimizer with a hyperparameter of learning rate \( 5 \times 10^{-4} \) with exponential decay rate 0.98; (5) L-BFGS-B optimizer only.

Figure 5 is the convergence of loss function over different optimization iterations. As we can observe both in Fig. 5(a) and 5(b) the two-step approach allow a much faster convergence and even a better precision than just using Adam only. This is because BFGS uses the Hessian matrix (curvature in highly dimensional space) to calculate the optimization direction and provides more accurate results. However, if BFGS is used directly without using the Adam optimizer, the optimization can rapidly converge to a local minimum [47]. For this reason, the Adam optimizer is used first to avoid local minima, and then the solution is refined by BFGS. Here we use the L-BFGS-B method instead of the BFGS method in our work.

Figure 6 show the convergence of loss function over optimization iteration obtained with two different neural network structures for test case 5.1(a), the first is neural network with Resnet (Res) blocks, the second is feedforward (FF) neural network, note that these two neural networks have the same number of 8 hidden layers and each hidden layers have the same number of 20 neurons, except that the neural network with Resnet blocks has a residual connection every two layers. We can see that the neural network with Resnet blocks converges faster and gives a better precision than classical feedforward network. Hence neural network with Resnet blocks is adopted for all the subsequent test cases in this work.

The numerical approximations obtained with the VPVnet method (using \textit{sinusoid} activation function) for the speed, velocity and pressure are plotted in Figure 3. The \( L_2 \)
Figure 6: Convergence of loss function over optimization iterations obtained with different neural network structures for test case 5.1(a): (1) neural network with ResNet blocks, 2000 Adam optimization iteration plus L-BFGS-B optimization process; (2) feedforward neural network, 2000 Adam optimization iteration plus L-BFGS-B optimization process; (3) feedforward neural network, 5000 Adam optimization iteration plus L-BFGS-B optimization iterations.

Table 1: Numerical errors of the VPVnet method for test case (5.1a).

| Network | VPVnet method |
|---------|---------------|
|         | 20 × 20 Quadrature points | 50 × 50 Quadrature points |
|         | $e_u$ | $e_v$ | $e_p$ | $e_u$ | $e_v$ | $e_p$ |
| 4 × 8   | 7.88e-4 | 8.68e-4 | 1.36e-2 | 7.23e-4 | 7.58e-4 | 5.16e-3 |
| 8 × 8   | 8.35e-4 | 8.26e-4 | 6.62e-3 | 5.09e-4 | 5.45e-4 | 3.31e-3 |
| 4 × 16  | 6.84e-4 | 5.50e-4 | 1.02e-2 | 3.46e-4 | 4.47e-4 | 5.71e-3 |
| 8 × 16  | 5.24e-4 | 5.49e-4 | 4.56e-3 | 4.34e-4 | 4.99e-4 | 5.51e-3 |

Table 1 shows the numerical errors of each variable $u, v, p$ are presented in Table 1. We observe that for this smooth test case, we can have a rather good approximation with a small number of hidden layers and neurons. Although the numerical error varies due to the variability of random initialization of the parameters, we can observe that by using either more hidden layers, more number of neurons or more quadrature points, a tendency of better precision can be observed within certain limits.

Table 2: Numerical error of the VPVnet method for test case (5.1a) using different activation function, a neural network of 8 × 16 hidden layers and neurons, and 50 × 50 quadrature points.

| Activation function | VPVnet method |
|---------------------|---------------|
| $\sin(x)$          | $e_u$ | $e_v$ | $e_p$ |
| $\tanh(x)$         | 4.34e-4 | 4.99e-4 | 5.51e-3 |
| $\text{Sigmoid}(x)$| 5.11e-4 | 5.20e-4 | 9.98e-3 |

Table 2 shows the $L_2$ numerical errors using different activation functions. The errors are obtained using a neural network with 8 × 16 hidden layers and neurons, and 50 × 50 quadrature points. The results obtained with activation functions $\sin(x)$, $\tanh(x)$ and $\text{Sigmoid}(x)$ are all of satisfying precision. We remark that the one of most commonly
used activation function ReLU is not considered here, since the function is not differentiable everywhere, while first-order derivative is presented in the loss functions and we use the automatic differentiation, i.e. `tf.gradient()` to approximate the derivative, which will cause problems. A possible remedy to this problem might be using \( \max(0, x^3) \) instead of ReLU := \( \max(0, x) \) as proposed in [19]. Or we can approximate the derivative by using the finite difference quotient, as did by Cai et al. in [10]. The choice of finite difference quotient for the approximation of the derivative of different variable in 2D or 3D is a question that needs careful consideration. For the sake of clarity, we avoid such investigations here. In the rest of this paper, without specification, we will use \( \sin(\cdot) \) as the activation function.

We use neural networks of 8×16 hidden layers and neurons to approximate the solution of the 3D test case (5.1b). The quadrature points for the evaluation of the loss function is uniform grid data in the cube domain, which consist of 20×20×20 points in total. Note that BFGS instead of the L-BFGS-B algorithm is used to finetune the results obtained with the Adam optimizer in this 3D test case, as a better precision can be achieved by using the former. The \( L_2 \) numerical errors obtained with the VPVnet method (using \( \sinusoid \) activation function) is 1.24e−5, 1.37e−5, 1.37e−5, 4.85e−5, 4.95e−3, for variable \( (u, p) \), where \( u = (u^{(1)}, u^{(2)}, u^{(3)})^T \) in 3D, respectively. Similar precisions are reached for \( \tanh \) and \( \sigmoid \) activation functions, which will not be listed here for simplicity. The numerical approximations obtained with the VPVnet method for the speed, velocity and pressure are plotted in Figure 4. The numerical results show that VPVnet method is accurate and efficient in 3D.

The test cases (5.1a) and (5.1b) have been performed on more complex geometries using random sampling points. In this case, the loss function (3.12) is employed. The test case (5.1a) is preformed on a butterfly-like domain \( \Omega \). The boundary \( \Gamma \) of the domain \( \Omega \) is defined by the following equation [56]:

\[
\Gamma(x, y) = \{(x, y)^T \in \mathbb{R}^2 : x(\theta) = r(\theta)\cos(\theta), y(\theta) = 1.4r(\theta)\sin(\theta), \theta \in [0, 2\pi]\},
\]

where \( r(\theta) = 2\sqrt{\cos^2(5\theta) + \cos^2(2\theta)} + \cos^2(\theta) \). We used 200 and 2500 random sampling points (cf Fig. 7(a)), respectively, on the boundary and inside of the computation domain. As for the 3D test case (5.1b), the domain boundary is described by the zero iso-value of the function \( \phi \) [24]:

\[
\Gamma(x, y, z) = \{(x, y, z)^T \in \mathbb{R}^3 : \phi(x, y, z) = (x^2 + \frac{2}{3}y^2 + z^2 - 1)^3 - x^2z^3 - \frac{9}{32}y^2z^3 = 0\},
\]

2500 and 8000 random sampling points are employed (cf Fig. 7(c)), respectively, on the boundary and inside of the computation domain. Convergence can be easily obtained for both test cases using the same optimization processes as previously used. The numerical results (cf. Fig. 7) show that the method is very flexible on complex geometries. The \( L_2 \) numerical errors obtained with the VPVnet method (using \( \sinusoid \) activation function) is 3.20e−03, 3.16e−03 and 3.34e−02 for variable \( (u, p) \), where \( u = (u^{(1)}, u^{(2)})^T \) in 2D, re-
Figure 7: Numerical results of test case 5.1(a) and test case 5.1(b) obtained with VPVnet method using random sampling points in irregular domains.

spectively (and $5.88e-05$, $1.30e-04$, $2.58e-05$, $7.77e-3$ for $(u, p)$ in 3D). The precision can be further improved by adaptive sampling [62] which is the subject for future study.
5.2 Pressure-robust test case

In this example, let $\Omega = (0,1)^2$, and choose the proper right-hand side function such that the following functions satisfy system (1.1).

$$u(x,y) = \left( -e^x (y \cos(y) + \sin(y)) \right), \quad p(x,y) = 2e^x \sin(y).$$

(5.2)

Since $-\Delta u = (-2e^x \sin(y), -2e^x \cos(y))^T$, it is easy to check that when $\nu = 1$, we have a homogeneous equation with $f = 0$. When $\nu = 0$, $f$ is a nonzero function. It is well known that the classical finite element methods, such as the mini element, Taylor-Hood element, etc, suffer from a common lack of robustness when $\nu \ll 1$. The velocity error can become extremely large. It is also called poor mass conservation sometimes, since for $H^1$-conforming mixed methods such large velocity errors are accompanied by large divergence errors. In this test case, we will report the robustness results of the VPVnet method.

![Figure 8: Numerical results of divergence of $u$ for test case (5.2) obtained with VPVnet method using $50 \times 50$ uniformly distributed quadrature points in $\Omega$.](image)

![Table 3: Numerical error of the VPVnet method with different value of $\nu$ for test case (5.2), using a neural network with $8 \times 16$ hidden layers and neurons, and $20 \times 20$ and $50 \times 50$ quadrature points.](table)

The numerical experiments are carried out for $\nu = 1e-2, 1e-4, 1e-6$. The numerical results are obtained using a neural network with $8 \times 16$ hidden layers and neurons, and $20 \times 20$ and $50 \times 50$ quadrature points. The optimization problem is computed with 2000 Adam and at most 5000 L-BFGS-B iterations, respectively. The $L_2$ numerical errors
obtained with the VPVnet method of variable $u, v, p$ are presented in Table 3. We obtain very good approximations of the solutions for any value of $\nu$, including the value $\nu$ very small. The divergence of the velocity $\text{div}u = u_x + v_y$ for different values of $\nu$ is plotted in Figure 8. Hence the VPVnet method is divergence-free and pressure-robust.

5.3 Smooth test case on L-shape domain

For further examination, the more complicated domain, L-shape domain, is considered to validate in this research. The domain is defined by $\Omega = (-1,1)^2 \setminus ([0,1] \times [-1,0])$. It is called the backward facing step domain of flow problems. The parameter $\nu$ is of value 1 for all domain, $\nu = 1$. The test case is of exact solution as shown in Eq. (5.2). The same test case has been considered in [59] and many other references.

![Numerical results of test case](image)

We obtain good approximations for the test case on L-shape domain with the VPVnet method. Note that the test case has the same exact solution as (5.2) in subsection 5.2 hence similar computation configuration has been used and the numerical errors are not shown here again. The numerical approximations obtained with the VPVnet method for
the speed, velocity, pressure and the streamline are plotted in Figure 9.

5.4 Singular test case on L-shape domain

Hereafter, we consider a singular solution on the L-shaped domain \( \Omega \):= \((-1,1)^2 \setminus [0,1) \times (-1,0) \). The test case is proposed by Verfürth in [58] and has been widely used in the literature [14,41]. Let \( \omega = 3\pi/2 \), \( \delta = 0.5444837 \). Given \( (r,\theta) \) the polar coordinates at the re-entrant corner \((0,0)\), introduce the auxiliary function

\[
\psi(r,\theta) = \frac{\sin((1+\delta)\theta)\cos(\delta \omega)}{1+\delta} - \cos((1+\delta)\theta) - \frac{\sin((1-\delta)\theta)\cos(\delta \omega)}{1-\delta} + \cos((1-\delta)\theta). \tag{5.3}
\]

The singular solution we approximate is

\[
\begin{cases}
  u(x,y) = r^\delta \left((1+\delta)\sin(\theta)\psi(\theta) + \cos(\theta)\psi'(\theta)\right), \\
v(x,y) = r^\delta \left(\sin(\theta)\psi'(\theta) - (1+\delta)\cos(\theta)\psi(\theta)\right), \\
p(x,y) = r^{\delta-1} \left((1+\delta)^2\psi'(\theta) + \psi^{(3)}(\theta)\right) / (1-\delta).
\end{cases} \tag{5.4}
\]

The Stokes equation is homogeneous with this solution, i.e \( f = 0 \). Moreover, the Dirichlet conditions are homogeneous along the edges except at the re-entrant corner. We emphasize that the pair \((u,p)\) is analytical in \( \Omega \setminus \{(0,0)\} \), but \( \nabla u \) and \( p \) are singular at the origin. Especially, \( u \not\in [H^2(x)]^2(\Omega) \) and \( p \not\in H^1(\Omega) \). This solution reflects perfectly the typical behavior of the solution of the Stokes problem near a reentrant corner. In general, for solution \((u,p)\) to the Stokes problem in a non-convex domain \( \Omega \), we can expect \( u \in [H^s(\Omega)]^2 \) and \( p \in H^{s-1} \) for \( s < 1+\delta \). Hence the solution \( u \) is in \( C^0 \), and theoretically, the neural networks is capable of approximating a \( C^0 \) function at any error. It is expected by construction that the VPVnet method can capture such singular solutions, which is confirmed by the following numerical results.

| Quadrature points | VPVnet method | 
|-------------------|---------------|
| \( 20 \times 20 \times 3 \) | 2.79e-2, 2.22e-2, 5.21e-1 |
| \( 50 \times 50 \times 3 \) | 6.45e-3, 5.92e-3, 1.96e-1 |
| \( 100 \times 100 \times 3 \) | 8.60e-3, 8.63e-3, 3.17e-1 |

The computation configuration is as follows: The parameters \( \alpha = 1 \) and a neural network with \( 12 \times 16 \) hidden layers and neurons is employed for all experiments in this test case. Uniform partitions of size \( 20 \times 20 \times 3 \), \( 50 \times 50 \times 3 \) and \( 100 \times 100 \times 3 \) are considered. (Note that we use \( * \times * \times 3 \) QPs because the L-shape domain is composed of 3 square domain.) One point quadrature rules is employed for all cases. Hence we have \( h = 1/20 \) (or \( h = 1/50, h = 1/100 \)) for the uniform partitions. As for the boundary condition of \( u, v \), the partition is in accordance with the partitions interior domain. For the optimization,
Figure 10: Numerical results of test case (5.4) obtained with VPVnet method using $50 \times 50 \times 3$ quadrature points in $\Omega$. 

(a) Exact(left) & approximate(right) pressure for domain $(-1.0,1.0)^2$

(b) Exact(left) & approximate(right) pressure(zoom) domain $(-0.15,0.15)^2$

(c) Exact & approximate pressure plot in 3D

(d) Speed

(e) Velocity

(f) Streamline
2000 Adam iteration is used, while more L-BFGS-B iterations is needed (less than 50000 iterations), respectively.

The numerical solutions obtained with the VPVnet method for the speed, velocity, pressure and the streamline, and the exact solution for the pressure are are plotted in Figure 10. The numerical results in Table 4 clearly show that the VPVnet method can provide good approximations for $u \in [H^s(\Omega)]^2$, $2 > s > 1$. While the errors obtained with the VPVnet method for $p \in H^{s-1}$, $2 > s > 1$ are much larger, due to the strong pressure singularities near the re-entrant corner $(0,0)$, which can also be observed in Figure 10(a), 10(b) and 10(c). The numerical errors can be reduced by refinement within a certain limit, then it will not get any better, indicating more efforts are needed to reach a satisfying precision. Recall that only smooth activation function is considered here in this paper, and it has been shown in [10] that the piecewise activation functions, e.g. Leaky ReLU gives a better performance than the smooth activation functions for a interface problem, hence it would be interesting to try continuous piecewise activation functions as remarked in subsection 5.1 which is subject to further investigations.

5.5 Lid-driven cavity test cases

The lid-driven cavity test case has been used as a benchmark problem for many numerical methods due to its simple geometry and complicated flow behaviors [8]. The computation domain is $\Omega = (0,1)^2$ in 2D. And the Dirichlet boundary condition is given as $u_{\Gamma} = g = (1,0)^T$ for $y = 1$ and $g = (0,0)^T$ on the rest of the boundary.

![Quadrature points in 2D and 3D](image)

Figure 11: Local refined quadrature points of lid-driven cavity test case used for VPVnet method.

The main difficulty of this problem comes from the presence of singularities at corners of the domain. The pressure and the vorticity are not finite at the two top corners of the domain. Singularities also present at the corners $(0,0)$ and $(1,0)$, where the second
derivative of the velocity is unbounded, which is weaker than the previous ones. By calculating the norm, one can see that $g \in H^{1/2-\delta}(\partial\Omega, \mathbb{R}^2)$ for all $0 < \delta < 1/2$ but $g \notin H^{1/2}$, hence the velocity is not in $H^1$. Thus the usual variational formulation for the Stokes’ problem does not apply for the lid-driven cavity problems. However, this test case is used in many papers to test finite elements methods and the discontinuous boundary condition is usually applied directly to the finite element discretization in practice. In fact, on one hand, we can approximate the discontinuous boundary data by a smooth one and then apply a finite element method to the regularized problem, on the other hand, it has been observed that for properly defined continuous boundary condition, the discrete system will not see the difference between the discontinuous and the continuous boundary condition [10,18]. In this paper, we expect to explore the capacity of the VPVnet method to solve this problem.

We use the following computation configuration for 2D case computation: the parameters $\alpha = 1$ and a neural network with $12 \times 16$ hidden layers and neurons is employed. A locally refined partition of size $50 \times 50$ which is geometrically refined towards all the corners, i.e. $(0,0), (1,0), (0,1)$ and $(1,1)$, are considered. The locally refined partition has elements of size varies from $4.0e-3$ to $4.8e-2$, an illustration is shown in Fig. 11(a). One point quadrature rules is employed. We pick the value of $h = 1/50$. As for the bound-
ary condition of $u$, $v$, the partition is in accordance with the partitions interior domain, i.e. locally refined points on the boundaries of $\Omega$ are used. The discontinuous boundary condition is applied directly.

The numerical approximations obtained with the VPVnet method of the speed, velocity, pressure and the streamline for the lid-driven cavity test case are plotted in Figure 12. The exact solution of lid-driven problem is not known for the Stokes' problem. The shape of streamlines is similar to the result given in [41], strong pressure singularities are present at the top two corners, the results look quite reasonable. Note that, there are some perturbations which can be observed near the domain boundaries, particularly in the figure for the streamline [12(e)]. We refine the partition of the domain by 200 $\times$ 200 rectangular element with size from $1.0e^{-3}$ to $8.0e^{-3}$. The perturbations turns to diminish (see Fig. 13).

![Numerical results of lid-driven cavity test case](image)

We also consider the 3D lid-driven cavity test case in a unit cubic domain. A tangential velocity $u_T = (1,0,0)^T$ is imposed at the top surface of the domain, while $u = (0,0,0)^T$ is imposed on the rest of the boundary. The parameters $\alpha = 1$ and a neural network with $16 \times 16$ hidden layers and neurons is employed for the computation. Locally refined partitions of size $10 \times 10 \times 10$ and $20 \times 20 \times 20$ which are geometrically refined towards the domain corners are considered. The locally refined partition has elements of size varies
Figure 14: Numerical results of 3D lid-driven cavity test case obtained with VPVnet method using 1000 and 8000 locally refine quadrature points in $\Omega$.

from $2.0e^{-2}$ to $2.0e^{-1}$ for the $10 \times 10 \times 10$ partition, and $1.0e^{-2}$ to $1.0e^{-1}$ for $20 \times 20 \times 20$

6 Discussion and conclusion

A deep neural network method is proposed and verified to approximate the solution of the Stokes’ equations. The method makes use of least square functionals based on the first-order VPV formulation (2.1) of the Stokes’ equations as loss functions. It has less regularity requirements on the solutions compared with other methods such as the methods based on the original form of the PDEs [40,51]. Convergence and error estimates have been established for this method. We observed that smooth solutions of the Stokes’ equations can be well approximated by the VPVnet method using Resnet neural networks with a rather small number of neurons and hidden layers. Within a certain limit, a better precision can be obtained by using more hidden layers, more neurons and more quadrature points. Either smooth activation function, such as $\text{Sin}(x)$, $\text{Tanh}$, $\text{Sigmoid}$, etc can be employed and all of them can provide satisfying approximations. The method is divergence-free and pressure-robust.

We focus on the ability of the method for the approximation of Stokes’ equations with non-smooth/singular solutions, i.e. $u \notin H^2$ and $p \notin H^1$. For the velocity variable $u \in H^s$, $2 > s > 1 \ (\in C^0)$, the VPVnet methods can approximate it very well, which is in consistent with our theoretical analysis. As for the pressure variable $p \in H^{s-1}$, $2 > s > 1 \ (\notin C^0)$, the approximation turn to be more difficult. Refinement can help to improve the precision,
while more efforts is needed to reach a precision satisfying. The future work will be aimed at further investigation on this aspect. In the meantime, a variant of the proposed VPVnet method is under investigation, namely the VVGPnet method which is based on the first-order system of velocity/velocity gradient/pressure (VVGP) formulation [45] of the Stokes’ equations. The VVGP system also provides a $H^1$ least square functional and can be approximated similarly as the VPVnet method. The VVGPnet method is quite suitable for the approximation of the Stokes interface problems [60], which usually has jumps of the gradient of the velocity. The method will be detailed in a forthcoming paper dedicated for the interface problems.

The method presented here has already been extended to the incompressible Navier-Stokes Equations and will be discussed in a forthcoming paper.

**Acknowledgments**

The authors would like to thank Hailong Sheng for helpful discussions. The research of Liu was partially supported by China National Natural Science Foundation (No. 12001306), Guangdong Provincial Natural Science Foundation (No. 2017A030310285). The research of Yang (corresponding author) was funded in part by Beijing Academy of Artificial Intelligence.
References

[1] S. Agmon, A. Douglis, L. Nirenberg. Estimates near the boundary for solutions of elliptic partial differential equations satisfying general boundary conditions II, Communications on pure and applied mathematics, 17(1), 35-92.

[2] T. Arbogast, D. S. Brunson. A computational method for approximating a Darcy-Stokes system governing a vuggy porous medium. Computational Geosciences, 2007, 11(3):207-218.

[3] A. G. Baydin, B. A. Pearlmutter, A. Radul, J. M. Siskind, Automatic differentiation in machine learning: a survey. Journal of Machine Learning Research, 18(1), 5595-5637, 2017.

[4] C. Beck, M. Hutzenthaler, A. Jentzen, B. Kuckuck, An overview on deep learning-based approximation methods for partial differential equations. arXiv preprint :2012.12348, 2020.

[5] A. Biswas, J. Tian, S. Ulusoy. Error Estimates for Deep Learning Methods in Fluid Dynamics. 2020, arXiv preprint arXiv:2008.02844

[6] P. B. Bochev, M. D. Gunzburger, Analysis of least squares finite element methods for the Stokes equations[J]. Mathematics of Computation, 1994, 63(208): 479-506.

[7] P. B. Bochev, D. Gunzburger, Finite element methods of least-squares type. SIAM Review, 1998, 40:789-837.

[8] O. Botella, R. Peyret. Benchmark spectral results on the lid-driven cavity flow. Computers & Fluids, 1998, 27(4): 421-433.

[9] S. L. Brunton, B. R. Noack, P. Koumoutsakos, Machine learning for fluid mechanics, Annual Review of Fluid Mechanics, 2020, 52: 477-508.

[10] Z. Cai, J. Chen, M. Liu, and X. Liu. Deep least-squares methods: An unsupervised learning-based numerical method for solving elliptic PDEs. Journal of Computational Physics, 2020, 420: 109707.

[11] W. Cai, X. Li, L. Liu, A Phase Shift Deep Neural Network for High Frequency Approximation and Wave Problems. SIAM Journal on Scientific Computing, 42(5), A3285-A3312, 2020.

[12] Z. Cai, Y. Wang. An error estimate for two-dimensional Stokes driven cavity flow. Mathematics of computation, 2009, 78(266): 771-787.

[13] C. L. Chang, J. Bo-Nan. An error analysis of least-squares finite element method of velocity-pressure-vorticity formulation for Stokes problem. Computer methods in applied mechanics and engineering, 1990, 84(3), 247-255.

[14] A. Chernov, C. Marcati, L. Mascotto, p-and hp-virtual elements for the Stokes equation. SAM Research Report, 2020.

[15] R. Codina, O. Soto. Finite element solution of the Stokes problem with dominating Coriolis force. Computer Methods in Applied Mechanics and Engineering, 1997, 142(3-4):215-234.

[16] J. M. Deang and M. D. Gunzburger. Issues related to least-squares finite element methods for the Stokes equations. SIAM Journal on Scientific Computing, 20(3), 878-906, 1998.

[17] M. W. M. G. Dissanyake, N. Phan-Thien. Neural network based approximations for solving partial differential equations. Communications in Numerical Methods in Engineering, 10(3):195-201, 1994.

[18] R. Duran, L. Gastaldi, A. Lombardi, Analysis of Finite Element Approximations of Stokes Equations with NonSmooth Data. SIAM Journal on Numerical Analysis, 2020, 58(6): 3309-3331.

[19] W.-N. E, B. Yu, The Deep Ritz Method: A Deep Learning-Based Numerical Algorithm for Solving Variational Problems. Communications in Mathematics & Statistics, 2018, 6(1):1-12.

[20] W.-N. E, Machine Learning and Computational Mathematics. Communications in
[21] F. Gibou, D. Hyde, R. Fedkiw, Sharp interface approaches and deep learning techniques for multiphase flows. Journal of Computational Physics, 2019, 380: 442-463.

[22] V. Girault and P.-A. Raviart, Finite Element Methods for Navier-Stokes Equations, Springer-Verlag, Berlin, 1986.

[23] M. Gunzburger and J. Petersen, On conforming finite element methods for the inhomogeneous stationary Navier-Stokes equations, Numer. Math., 1983, 42,173-194.

[24] C. He, X. Hu, L. Mu. A Mesh-free Method Using Piecewise Deep Neural Network for Elliptic Interface Problems. arXiv preprint[arXiv:2005.04847] 2020.

[25] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, 770-778.

[26] J. He, L. Li, J. Xu, and C. Zheng. ReLU deep neural networks and linear finite elements[J]. arXiv preprint[arXiv:1807.03973] 2018.

[27] K. Hornik, Approximation capabilities of multilayer feedforward networks. Neural networks, 1991, 4(2): 251-257.

[28] A. Iskhakov, N. Dinh, Physics-integrated machine learning; embedding a neural network in the Navier-Stokes equations. arXiv preprint[arXiv:2008.10509] 2020.

[29] A. Iskhakov, N. Dinh, Physics-integrated machine learning; embedding a neural network in the Navier-Stokes equations. Part II. arXiv preprint[arXiv:2009.11213] 2020.

[30] B.-N. Jiang, C. L. Chang, Least-squares finite elements for the Stokes problem[J]. Computer Methods in Applied Mechanics and Engineering, 1990, 78(3): 297-311.

[31] B.-N. Jiang, A least-squares finite element method for incompressible Navier-Stokes problems[J]. International Journal for Numerical Methods in Fluids, 1992, 14(7): 843-859.

[32] B.-N. Jiang, T. L. Lin, L. A. Povinelli, Large-scale computation of incompressible viscous flow by least-squares finite element method[J]. Computer Methods in Applied Mechanics and Engineering, 1994, 114(3-4): 213-231.

[33] X. Jin, S. Cai, H. Li, G. E. Karniadakis, NSFnets (Navier-Stokes Flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations. arXiv preprint[arXiv:2003.06496] 2020

[34] E. Kharazmi, Z. Zhang, G. E. Karniadakis, Variational physics-informed neural networks for solving partial differential equations.

[35] R. Khodayi-Mehr, M. Zavlanos, VarNet: Variational neural networks for the solution of partial differential equations, In Learning for Dynamics and Control, 2020, 298-307.

[36] D. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint[arXiv:1412.6980] 2014.

[37] I. E. Lagaris, A. Likas, D. I. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, IEEE Transactions on Neural Networks, 1998, 9(5): 987-1000.

[38] I. E. Lagaris, A. C. Likas, D. G. Papageorgiou, Neural-network methods for boundary value problems with irregular boundaries, IEEE Transactions on Neural Networks, 2000, 11, 1041-1049.

[39] H. C. Lee, Efficient computations for linear feedback control problems for target velocity matching of Navier-Stokes flows via POD and LSTM-ROM. Electronic Research Archive, 2021, 29(3): 2533.

[40] J. Li, J. Yue, W. Zhang, W. Duan. The Deep Learning Galerkin Method for the General Stokes Equations. arXiv preprint[arXiv:2009.11701] 2020.

[41] R. Li, Z. Sun, F. Yang, Z. Yang, A finite element method by patch reconstruction for the Stokes problem using mixed formulations. Journal of Computational and Applied
[42] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, A. Anandkumar. Fourier neural operator for parametric partial differential equations. arXiv preprint arXiv:2010.08895, 2020.

[43] Y. Liao, P. Ming, Deep Nitsche Method: Deep Ritz Method with Essential Boundary Conditions, Communications in Computational Physics, 2021, 29: 1365-1384.

[44] D. C. Liu, J. Nocedal, On the limited memory BFGS method for large scale optimization, Mathematical programming, 1989, 45: 503-528.

[45] K. Liu. Hybrid First-Order System Least-Squares Finite Element Methods With The Application To Stokes And Navier-Stokes Equations, Doctoral dissertation, University of Colorado at Boulder, 2012.

[46] K. Liu, T. A. Manteuffel, S. F McCormick, J. W. Ruge, L. Tang. Hybrid first-order system least squares finite element methods with application to Stokes equations. SIAM Journal on Numerical Analysis, 2013, 51(4), 2214-2237.

[47] S. Markidis. Physics-Informed Deep-Learning for Scientific Computing. arXiv preprint arXiv:2103.09655, 2021.

[48] R. Ranade, C. Hill, J. Pathak, DiscretizationNet: A machine-learning based solver for Navier-Stokes equations using finite volume discretization. Computer Methods in Applied Mechanics and Engineering, 2021, 378: 113722.

[49] K. S. McFall, J. R. Mahan, Artificial neural network method for solution of boundary value problems with exact satisfaction of arbitrary boundary conditions, IEEE Transactions on Neural Networks, 2009, 20, 1221-1233.

[50] S. Mehrkanoon, J. A. K. Suykens, Learning solutions to partial differential equations using LS-SVM, Neurocomputing, 2015, 159: 105-116.

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

[52] M. Raissi, A. Yazdani, G. E. Karniadakis, Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations, Science, 2020, 367(6481): 1026-1030.

[53] C. Reisinger, Y. Zhang, Rectified deep neural networks overcome the curse of dimensionality for nonsmooth value functions in zero-sum games of nonlinear stiff systems. arXiv preprint arXiv:1903.06652, 2019.

[54] H. Sheng, C. Yang. PFNN: A Penalty-Free Neural Network Method for Solving a Class of Second-Order Boundary-Value Problems on Complex Geometries, Journal of Computational Physics, 2021, 427:110085.

[55] J. Sirignano, K. Spiliopoulos. DGM: A deep learning algorithm for solving partial differential equations. Journal of computational physics, 2018, 375, 1339-1364.

[56] Y. Tang, X. Li. A meshless complex variable Galerkin boundary node method for potential and Stokes problems. Engineering Analysis with Boundary Elements, 2017, 83: 204-216.

[57] N. Thuerey, K. Weißeneow, L. Prantl, X. Hu, Deep learning methods for Reynolds-averaged Navier-Stokes simulations of airfoil flows, AIAA Journal, 2020, 58(1): 25-36.

[58] R. Verfürth. A Review of a Posteriori Error Estimation and Adaptive Mesh-Refinement Techniques. Chichester-Stuttgart, Wiley-Teubner, 1996.

[59] T. Vu-Huu, C. Le-Thanh, H. Nguyen-Xuan, M. Abdel-Wahab, An equal-order mixed polygonal finite element for two-dimensional incompressible Stokes flows. European Journal of Mechanics-B/Fluids, 2020, 79, 92-108.

[60] B. Wang, B. C. Khoo, Hybridizable discontinuous Galerkin method (HDG) for Stokes inter-
face flow. Journal of Computational Physics, 247, 262-278.

[61] B. Wang, W. Z. Zhang, W. Cai, Multi-Scale Deep Neural Network (MsdeDNN) Methods for Oscillatory Stokes Flows in Complex Domains. Communications in Computational Physics, 2020, 28: 2139-2157.

[62] C. L. Wight, J. Zhao, Solving Allen-Cahn and Cahn-Hilliard equations using the adaptive physics informed neural networks. arXiv preprint arXiv:2007.04542, 2020.

[63] T. Xie, F. Cao, The errors of simultaneous approximation of multivariate functions by neural networks. Computers & Mathematics with Applications, 2011, 61(10), 3146-3152.

[64] J. Xu The finite neuron method and convergence analysis. arXiv preprint arXiv:2010.01458, 2020.