AppliedMath

MDPI

Article

QPDE: Quantum Neural Network Based Stabilization Parameter Prediction for Numerical Solvers for Partial Differential Equations

Sangeeta Yadav

Department of Computational and Data Sciences, Indian Institute of Science, Bengaluru 560012, Karnataka, India; sangeetay@iisc.ac.in

Abstract: We propose a Quantum Neural Network (QNN) for predicting stabilization parameter for solving Singularly Perturbed Partial Differential Equations (SPDE) using the Streamline Upwind Petrov Galerkin (SUPG) stabilization technique. SPDE-Q-Net, a QNN, is proposed for approximating an optimal value of the stabilization parameter for SUPG for 2-dimensional convection-diffusion problems. Our motivation for this work stems from the recent progress made in quantum computing and the striking similarities observed between neural networks and quantum circuits. Just like how weight parameters are adjusted in traditional neural networks, the parameters of the quantum circuit, specifically the qubits' degrees of freedom, can be fine-tuned to learn a nonlinear function. The performance of SPDE-Q-Net is found to be at par with SPDE-Net, a traditional neural network-based technique for stabilization parameter prediction in terms of the numerical error in the solution. Also, SPDE-Q-Net is found to be faster than SPDE-Net, which projects the future benefits which can be earned from the speed-up capabilities of quantum computing.

Keywords: partial differential equations; quantum neural networks

Citation: Yadav, S. QPDE: Quantum Neural Network Based Stabilization Parameter Prediction for Numerical Solvers for Partial Differential Equations. *AppliedMath* **2023**, *3*, 552–562. https://doi.org/10.3390/appliedmath3030029

Academic Editors: Artur Czerwinski and Xiangji Cai

Received: 4 May 2023
Revised: 11 June 2023
Accepted: 20 June 2023
Published: 13 July 2023

1. Introduction

Quantum computing has the potential to accelerate traditional algorithms in science and engineering, including the data-driven schemes for solving Partial Differential Equations (PDEs). Since this research area is quite new, we don't see much literature available for speeding up traditional solvers with quantum computing. There are a lot of problems in the literature on scientific computing which can harness the advantages of quantum computing. One such possibility is to improve traditional fluid flow simulations as they are compute-intensive and scale with the size of the problem at hand. To address this, we propose the development of an approach that leverages the advantages of quantum computing and deep learning methods for solving PDEs. Our main objective is to develop a Quantum Neural Network (QNN) for predicting stabilization parameters when solving SPDEs using the Streamline Upwind Petrov Galerkin (SUPG) technique. We introduce SPDE-Q-Net, a QNN for approximating the value of the stabilization parameter for SUPG in 2-dimensional convection-dominated problems. Our results show that the performance of SPDE-Q-Net is comparable to that of SPDE-Net, a traditional neural network-based technique for stabilization parameter prediction. This research has the potential to address a significant challenge in fluid flow simulations by providing stabilized and efficient solutions. By combining quantum computing with neural networks, our approach could lead to significant improvements in a wide range of fields, including engineering and science. However, further research is needed to address the complex challenges in this area and to fully realize the potential of this approach.

1.1. Neural Networks for Solving PDEs

The development of data-driven PDE solvers has spawned a plethora of studies. Several data-driven numerical PDE solving techniques exist such as Physics Informed Neural Networks (PINNs) [1,2], Fourier Neural Operator (FNO) [3,4]. PINNs use a neural network to approximate the numerical solution by minimizing the strong form of the residual. Their performance, however, is limited for singularly perturbed partial differential equations (SPDEs). To solve this issue, Yadav and Ganesan [5] proposed to use a hybrid scheme to solve SPDEs. This hybrid scheme uses Artificial Neural Networks (ANN) to predict the stabilization parameter and stabilized FEM for solving the equation. Having a neural network in the solving pipeline makes the entire process slow, so we proposed to use a QNN so that we can show a quicker convergence of the proposed neural network. In this current work, we leverage Quantum Computing to predict the stabilization parameter for solving SPDEs.

1.2. Quantum Computing

As explained in [6], Quantum computing generally refers to the controlled evolution of a set of quantum two-level systems of a 2-dimensional Hilbert space (H_2) with basis $|0\rangle, \langle 1|$, which are called qubits. This can better be explained with a wavefunction as

$$\begin{aligned} |\psi\rangle &= \alpha\,|0\rangle + \beta\,|1\rangle \\ |\alpha|^2 + |\beta|^2 &= 1, \alpha, \beta \in \mathbb{C} \end{aligned} \tag{1}$$

Unitary quantum gates manipulate qubits in the same way logic gates affect conventional bits in a computer, allowing qubits to evolve through so-called quantum channels. Quantum computers have been proposed as linear algebra accelerators, paving the way for quantum machine learning (QML). The use of quantum computing devices to do QML is still in its early phases of development [7–11]. While completely quantum algorithms have been shown to have an asymptotic advantage, deep protocols are still out of reach of current QC capabilities. This necessitates new ways based on a hybrid quantum-classical process, which is primarily motivated by deep learning's success in scientific computing. Data embedding circuits (feature maps) [6] and parametrized variational circuits are the building blocks of variable QML models, which may be trained to execute various unsupervised tasks [8]. These can very well be used for regression problems [12].

1.3. Quantum Neural Network

The primary idea behind introducing quantum features into classical Neural Networks is to replace the classical neurons $z = \{-1, 1\}$ with 'qurons' $|z\rangle$ with basis $\{|0\rangle, |1\rangle\}$ [13,14]. It is a parametrized quantum circuit. Quantum feature maps are used to encode the inputs to the feature space. This feature map is a unitary circuit consisting of x-dependent gates. When this feature map is applied to any initial state, it transforms them into quantum states. A variational quantum circuit then processes the quantum states.

A QNN is a machine learning model or algorithm that combines concepts from quantum computing and ANNs. QNN is used to represent variational or parametrized quantum circuits. While mathematically somewhat different from the inner workings of neural networks, the analogy highlights the "modular" nature of quantum gates in a circuit and the wide use of tricks from training neural networks used in the optimization of quantum algorithms. We extend the network proposed by [15]. Every input data is stored in s register, $|\alpha\rangle_i$, the hidden states will be computing the weighted sum of the inputs and will represent them as $|\psi_{ij}\rangle$ for j^{th} hidden unit.

1.4. Numerical Methods for Solving SPDEs

Higher-order convection-diffusion equations do not have analytical, closed-form solutions, and numerical methods are often used to approximate their solution. Mesh-based methods, such as Finite Difference, FE, and FV, are famous for the numerical simulation of

such equations. These techniques have a conservative formulation, higher-order accuracy, and the ability to accommodate complex geometries. Further, many of these methods are amenable to parallelization. However, solving SPDEs is challenging since it requires the numerical schemes to carefully treat high-gradient regions (boundary layer/interior layer) typically associated with such equations to avoid spurious oscillations. The numerical schemes often use a stabilization term to deal with the singularly perturbed equation. Several stabilization strategies exist, such as slope limiting, flux limiting, the introduction of artificial dissipation, etc. In the context of FEM, the SUPG method is a remarkably successful stabilization scheme that follows the philosophy of artificial dissipation. The scheme has been widely used to solve SPDEs for problems such as contaminant transport in shallow water, evolutionary convection-diffusion-reaction equations, arbitrary Lagrangian-Eulerian framework for time-dependent convection-diffusion equations, linear hyperbolic problems, and reduced order models among other areas. In addition to the SUPG technique, the local projection stabilization, edge stabilization, orthogonal subgrid scale, Galerkin least-squares [16] and continuous interior penalty method are some other popular techniques used to stabilize numerical methods for simulation of convection-dominated equations.

1.5. Streamline Upwind/Petrov Galerkin

Many techniques exist for solving SPDEs, most of them have some sort of stabilization in them. One such popular stabilization technique is the streamline upwind/Petrov Galerkin technique, generally called SUPG. It was introduced in [16] and is the focus of our current research work. It provides a nodally correct solution to the 1-d convection-diffusion problem. The accuracy of the SUPG technique for higher-dimensional problems is limited as it depends on the appropriate value of the stabilization parameter. This stabilization parameter is hard to find and is a topic of research for the last few decades.

1.6. On the Choice of Stabilization Parameter for SUPG

The value of the stabilization parameter has a significant impact on the correctness of the SUPG numerical solution. This parameter specifies how much artificial diffusion should be used in the equation to achieve a steady result. Artificial diffusion with a high value can cause the solution to smear, but under-diffusion will not eliminate the false oscillations. Many approaches for determining the ideal value of the stabilization parameter have been developed, but each has its own set of restrictions and is difficult to generalize to higher dimensions.

1.7. SPDE-Net

Many techniques have been proposed for estimating the value of the stabilization parameters. Yadav and Ganesan [5,17–19] employed a neural network to estimate the stabilization parameter for the SUPG methodology with finite element methods-based loss formulation for solving SPDEs. In [5], SPDE-Net is used as an optimizer to estimate the stabilization parameter (τ) for 1-dimensional problems. The performance of both supervised and unsupervised approaches is compared and shown to be comparable, implying that unsupervised training can be self-sufficient. Furthermore, the L^2 error from SPDE-Net was significantly lower than that from the state-of-the-art PINN networks. It suggests that we need to supplement PINN's loss formulation with a stabilization term for higher-dimensional challenges.

1.8. Contributions

- We propose a Quantum Neural Network for predicting stabilization parameters for solving SPDEs using the SUPG stabilization technique
- Developed an unsupervised quantum model for SPDE-Net, based on equation coefficients and the local gradients based normalization
- Compared the performance of classical neural network (SPDE-Net) and SPDE-Q-Net in terms of different errors such as L^2, H^1, L_∞, relative l^2

- SPDE-Q-Net has been tested for different mesh refinements to check its generalization ability

1.9. Organization of the Paper

In Section 1, we provide an introduction to the problem addressed in this research. Section 2 covers the necessary background and foundational information for the proposed work. We delve into the specifics of the proposed network architecture in Section 3. The experimental methodology, along with testing examples, is presented in detail in Section 4. Finally, we conclude the manuscript in Section 6, summarizing the key findings and their implications.

2. Preliminaries

In this paper, our focus is on solving the convection-diffusion equation, a fundamental equation in the field of computational science and engineering. The convection-diffusion equation arises in various domains, including fluid dynamics, heat transfer, and mass transport, and plays a crucial role in modeling physical phenomena. It combines the effects of advection (convection) and diffusion, making it a challenging problem to solve accurately and efficiently.

2.1. Convection-Diffusion Equation

$$-\epsilon\Delta u + \mathbf{b}\cdot\nabla u = f \text{ in } \Omega \text{ and } u = u_b \text{ on } \partial\Omega. \tag{2}$$

Here $\Omega \subset \mathbb{R}^2$, is a bounded domain with a polygonal Lipschitz-continuous boundary $\partial\Omega$, $\mathbf{n}$ is outward normal vector to the boundary $\partial\Omega$, $\epsilon > 0$ is the diffusion coefficient, $\mathbf{b} = [b_1, b_2] \in W^{1,\infty}(\Omega)^d$ is the flow velocity, $f \in L^2(\Omega)$ is the external source term, u is the unknown scalar term and $u_b \in H^{1/2}(\partial\Omega)$. First, the variational form of the equation is derived using the Galerkin technique. It is to find u such that for all $v \in H_0^1(\Omega)$

$$a(u, v) = (f, v) \tag{3}$$

where the bilinear form $a(\cdot,\cdot) : H^1(\Omega) \times H_0^1(\Omega) \to R$ is defined by

$$a(u, v) = \int_\Omega \epsilon\nabla u \cdot \nabla v dx + \int_\Omega \mathbf{b}\nabla u v dx \tag{4}$$

$$(f, v) = \int_\Omega f v dx \tag{5}$$

$(\cdot,\cdot)$ is the $L^2(\Omega)$ inner product. Let $\mathcal{T}_h$ be a family of triangulations of Ω parametrized by positive parameters h whose only accumulation point is zero. The triangulations $\mathcal{T}_h$ are assumed to consist of a finite number of open polygonal subsets K of Ω such that $\bar{\Omega} = \bigcup_{K\in\mathcal{T}_h} \bar{K}$. The domain Ω is partitioned into $\Omega_h \in P_2$ and the weak formulation for this discretized domain is to choose a finite-dimensional space $V_h \subset H_0^1(\Omega)$ comprising continuous piecewise polynomials and find $u_h \in V_h$ such that for all $v_h \in V_h$ we have

$$\begin{aligned} a_h(u_h, v_h) &= (f, v_h) \\ a(u_h, v_h) &:= \varepsilon(\nabla u_h, \nabla v_h) + (\mathbf{b}\cdot\nabla u_h, v_h) \\ &= (f, v_h) \quad \forall v_h \in V_h \end{aligned} \tag{6}$$

The solution of this variational form contains spurious oscillations near the boundary layer, so a weighted residual term has been added to it as explained in the next subsection.

2.2. SUPG Stabilization

The residual of equation [2] is :

$$R(u) = -\epsilon\Delta u + b\cdot\nabla u - f$$

The term $R(u)$ has been added to the discretized weak formulation given in Equation (6). Now, modified weak form for domain Ω_h, is to find $u_h \in V_h$ such that $\forall v_h \in V_h$:

$$\begin{aligned} &a_h(u_h, v_h) = f_h(v_h) \forall \quad v_h \in V_h \\ &\text{where, } a_h(u_h, v_h) = \epsilon(\nabla u_h, \nabla v_h) + (\mathbf{b} \cdot \nabla u_h, v_h) + \sum_{i \in \Omega_h} \tau_i(-\epsilon \Delta u_h + \mathbf{b} \cdot \nabla u_h - f_h, \mathbf{b} \cdot \nabla v_h)_{\Omega_h} \end{aligned} \tag{7}$$

$\tau_i \in L_2(\Omega)$ is a user-chosen non-negative stabilization parameter. Its value plays an important role in the quality of the approximated solution. A very large value can show unexpected smearing, whereas a low value will not remove the spurious oscillations. So, an optimal value of stabilization parameter is required, which can control oscillations and smearing both. Many studies have established the bounds for τ for the SUPG technique in order to achieve quasi-optimal numerical convergence. One such widely used expression is given as follows.

2.3. Standard Stabilization Parameter

There is a standard mathematical expression to calculate the value of the stabilization parameter and it gives a nodally precise solution for a 1-dimensional problem and simple 2-dimensional problems. However, it is not generalizable to complex 2-dimensional problems such as problems with variable coefficients. To make it generalized we are proposing a NN-based method for predicting the stabilization parameter.

$$\begin{aligned} \tau_{std}|_T = \frac{h_T}{2|\mathbf{b}|}\xi_0(Pe_T), Pe_T = \frac{|\mathbf{b}|h_T}{2\varepsilon} \\ \text{where } \xi_0(\alpha) = \coth\alpha - \frac{1}{\alpha}, \end{aligned} \tag{8}$$

We will call it Std. τ for the sake of convenience. We are using a piecewise constant element to define τ on the mesh. τ_K will be locally defined for each cell in the mesh.

3. Network Architecture

We predict a single stabilization parameter for the entire mesh and then locally divide it with the norm of gradients of the Galerkin solution in that cell. This way, we have taken care of cell-wise stabilization. By dividing with the gradient of the stabilization parameter, we obtain an adaptive τ. In this research, we use the QNN proposed by [15] and find the output for each input feature, namely $|\psi\rangle$ of size M. The predicted stabilization parameter can be represented as: $\hat{\tau} = \frac{\sum_{i=1}^{M}|\psi\rangle_i}{M}$.

3.1. SPDE-Q-Net

We use QNN for approximating the stabilization parameter τ. In a QNN, we use initial states and then map them to quantum bits (qubits) using a quantum feature map. Generally, the feature map consists of a unitary circuit $\hat{Q}_\phi(x)$, which is a concatenation of x-dependent gates and performs single qubit rotation. When applied to an initial state $|\varnothing\rangle$ (a quantum initial state), the feature map transforms $x \sim P_\Omega$ data points into quantum states $|\psi(x)\rangle := \hat{Q}_\phi(x)|\varnothing\rangle$. These quantum states are processed by variational quantum circuit $\hat{V}_\theta$ parametrized by parameters θ. Mathematically a QNN can be expressed as follows:

$$\begin{aligned} Input: I = \{\epsilon, b_1, b_2, h_T\}, Pe_T = \frac{|\mathbf{b}|h_T}{2\epsilon} \\ \tau_{std}|_T = \frac{h_T}{2|\mathbf{b}|}(\coth(Pe_T) - \frac{1}{Pe_T}) \\ u_{std} = Solve_{supg}(\tau_{std}) \end{aligned} \tag{9}$$

$$
\begin{aligned}
\hat{Q}_\theta(x) &= \prod_{r=1}^{R}(\hat{V}_\theta^{(r)}\hat{Q}_\theta^{(r)}(x))\hat{V}_\theta^{(0)} \\
\tau(\theta) &= \langle\varnothing|\,\hat{Q}_\theta(x)^\dagger\hat{C}\hat{Q}_\theta(x)\,|\varnothing\rangle\,/(|\nabla u_{std}|) \\
m(s) &= \begin{cases}\sqrt{s} & s>1\\ 2.5s^2-1.5s^3 & \text{otherwise}\end{cases} \\
\mathbf{b}^\perp &= \begin{cases}\dfrac{(b_2,-b_1)}{|\mathbf{b}|} & \text{when } |\mathbf{b}|\neq 0\\ 0 & \text{when } |\mathbf{b}|=0\end{cases} \\
Loss(\theta) &= \epsilon(\nabla u_h,\nabla v_h)+(\mathbf{b}\cdot\nabla u_h,v_h)+\sum_{i\in\Omega_h}\tau_i(-\epsilon\Delta u_h+\mathbf{b}\cdot\nabla u_h \\
&\quad -f_h,\mathbf{b}\cdot\nabla v_h)_{\Omega_h}-f_h(v_h)+m(|\mathbf{b}^\perp\cdot\nabla u_h|) \\
\theta^* &= argminLoss(I)
\end{aligned}
\tag{10}
$$

u_h is the SUPG solution of Equation (7), h_T is the mesh size, τ is stabilization parameter. We must learn $\hat{Q}$ by finding optimal θ^* through iterative back propagation of loss. SPDE-Q-Net has R number of layers, and initially, $\hat{V}_\theta^0$ operates the initial state. τ is the expected value of the cost operator. Here the cost operator is the weighted sum of Pauli operators, $\hat{C}=\sum_j c_j\hat{P}_j$ or fix $c_j=$ constant$\forall j$. SPDE-Q-Net has five input qubits and one output qubit. It has an Adam optimizer and Sigmoid activation function for six hidden layers with twenty quorons in each hidden layer.

3.2. Back Propagation in Quantum Neural Network

We discuss the back-propagation pipeline of SPDE-Q-Net here. We have numerically approximated the gradients of the expected value of cost by finite differences. It is affected by the quantum measurement noise, so there will be a bias involved in learning the network. The other way could be to use analytic derivatives of these expressions. Here, we use the parameter shift rule as described in [6]. Let us take an example case of unitary operator $\hat{U}(x)=exp(-\iota\phi(x)\hat{G}/2)$ generated by a Hermitian operator $\hat{G}$. $\psi(x)$ denotes an extra function applied to the feature variable, such as a feature non-linearity. Next, let us consider a simplified quantum model

$$q(x)=\langle\tilde{\psi}|\,\hat{U}^\dagger(x)\tilde{C}\hat{U}(x)\,|\tilde{\psi}\rangle \tag{11}$$

where $|\tilde{\psi}\rangle$ and $\tilde{C}$ are some generic input states and dressed cost operators that do not depend on x. The circuit derivative reads

$$\frac{d\tilde{U}(x)}{dx}=(-\iota\psi'(x)/2)\hat{G}\hat{U}(x) \tag{12}$$

and we can differentiate the quantum model as

$$\frac{dq(x)}{dx}=(\iota\psi'(x)/2)\,\langle\tilde{\psi}|\,\hat{U}^\dagger(x)[\hat{G},\tilde{C}]\hat{U}(x)\,|\tilde{\psi}\rangle \tag{13}$$

where $[\cdot,\cdot]$ denotes a commutator. In the general form, the expression above can be measured as an overlap between quantum states. However, to simplify the readout, derivatives of unitaries with a known spectrum of $\hat{G}$ can be calculated exactly (without bias) by evaluating the model at shifted arguments of $q(x)$. Specifically, for unitaries that Pauli operators generate $\hat{G}:=\hat{P}_j$ (i.e., single qubit rotations at site j), where operators satisfy involutory property $\hat{P}_j^2=\mathbb{1}_j$ and have just two unique eigenvalues of ±1. $g'(x)=\psi'(x)[g(x+\pi/2)-g(x-\pi/2)]/2$.

3.3. SPDE-Net

We compare the performance of SPDE-Q-Net with a classical neural network SPDE-Net trained for the same task and detailed as below. For a given input data sample,

$$\begin{aligned} Input: I &= \{\epsilon, b_1, b_2, h_T\}, Pe_T = \frac{|\mathbf{b}|h_T}{2\epsilon} \\ \tau_{std}|_T &= \frac{h_T}{2|\mathbf{b}|}(\coth(Pe_T) - \frac{1}{Pe_T}) \\ \tau(\theta) &= \frac{G_\theta(Pe_T)}{||\nabla u_{std}||_2} \\ Loss(\theta_k) &= (\int_\Omega (\hat{u}(\hat{\tau}(\theta_k)) - u)dx)^{\frac{1}{2}} \\ \theta^* &= argmin Loss(I, u(\tau(\theta))) \end{aligned} \tag{14}$$

where G_θ is θ parameterized SPDE(Net), $u(\tau(\theta))$ is the SUPG solution of Equation (7), and $\tau(\theta)$ is the predicted stabilization parameter. We have to learn G_θ by finding optimal θ^* by back propagating the loss iteratively.

3.4. Error Metrics

Here we discuss the metrics used to quantify the numerical errors in the solution. Four different error metrics are presented. The first is the L^2-error, denoted as $|e_h|0$, which measures the difference between the computed solution u_h and the known analytical solution u in the L^2 norm over the domain Ω. The relative l^2-error, denoted as $|e_h|0, \ell$, is a sum of the pointwise differences between u_h and u normalized by the l^2 norm of u at specific points x_i in the domain Ω_h. The H^1-error, denoted as $|e_h|1$, measures the difference between the gradients of u_h and u in the L^2 norm over Ω. Finally, the L^∞-error, denoted as $|e|L^\infty(\Omega)$, represents the essential supremum of the absolute difference between u_h and u over the domain Ω. We use the following metrics to calculate numerical errors in the solution.

$$\begin{aligned} L^2\text{-error: } \|e_h\|_0 &= \|u_h - u\|_{L^2(\Omega)} = \left(\int_\Omega (u_h - u)^2 dx\right)^{\frac{1}{2}} \\ \text{Relative } l^2\text{-error: } \|e_h\|_{0,\ell} &= \sum_{i=1}^{N} \frac{\|u_h(x_i) - u(x_i)\|_{0,\ell}}{\|u\|_{0,\ell}}, \qquad x_i \in \Omega_h \\ H^1\text{-error: } |e_h|_1 &= \|\nabla u_h - \nabla u)\|_{L^2(\Omega)} = \left(\int_\Omega (\nabla u_h - \nabla u)^2 dx\right)^{\frac{1}{2}} \\ L^\infty\text{-error: } \|e\|_{L^\infty(\Omega)} &= ess \quad sup\{|u_h - u| : x \in \Omega\}. \end{aligned} \tag{15}$$

Here, $\hat{\tau}$ is the predicted stabilization parameter, u is the known analytical solution, $u_h(\hat{\tau})$ is the SUPG solution calculated with $\hat{\tau}$, and D^α is the weak derivative.

3.5. Order of Convergence

Given a sequence of approximations $\{x_n\}$ converging to a true solution x as n approaches infinity, the order of convergence of a numerical method is defined as follows:

Let $\{\epsilon_n\}$ denote the sequence of errors, where $\epsilon_n = |x_n - x|$. If there exist positive constants C and p such that

$$\lim_{n\to\infty} \frac{\epsilon_{n+1}}{\epsilon_n^p} = C,$$

where p is known as the order of convergence, then the numerical method is said to converge with order p or is p-th order convergent. A higher value of p implies faster convergence. The value of C can provide additional information about the convergence behavior. If $C = 0$, the method is said to have *superlinear convergence*, whereas $C > 0$ indicates *sublinear convergence*. If $C = 1$, the method exhibits *linear convergence*, and if $C > 1$,

it is *superlinearly convergent*. The order of convergence provides a measure of how rapidly a numerical method approaches the true solution as the number of iterations increases.

4. Numerical Experiments

In this section, we consider four different examples of SPDEs based on the location of interior and boundary layers. For each example, we predict the τ using SPDE-Net, and SPDE-Q-Net. We compare the performance of different techniques in terms of errors metrices defined in Section 3.4. All these errors are computed with respect to the analytical solution. SPDE-Q-Net is optimized by minimizing the weak residual of the equation as mentioned in [20]. The network training was carried out for different mesh sizes, but here we only present the mesh results with 40 cells in the x and y-direction. The computation time taken by different techniques is given in Table 1.

Example 1. *We have taken a two-dimensional convection–diffusion problem as given by Equation (2) with the following data for cheking the performance of SPDE-Q-Net:*

$$\begin{aligned} &\epsilon = 10^{-8}, \quad \mathbf{b} = (1,0), \quad f = 1, \quad \Omega = (0,1)^2 \\ &u = 0 \quad on \quad \partial\Omega \end{aligned} \tag{16}$$

It has an exponential layer at $x = 1$ and two parabolic layers at $y = 0$ and $y = 1$, respectively. In the interior domain, the analytical solution $u(x, y)$ is quite close to x. The SPDE-Q-Net captures the pattern of τ in accordance with the location of the interior/boundary layer region. The L^2 error produced by all the considered techniques is compared in Table 2, and we can see that the L^2 error produced by SPDE-Q-Net is at par with SPDE-Net.

Example 2. *Next, we consider Equation (2) with following data:*

$$\begin{aligned} &\epsilon = 10^{-8}, \mathbf{b} = (2,3), \Omega = (0,1)^2, \quad u = 0 \quad on \quad \partial\Omega \\ &u(x,y) = xy^2 - xexp(\frac{(3(y-1))}{\epsilon}) - y^2 exp(\frac{(2(x-1))}{\epsilon}) \\ &+ exp(\frac{(2(x-1)+3(y-1))}{\epsilon}) \end{aligned} \tag{17}$$

It has two outflow boundary layers and is hence a suitable test case for checking the performance of the SPDE-Q-Net. We observed the clear pattern of the boundary layers in the heat map of the predicted τ. In the non-boundary region, the predicted τ is close to 1.0, whereas, in the boundary region, the value is close to the std. tau value and hence this localized τ ensures enough stabilization in the areas where it is required the most.

Example 3.

$$\begin{aligned} &\epsilon = 10^{-8}, \theta = -\pi/3, \mathbf{b} = (cos(\theta), sin(\theta)), \\ &f = 0.0, \Omega = (0,1)^2, u = u_b \quad on\partial\Omega, \quad \partial\Omega_N = \phi \\ &u_b = \begin{cases} 0, & for\ x = 1\ or\ y \leq 0.7 \\ 1, & otherwise \end{cases} \end{aligned} \tag{18}$$

This example contains both the exponential and boundary layer. The value of the standard stabilization parameter τ for this example is 0.0049, whereas all the predicted τ values are more than 0.01. Both the interior and exponential layers are very well captured by the SPDE-Q-Net and it is quantitatively shown in terms of different error metrics given in Tables 2–4.

Table 1. Computational Time: Average of the training time taken by all the four examples in seconds. N_c is the number of elements used to discretize the domain Ω in x-direction.

N_c	**10**	**20**	**30**	**40**
SPDE-Net	200	316	473	598
SPDE-Q-Net	40	96	154	189

Table 2. L^2 *Error*.

	Examples			
	1	**2**	**3**	**4**
Std. τ	1.32×10^{-5}	6.77×10^{-6}	1.41×10^{-5}	3.63×10^{-6}
Std. τ with $\|\|\nabla u\|\|$	4.04×10^{-5}	1.63×10^{-5}	1.76×10^{-5}	3.85×10^{-6}
SPDE-Net	7.03×10^{-6}	5.08×10^{-6}	1.33×10^{-5}	3.63×10^{-6}
SPDE-Q-Net	6.05×10^{-6}	5.04×10^{-6}	1.20×10^{-5}	3.63×10^{-6}

Table 3. Relative l^2 Error.

	Examples			
	1	**2**	**3**	**4**
PINN	4.85×10^{-1}	$4.85 \times 10^{+1}$	8.69×10^{-1}	
Std.	1.17×10^{-1}	1.36×10^{-1}	8.02×10^{-2}	4.63×10^{-2}
Std. with $\|\|\nabla u\|\|$	3.35×10^{-1}	3.06×10^{-1}	9.68×10^{-2}	4.90×10^{-2}
SPDE-Net	6.1×10^{-2}	9.9×10^{-2}	6.99×10^{-2}	4.80×10^{-2}
SPDE-Q-Net	6.17×10^{-2}	9.73×10^{-2}	6.94×10^{-2}	4.60×10^{-2}

Table 4. L^∞ Error.

	Examples			
	1	**2**	**3**	**4**
Std. τ	9.07×10^{-5}	7.29×10^{-5}	1.13×10^{-4}	7.60×10^{-6}
Std. τ with $\|\|\nabla u\|\|$	1.32×10^{-4}	8.70×10^{-5}	1.47×10^{-4}	1.06×10^{-5}
SPDE-Net	3.99×10^{-5}	4.45×10^{-5}	9.08×10^{-5}	7.96×10^{-5}
SPDE-Q-Net	3.95×10^{-5}	4.05×10^{-5}	9.08×10^{-5}	7.35×10^{-6}

Example 4.

$$\begin{aligned} &\epsilon = 10^{-8}, \quad \mathbf{b} = (1,0)^T, u = 0 \quad on \\ &\partial\Omega_D, \quad \partial\Omega_D = \partial\Omega, \quad \partial\Omega_N = \phi \\ &f = \begin{cases} 0, & if \quad |x-0.5| \geq 0.25 \cup |y-0.5| \geq 0.25 \\ -32(x-0.5), & otherwise \end{cases} \\ &u = \begin{cases} 0, & if\, |x-0.5| \geq 0.25 \quad \cup |y-0.5| \geq 0.25 \\ -16(x-0.25)(y-0.75), & otherwise \end{cases} \end{aligned} \tag{19}$$

This example has two interior characteristic layers in the convection direction. Its location is between (0.25, 0.25) and (0.25, 0.75). For this example, the solution of both the std. τ and SPDE-Q-Net has minimal oscillations. For all the 4 testing example we have computed the computational time taken by both SPDE-Net and SPDE-Q-Net as shown in the Table 1, wherein we can see that SPDE-Net takes longer time to converge for all the mesh-sizes.

5. Results

Although it is not recommended to compare a classical network with the quantum neural network, as the network design and training are quite different, we compare the

two predictions in terms of different errors, such as L^2, H^1, relative l^2, and L_∞, as shown in Tables 2–5. We demonstrate the prediction for different examples from both simulated SPDE-Q-Net and SPDE-Net. SPDE-Net is implemented as mentioned in [5]. We compare the computational time taken by SPDE-Net and SPDE-Q-Net in Table 1 and find the SPDE-Q-Net to be always faster than SPDE-Net.

Table 5. H^1 Error.

	Examples			
	1	**2**	**3**	**4**
Std. τ	1.30×10^{-3}	6.74×10^{-4}	1.43×10^{-3}	3.28×10^{-4}
Std. τ with $\|\|\nabla u\|\|$	3.65×10^{-3}	1.51×10^{-3}	1.72×10^{-3}	3.69×10^{-4}
SPDE-Net	6.01×10^{-4}	4.90×10^{-4}	1.53×10^{-3}	4.1×10^{-4}
SPDE-Q-Net	5.98×10^{-4}	4.80×10^{-4}	1.23×10^{-3}	3.29×10^{-4}

6. Conclusions

In this study, we have explored the application of Quantum Neural Networks (QNNs) in predicting stabilization parameters for solving Singularly Perturbed Partial Differential Equation (SPDEs) using the Streamline-Upwind Petrov-Galerkin (SUPG) technique. The SUPG technique is commonly used in fluid flow simulations to stabilize numerical solutions and improve accuracy. By leveraging the capabilities of QNNs, we aim to enhance the efficiency and accuracy of the SUPG method.

To effectively employ QNNs, we utilize the parameter shift rule, a technique that enables the calculation of derivatives up to the second order. This rule plays a crucial role in determining the appropriate stabilization parameters for the SUPG method. By leveraging quantum computing resources, specifically a simulated quantum circuit provided by Qiskit, we carry out all the necessary computations and simulations.

Our experiments are designed as a proof of concept, focusing on a basic set of scenarios. These initial experiments demonstrate the feasibility of using QNNs for predicting stabilization parameters in fluid flow simulations. By successfully calculating derivatives and optimizing the stabilization parameters using QNNs, we have laid the foundation for addressing more complex problems in fluid dynamics.

The application of QNNs in fluid flow simulations has significant potential for tackling larger and more intricate problems. By harnessing the power of quantum computing and the capabilities of QNNs, we can potentially achieve improved accuracy and computational efficiency in solving SPDEs with the SUPG technique. These preliminary experiments provide valuable insights and open up avenues for future research in the field of quantum-assisted fluid dynamics simulations.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Long, Z.; Lu, Y.; Ma, X.; Dong, B. PDE-Net: Learning PDEs from Data. In Proceedings of the Machine Learning Research, Stockholm, Sweden, 10–15 July 2018; pp. 3208–3216.
2. Raissi, M.; Perdikaris, P.; Karniadakis, G.E. Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations. *arXiv* **2017**, arXiv:1711.10561.
3. Li, Z.; Kovachki, N.; Azizzadenesheli, K.; Liu, B.; Bhattacharya, K.; Stuart, A.; Anandkumar, A. Fourier Neural Operator for Parametric Partial Differential Equations. *arXiv* **2020**, arXiv:2010.08895.
4. Li, Z.; Kovachki, N.; Azizzadenesheli, K.; Liu, B.; Bhattacharya, K.; Stuart, A.; Anandkumar, A. Neural Operator: Graph Kernel Network for Partial Differential Equations. *arXiv* **2020**, arXiv:2003.03485.

5. Yadav, S.; Ganesan, S. SPDE-Net: Neural Network-based prediction of the stabilization parameter for SUPG technique. In Proceedings of the 13th Asian Conference on Machine Learning, Virtual, 17–19 November 2021; pp. 268–283. Available online: https://proceedings.mlr.press/v157/yadav21a.html (accessed on 10 June 2023).
6. Schuld, M.; Sweke, R.; Meyer, J.J. Effect of data encoding on the expressive power of variational quantum-machine-learning models. *Phys. Rev. A* **2021**, *103*, 032430. [CrossRef]
7. Biamonte, J.; Wittek, P.; Pancotti, N.; Rebentrost, P.; Wiebe, N.; Lloyd, S. Quantum machine learning. *Nature* **2017**, *549*, 195–202. [CrossRef] [PubMed]
8. Aïmeur, E.; Brassard, G.; Gambs, S. Machine Learning in a Quantum World. In Proceedings of the Advances in Artificial Intelligence; Quebec, QC, Canada, 7–9 June 2006; Lamontagne, L., Marchand, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2006; pp. 431–442.
9. Carleo, G.; Troyer, M. Solving the quantum many-body problem with artificial neural networks. *Science* **2017**, *355*, 602–606.
10. Tiersch, M.; Ganahl, E.; Briegel, H. Adaptive quantum computation in changing environments using projective simulation. *Sci. Rep.* **2014**, *5*, 12874. [CrossRef] [PubMed]
11. Lovett, N.B.; Crosnier, C.; Perarnau-Llobet, M.; Sanders, B.C. Differential Evolution for Many-Particle Adaptive Quantum Metrology. *Phys. Rev. Lett.* **2013**, *110*, 220501. [CrossRef] [PubMed]
12. Dunjko, V.; Briegel, H.J. Machine learning & artificial intelligence in the quantum domain: a review of recent progress. *Rep. Prog. Phys.* **2018**, *81*, 074001. [CrossRef]
13. Alvarez-Rodriguez, U.; Lamata, L.; Escandell-Montero, P.; Martín-Guerrero, J. Supervised Quantum Learning without Measurements. *Sci. Rep.* **2017**, *7*, 13645. [CrossRef] [PubMed]
14. Wan, K.H.; Dahlsten, O.C.O.; Kristjánsson, H.; Gardner, R.; Kim, M.S. Quantum generalisation of feedforward neural networks. *npj Quantum Inf.* **2016**, *3*, 36. [CrossRef]
15. Ricks, B.; Ventura, D. Training a Quantum Neural Network. In Proceedings of the Advances in Neural Information Processing Systems, Whistler, BC, Canada, 9–11 December, 2003; Thrun, S., Saul, L., Schölkopf, B., Eds.; MIT Press: Cambridge, MA, USA, 2003; Volume 16.
16. Hughes, T.J.R.; Franca, L.P.; Hulbert, G.M. A new finite element formulation for computational fluid dynamics: VIII. The galerkin/least-squares method for advective-diffusive equations. *Comput. Methods Appl. Mech. Eng.* **1989**, *73*, 173–189. [CrossRef]
17. Yadav, S.; Ganesan, S. How Deep Learning performs with Singularly Perturbed Problems? In Proceedings of the 2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE), Sardinia, Italy, 3–5 June 2019; pp. 293–297. [CrossRef]
18. Yadav, S.; Ganesan, S. AI-augmented stabilized finite element method. *arXiv* **2022**, arXiv:2211.13418.
19. Yadav, S.; Ganesan, S. SPDE-ConvNet: Predict Stabilization Parameter for Singularly Perturbed Partial Differential Equation. ECCOMAS Congress 2022. Available online: https://www.scipedia.com/public/Yadav_Ganesan_2022a (accessed on 10 June 2023).
20. Khodayi-Mehr, R.; Zavlanos, M. VarNet: Variational Neural Networks for the Solution of Partial Differential Equations. In Proceedings of the 2nd Conference on Learning for Dynamics and Control, Berkeley, CA, USA, 11–12 June 2020; Proceedings of Machine Learning Research: PMLR; 2020; Volume 120, pp. 298–307. Available online: http://proceedings.mlr.press/v120/khodayi-mehr20a/khodayi-mehr20a.pdf (accessed on 10 June 2023).