Artificial Neural Network-augmented stabilized finite element method

Sangeeta Yadav*, Sashikumaar Ganesan

[a]*Department of Computational and Data Sciences, Indian Institute of Science, Bengaluru, 560012, Karnataka, India*

Abstract

An artificial neural network-augmented Streamline Upwind/Petrov-Galerkin finite element scheme (SPDE-NetII)† is proposed for solving singularly perturbed partial differential equations. In particular, an artificial neural network framework is proposed to predict optimal values for the stabilization parameter to be used in Streamline upwind/Petrov-Galerkin stabilization schemes. The neural network is trained by minimizing a physics-informed cost function, where the equation's mesh and physical parameters are used as input features. Further, the predicted stabilization parameter is normalized with the gradient of the solution to treat the boundary/interior layer region adequately. The proposed approach suppresses the undershoots and overshoots in the stabilized finite element solution and outperforms the existing neural network-based partial differential equation solvers such as Physics-Informed Neural Networks and Variational Neural Networks.

Keywords:
Singularly Perturbed Partial Differential Equations, Streamline upwind/Petrov-Galerkin, Finite Element Method, Artificial Neural Network

1. Introduction

The convection-diffusion partial differential equation (PDE) is used in numerous applications to model phenomena involving the transport of particles, energy, and species concentration. When convection dominates over

*Corresponding author
†For Reviewer 1,
For Reviewer 1 and 2 both

diffusion, this PDE becomes a Singularly Perturbed Partial Differential Equation (SPPDE). Finding an accurate numerical solution for SPPDEs is challenging due to the presence of interior and boundary layers. To mitigate spurious oscillations in the numerical solution of SPPDEs, stabilization techniques are often used with standard numerical methods like finite difference, finite volume, and finite element methods (FEM). These techniques include slope limiting, flux limiting, artificial dissipation, etc. [1, 2]. In nearly all finite element stabilization methods, the addition of artificial diffusion is a crucial idea. Among them, the Streamline/upwind Petrov-Galerkin (SUPG) stabilization method is the most widely employed for solving SPPDEs [3, 4, 5, 6, 7, 8, 9, 10, 11]. While SUPG suppresses spurious oscillations, it requires a well-tuned stabilization parameter to mitigate undershoots and overshoots in the numerical solution. However, determining the optimal value of the stabilization parameter is highly challenging [12, 13], and heuristic approaches are often employed.

In recent literature, several neural network-based solvers have been proposed as alternatives to mesh-based numerical methods for solving PDEs. One such technique is Physics-Informed Neural Networks (PINNs) introduced by Raissi et al. (2019) [14]. PINNs employ artificial neural networks (ANNs) to estimate the pointwise solution by minimizing the residual of the given differential equation. These neural network methods typically follow an unsupervised or semi-supervised approach, where labeled data is not required for training the neural network. Instead, the cost function of the neural network is defined as the residual of the governing PDE, which is implicitly minimized. Another variant of this approach, known as variational-PINNs (VPINNs), has been proposed by Kharazmi and Montazeri (2019) [15]. VPINNs utilize the residual of the variational form of the PDEs as the cost function. A similar technique called VarNet [16] incorporates a Reduced-Order Model (ROM) for efficient computations.

Recently, Kharazmi and Montazeri (2020) [17] introduced a novel framework called hp-variational PINNs (hp-VPINNs), which allows for hp-refinement of the approximate solution. Despite the robustness and promising performance of neural network-based solvers, it has been observed that mesh-based methods remain efficient and often achieve better accuracy with finer mesh resolutions. Furthermore, network-based solvers exhibit limited performance on Singularly Perturbed Partial Differential Equations (SPPDEs), requiring additional stabilization techniques. These observations motivate the present study, where we propose augmenting the Streamline upwind/Petrov-Galerkin

Finite Element Method (SUPG-FEM) with neural networks to obtain an optimal stabilization parameter.

Several ANN-augmented stabilization schemes have been proposed in the literature. Schwander et al. [18] proposed stabilization of Fourier Spectral Methods (FSMs) using artificial dissipation. The authors utilize ANNs to estimate the regularity of the local solution, identifying regions where artificial dissipation can be added effectively. Similarly, Discacciati et al. [19] employed a similar approach to stabilize Runge-Kutta discontinuous Galerkin (RKDG) methods. Veiga and Abgrall [20] proposed a method for parameter-free stabilization of FEM using ANNs. Ray and Hesthaven [21, 22] utilized multilayer perceptrons to identify troubled cells and perform slope limiting for high-order FEM, effectively controlling spurious oscillations in the solution.

For a detailed review of ANN-augmented methods in computational fluid dynamics, we refer the reader to Lye et al. [23]. Notably, most of the existing ANN-augmented stabilization methods primarily employ supervised learning, requiring a significant amount of labelled data to train the ANN. In our previous study on ANN-augmented stabilization [24], we employed an L^2-error minimization approach to train an ANN called SPDE-Net. This approach involved minimizing the error in the numerical solution, which necessitated the availability of the analytical solution.

A similar L^2-error minimization approach was used by Tomasso et al. [25] for solving two-dimensional SPPDEs. Similar schemes have also been developed for various numerical methods, including Fourier Spectral and discontinuous Galerkin. For instance, Lukas et al. [26] proposed a local ANN to estimate the local solution regularity, effectively controlling oscillations in Fourier Spectral methods using nonlinear artificial viscosity. In another work, Jian et al. [27] demonstrated the potential of ANNs for shock capturing in discontinuous Galerkin methods. Their proposed ANN model mapped the element-wise solution to a smoothness indicator for determining the artificial viscosity.

In this paper, we propose an ANN-augmented stabilization strategy for SUPG-FEM. In particular, we strive to train the ANN using the residual of the differential equation, eliminating the need for an analytical solution. The proposed approach follows unsupervised neural network training, unlike the existing ANN-augmented stabilized methods. The main contributions of this study are summarized below.

- A hybrid approach combining the strengths of the SUPG-FEM and

ANN, where an ANN predicts an optimal value of the stabilization parameter and is consequently used in the SUPG-FEM framework to solve SPPDEs.

- A physics-informed cost function based on *a posteriori* error estimator [28] to train the neural network. Unlike the existing approaches in the literature, this cost function does not require the analytical solution of the differential equation.

- The gradient normalization is applied to the predicted global stabilization parameter ($\hat{\tau_K}$) to incorporate the dynamics from the interior/boundary layers regions.

2. Preliminaries

Let $\Omega \subset \mathbb{R}^2$ be a bounded domain with a polygonal Lipschitz-continuous Dirichlet boundary denoted by $\Gamma_{\mathrm{D}} := \partial\Omega$. We adopt standard notations, such as $\mathrm{L}^{\mathrm{p}}(\Omega)$ and $\mathrm{W}^{\mathrm{k,p}}(\Omega)$, where $1 \leq \mathrm{p} < \infty$ and $\mathrm{k} \geq 0$, to represent Lebesgue and Sobolev spaces, respectively. Furthermore, we denote the Hilbert space by $\mathrm{H}^{\mathrm{k}}(\Omega)$, which is equivalent to $\mathrm{W}^{\mathrm{k},2}(\Omega)$. The inner product in the $\mathrm{L}^2(\Omega)$ space is represented by $(\cdot,\cdot)$, and $|c|$ signifies the Euclidean norm of $c \in \mathbb{R}$.

2.1. Convection-Diffusion Equation

We consider a two-dimensional convection-diffusion equation

$$\begin{aligned} -\varepsilon\Delta u + \mathbf{b}\cdot\nabla u &= f \text{ in } \Omega, \\ u &= u_b \text{ on } \partial\Omega, \end{aligned} \tag{1}$$

where $\varepsilon > 0$ is the diffusion coefficient, $\mathbf{b} = (b_1, b_2)^{\mathrm{T}}$ is the convective velocity, $f \in \mathrm{L}^2(\Omega)$ is an external source term, u is the unknown scalar solution. Here, $u_b \in \mathrm{H}^{1/2}(\partial\Omega)$ is a known function.

2.2. Weak Formulation

Let $U := \mathrm{H}^1(\Omega)$ denote the solution space. We can derive the weak form of equation (1) by multiplying it by a test function $v \in V := \mathrm{H}^1_0(\Omega)$ and then integrating over Ω. Here,

$$\mathrm{H}^1_0(\Omega) := \{v \in U \text{ and } v = 0 \text{ on } \Gamma_D\}.$$

By subsequently applying integration by parts, we obtain the following weak form:

Find $u \in U$ such that for all $v \in V$

$$a(u, v) = (f, v), \tag{2}$$

where the bilinear form $a(\cdot, \cdot) : U \times V \to \mathbb{R}$ is defined by

$$a(u, v) = \int_\Omega \varepsilon \nabla u \cdot \nabla v \, dx + \int_\Omega \mathbf{b} \cdot \nabla u \; v \, dx \tag{3}$$

$$(f, v) = \int_\Omega f \; v \, dx. \tag{4}$$

Let Ω_h represent an admissible decomposition of Ω into a finite number of cells, and let K denote a single cell within Ω_h. Additionally, consider $U_h \subset U$ and $V_h \subset V$ as finite-dimensional subspaces. Using these finite-dimensional subspaces, the discrete formulation of equation (2) can be expressed as follows:

Find $u_h \in U_h$ such that

$$a_h(u_h, v) := \varepsilon \left(\nabla u_h, \nabla v\right) + \left(\mathbf{b} \cdot \nabla u_h, v\right) = \left(f, v\right), \tag{5}$$

for all $v \in V_h$. Here, the subscript h in $a_h(\cdot, \cdot)$ denotes the integral over Ω_h. The standard Galerkin form (5) is widely recognized for its tendency to generate spurious oscillations in the solution, especially when boundary and internal layers are present. These oscillations can lead to inaccuracies and instabilities in the computed results. To mitigate these issues and ensure a more accurate and stable solution, the Streamline-Upwind Petrov-Galerkin (SUPG) stabilization term is introduced in the discrete formulation (5).

2.3. SUPG stabilization

In the SUPG method, a residual term is introduced into the variational form of the equation in the streamline direction. This additional term helps to improve the accuracy and stability of the solution.

Let us define $R(u)$ as the residual of the equation (1). The residual represents the difference between the equation's left-hand and right-hand

sides and measures the extent to which the equation is satisfied. It can be expressed as:

$$R(u) = -\varepsilon \Delta u_h + \mathbf{b} \cdot \nabla u_h - f. \tag{6}$$

Now, the SUPG weak form reads:

Find $u_h \in U_h$ such that:

$$a_h^{\mathrm{SUPG}}(u_h, v) = (f, v) \quad \forall\ v \in V_h, \tag{7}$$

where

$$\begin{aligned} a_h^{\mathrm{SUPG}}(u_h, v) = \varepsilon(\nabla u_h, \nabla v) + (\mathbf{b} \cdot \nabla u_h, v) \\ + \sum_{K \in \Omega_h} \tau_K (-\varepsilon \Delta u_h + \mathbf{b} \cdot \nabla u_h - f, \mathbf{b} \cdot \nabla v)_K. \end{aligned}$$

Here, the parameter $\tau_K \in \mathbb{R}$ is a non-negative stabilization parameter chosen by the user. This parameter plays a critical role in determining the accuracy of the approximated solution. The choice of τ_K directly influences the behaviour of the numerical method and is crucial for controlling both oscillations and smearing effects.

Selecting an appropriate value for τ_K is essential to balance suppressing oscillations and avoiding excessive numerical diffusion. If τ_K is set to a large value, it can lead to unintended smearing of the solution, blurring important features and reducing the overall accuracy. Conversely, if τ_K is chosen to be too small, it may not effectively dampen oscillations, resulting in spurious fluctuations in the computed solution.

The SUPG scheme faces a significant challenge in determining the optimal global stabilization parameter τ_K for general cases. While a standard expression exists for the one-dimensional scalar convection-diffusion equation, a general expression remains unknown. Therefore, a technique is required to find the optimal value of the stabilization parameter adaptively.

Machine learning, particularly ANNs, has experienced significant advancements in recent years and has been applied across various scientific fields due to their universal approximation capabilities [29]. In this work, we leverage the power of ANNs to approximate the SUPG stabilization parameter.

2.4. Error Metrics

We use the following metrics to calculate numerical errors in the solution obtained with the predicted $\hat{\tau}$. We will use them for comparison against the baselines as explained in section 3.3.

$$\begin{aligned}
L^2\text{-error: } \|e_h\|_0 &= \|u_h - u\|_{L^2(\Omega)} = \left(\int_\Omega (u_h - u)^2 dx\right)^{\frac{1}{2}} \\
\text{Relative } l^2\text{-error: } \|e_h\|_{0,\ell} &= \sum_{i=1}^{N} \frac{\|u_h(x_i) - u(x_i)\|_{0,\ell}}{\|u\|_{0,\ell}}, \qquad x_i \in \Omega_h \\
H^1 \text{ seminorm-error: } |e_h|_1 &= \|\nabla u_h - \nabla u)\|_{L^2(\Omega)} = \left(\int_\Omega (\nabla u_h - \nabla u)^2 dx\right)^{\frac{1}{2}} \\
L^\infty\text{-error: } \|e\|_{L^\infty(\Omega)} &= ess \quad sup\{|u_h - u| : x \in \Omega\}.
\end{aligned} \tag{8}$$

Here, $\hat{\tau}$ is the stabilization parameter predicted by SPDE-NetII, u is the known analytical solution, and u_h is the SUPG solution calculated with $\hat{\tau}$.

3. SPDE-NetII

We introduce a novel framework, SPDE-NetII, which utilizes a neural network-based approach to estimate a near-optimal stabilization parameter.

3.1. Network Architecture

The neural network architecture of the proposed SPDE-NetII framework is depicted in Figure 1. The fully connected network in SPDE-NetII consists of a few hidden layers, with an input layer size of four. We apply a sigmoid non-linearity for the output layer and use a tanh activation function after each hidden layer. The selection of hyperparameters and the number of hidden layers will be discussed in subsequent sections.

It is important to note that this architecture is designed explicitly for equations with constant coefficients. However, in Section 3.2, we propose an alternative variant of the architecture to handle equations with variable coefficients.

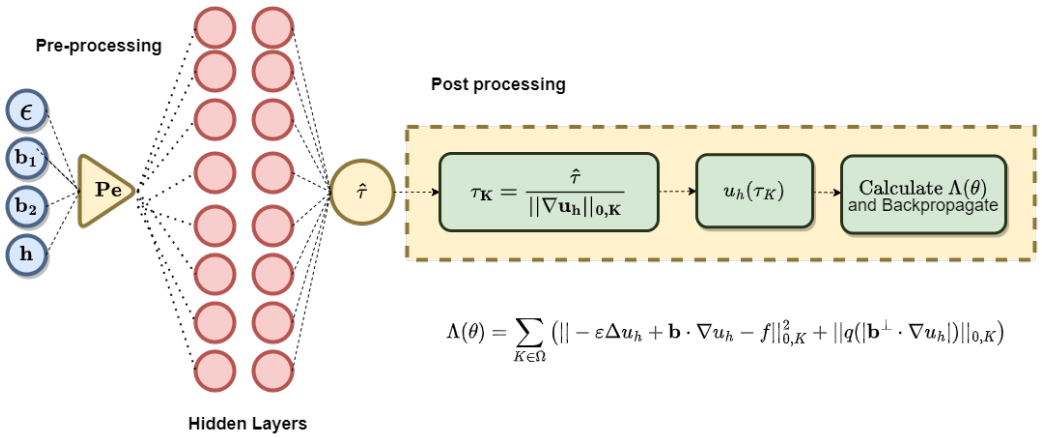


Figure 1: SPDE-NetII$_\theta$: Proposed Network Architecture to compute the stabilization parameter $\hat{\tau}$, where $Pe(\mathrm{I}) = |\mathbf{b}|h/2\varepsilon$ and the θ in the subscript denotes the weights of the network.

The backpropagation algorithm is employed in this study to optimize the network parameters θ and plays a crucial role in training neural networks. To quantify the error in the network's predictions, a specific cost function is utilized, combining a residual-based term and a cross-wind derivative term. This cost function, proposed for finite element simulations in [12, 30], is expressed as follows:

$$\begin{aligned} \Lambda(\theta) &= \sum_{K\in\Omega} \left(\| - \varepsilon \Delta u_h + \mathbf{b} \cdot \nabla u_h - f \|^2_{0,K} + \ \| q \left(|\mathbf{b}^\perp \cdot \nabla u_h| \right) \|_{0,K} \right), \\ \theta^* &= \arg\min_\theta \left(\Lambda(u_h(\theta)) \right). \end{aligned} \tag{9}$$

The objective of the SPDE-NetII$_\theta$ model is to minimize the numerical error (9), which is calculated based on the predicted $\hat{\tau}$. This is achieved by iteratively applying the backpropagation algorithm. In each iteration, the cost function is backpropagated through the network, enabling the update of the network parameters θ. Through iterative adjustment of θ, the network progressively converges to the optimal values that minimize the numerical error (9). During the training process, we employed the Adam optimizer to accelerate gradient descent and the StepLR scheduler to dynamically adjust the learning rate. The StepLR scheduler gradually reduces the learning rate by a gamma factor at each step. For more detailed information on the hyperparameters used in training, please refer to Table 1.

Table 1: Network hyper-parameters

Hyper-parameter	Value
Non-linearity	tanh
Optimizer	Adam
Initial LR	0.0001
Scheduler	LR
Gamma	0.1
Size of the hidden layers	[16, 16]

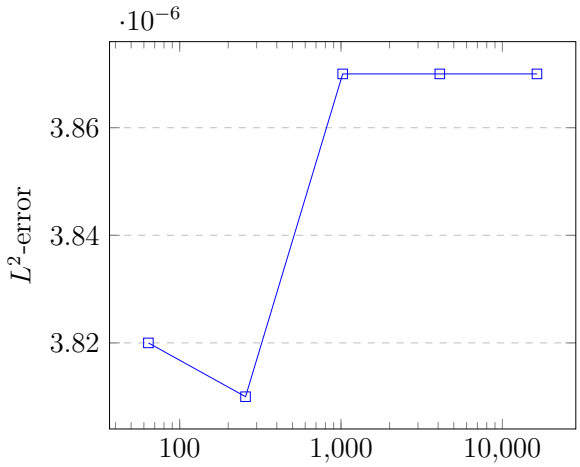


Figure 2: Hyper-parameter search for AI-stab FEM

To determine the optimal network depth and number of neurons per layer, we comprehensively evaluated fully connected neural networks. We assessed the L^2-error in the numerical solution at the final training epoch for different network sizes, as shown in Figure 2. Interestingly, we observed that increasing the network depth did not improve accuracy; instead, it resulted in increased error. Based on these findings, we selected a fully connected network architecture with two hidden layers, each containing 16 neurons. The implementation procedure for the SPDE-NetII framework is outlined in Algorithm 1. To utilize this algorithm, two parameters must be specified: the learning rate η and the number of training epochs n_{epochs}. The input array I is a crucial component of the SPDE-NetII scheme, containing the diffusion coefficient (ε), convection velocity ($\mathbf{b}$), and mesh size (h).

Algorithm 1 SPDE-NetII

Read η_0, n_{epochs}, $\mathrm{I} = \{\varepsilon, b_1, b_2, h\}$

Initialize $\tau_K = 0$ the weights, θ_0, of SPDE-NetII with random values

Initialize the optimizer (Adam in this case) and stepLR scheduler

$$q(s) = \begin{cases} \sqrt{s} & s > 1 \\ 2.5s^2 - 1.5s^3 & \text{otherwise,} \end{cases} \quad \mathbf{b}^{\perp} = \begin{cases} \dfrac{(b_2, -b_1)}{|\mathbf{b}|} & \text{when } |\mathbf{b}| \neq 0 \\ 0 & \text{when } |\mathbf{b}| = 0 \end{cases}$$

Solve equation (5) to get u_h

for $t = 0$ to n_{epochs} **do**

  if Predicting global τ **then**

    $\hat{\tau}(\theta_t) = \text{SPDE-NetII}_{\theta_t}(\mathrm{I})$

    $\tau_K = \dfrac{\hat{\tau}(\theta_t)}{\|\nabla u_h\|_{0,\Omega}}$

  else

    $\hat{\tau}(\theta_t) = \text{SPDE-NetII(local)}_{\theta_t}(\mathrm{I}, \|\nabla u_h\|_{0,\Omega})$

  end if

  $\eta_t = stepLR(t)$

  Solve equation (7) with τ_K to get u_h

  if $\left(\beta_h = \sum_{K\in\Omega} \| - \varepsilon\Delta u_h + \mathbf{b}\cdot\nabla u_h - f\|^2_{0,K}\right) < \beta_{thres}$ **then**

    break

  else

    $\Lambda(\theta_t) = \beta_h + \sum_{K\in\Omega} \|q(|\mathbf{b}^{\perp}\cdot\nabla u_h|)\|_{0,1,K}$

    Backpropogate: $\theta_{t+1} = \theta_t - \eta_t \nabla_{\theta_t}\Lambda(\theta_t)$

  end if

end for

Within Algorithm 1, the Péclet (Pe) number and the Galerkin solution are computed for the given input sample I. However, using a single global value

$\hat{\tau}$ for all cells is inadequate, as layered regions require different stabilization values compared to other cells. To overcome this limitation, we normalize the predicted $\hat{\tau}$ by dividing it by the cell-wise Euclidean norm of the solution's gradient. In Algorithm 1, the variable u_h represents the SUPG solution of equation (7), which is essential in computing the cost function.

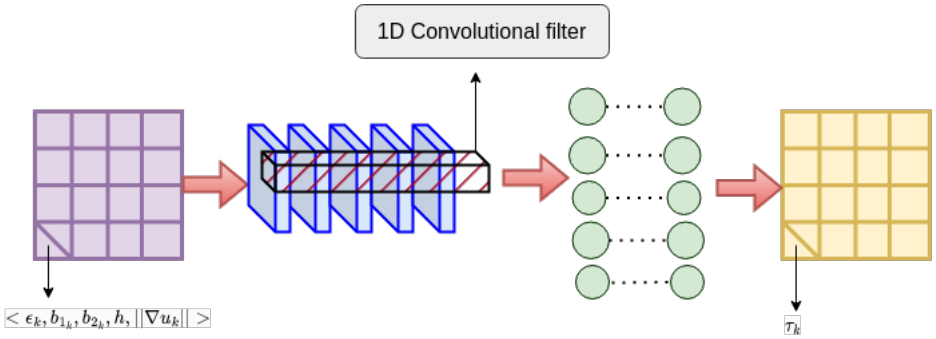


Figure 3: Network architecture of SPDE-NetII (local): applicable for variable coefficient cases of SPPDEs

3.2. SPDE-NetII (local) for problems with variable coefficients

The SPDE-NetII framework's input layer is designed specifically for equations with constant coefficients. However, we propose an alternative architecture called SPDE-NetII (local) to tackle cases with variable coefficients, as depicted in Fig. 3. This architecture employs a shared weight structure across all cells in the domain. We incorporate additional inputs into the network's input to effectively handle equations with variable coefficients. These inputs include the diffusion coefficient (ϵ), convective velocity in the x and y directions ($b_1(x, y)$, $b_2(x, y)$), mesh size (h), and the norm of the gradient of the solution for each cell. By including these inputs, the SPDE-NetII (local) architecture allows more effective handling of variable coefficient cases.

3.3. Baselines

To evaluate the performance of our proposed technique, we compare it against several baseline methods in this section. We consider two PDE solvers based solely on neural networks and two techniques for approximating the stabilization parameter (τ), which can be employed in a mesh-based solver. This allows us to comprehensively and fairly compare our proposed technique, the traditional neural network-based solvers, and the mesh-based stabilized solvers. The baseline techniques we include for comparison are as follows:

1. **Physics-Informed Neural Networks (PINNs):** PINNs are neural networks trained for solving PDEs by minimizing the residual of the given equation [31].
2. **Variational Neural Networks for the Solution of Partial Differential Equations (VarNet):** In this technique, the numerical solution of the equation is approximated by minimizing the residual of the weak form of the equation. For further details, refer to [16].
3. **Standard τ_{std}:** We compute τ_{std} using the following expression, where Pe is the local P'eclet number as given in Algorithm 1:

$$\tau_{std} = \frac{h}{2|\mathbf{b}|}\left(\coth\left(\mathrm{Pe}\right) - \frac{1}{\mathrm{Pe}}\right) \tag{10}$$

   This value of τ_{std} is used to solve the SUPG weak form (7) of the given equation. Table 9 shows the values of τ_{std} calculated using the standard formula for different testing examples. These values will be used for the baseline comparison.
4. **Standard τ_{std} normalized with $\|\nabla u_h\|_{0,K}$:** In this technique, we normalize τ_{std} (equation (10)) with the cell-wise Euclidean norm of the gradient of the numerical solution:

$$\tau_K = \frac{\tau_{std}}{\|\nabla u_h\|_{0,K}}$$

   As shown in Algorithm 1, the proposed technique involves normalizing $\hat{\tau}$ with $\|\nabla u_h\|_{0,K}$. Therefore, we normalize τ_{std} as well to ensure a fair comparison against the standard technique.

4. Computational Experiments

In this section, we present the experimental evaluation of the performance of SPDE-NetII. We performed a series of experiments using six benchmark examples, each representing a highly convective case with varying levels of complexity. The domain Ω is a unit square, uniformly divided into triangular cells. Here, N_{cell} denotes the number of cells in the horizontal direction. These cells are non-overlapping and have a regular shape. To approximate u_h, we utilize the $P_2(\Omega_h)$ finite element space, while the stabilization parameter (τ_K) is approximated using the piecewise constant $DG_0(\Omega_h)$ space. For a detailed definition of $P_2(\Omega_h)$ finite element, please refer to section 3.3 of [32].

4.1. Example 1:

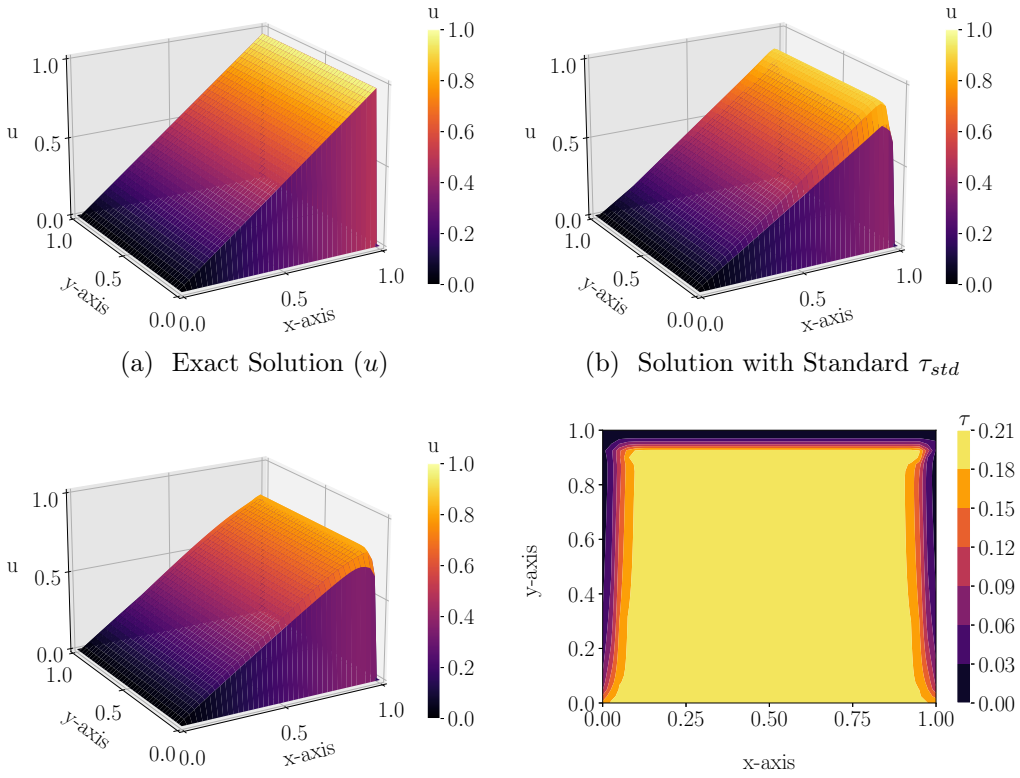
We consider the equation (1) with following data:

$$\varepsilon = 10^{-8}, \quad \mathbf{b} = (1,0)^{\mathrm{T}}, \quad f = 1, \quad \Omega = (0,1)^2, \; u_b = 0. \tag{11}$$

This example is taken from [33]. The exact solution of equation (11) is depicted in Figure 4 and its analytical form is as follows:

$$\begin{gathered} u(x,y) = \alpha x + \frac{(R - L - \alpha)(\exp^{(-\beta(1-x))} - \exp^{-\beta})}{1 - \exp^{-\beta}} + L, \\ \text{where } \alpha = \frac{f}{b_1}, \quad \beta = \frac{b_1}{\epsilon}, \quad L = 0, \quad R = 0. \end{gathered} \tag{12}$$

Readers are referred to the appendix for its derivation. The solution u exhibits distinct layers: an exponential layer at $x = 1$ and two parabolic layers at $y = 0$ and $y = 1$. In the SPDE-NetII, we input $I = [\varepsilon, b_1, b_2, h]$, and the predicted value $\hat{\tau}$ is normalized with $\|\nabla u_h\|_{0,K}$ to obtain τ_K. To solve the discretized form of the equation, we utilize a grid with 40 cells in the horizontal direction, resulting in $h_K = \sqrt{2}/40$. By solving the stabilized SUPG weak form of equation (7) with the τ_K, we obtain the numerical solution. We use analytical u to calculate the error metrics discussed in Section 2.4. The numerical results of SPDE-NetII for Example 1 are presented in Figure 4. In Figure 4(d), we show the values of τ_K over the domain Ω for example 1. Notably, the non-boundary region exhibits a constant predicted value, while τ_K decreases gradually towards the outer boundary. The intersection of the two boundary layers demonstrates the highest value of τ_K. Additionally, Figures 4(b) and 4(c) depict the solution with τ_{std} normalized by $\|\nabla u_h\|_{0,K}$ and the solution obtained using SPDE-NetII, respectively. Overall, SPDE-NetII outperforms all the baseline techniques discussed in Section 3.3 in terms of the error metrics outlined in Section 2.4. Detailed numerical values of the error metrics are provided in Table 5, 6, 7, and 8.



(a) Exact Solution (u)

(b) Solution with Standard τ_{std}

(c) Solution with τ_K obtained from SPDE-NetII

(d) τ_K predicted from SPDE-NetII

Figure 4: Results for Example 1: Presenting the comparison of exact solution with the τ_K obtained from Standard technique and SPDE-NetII, additionally we show the heatmap of the τ_K predicted from the proposed technique.

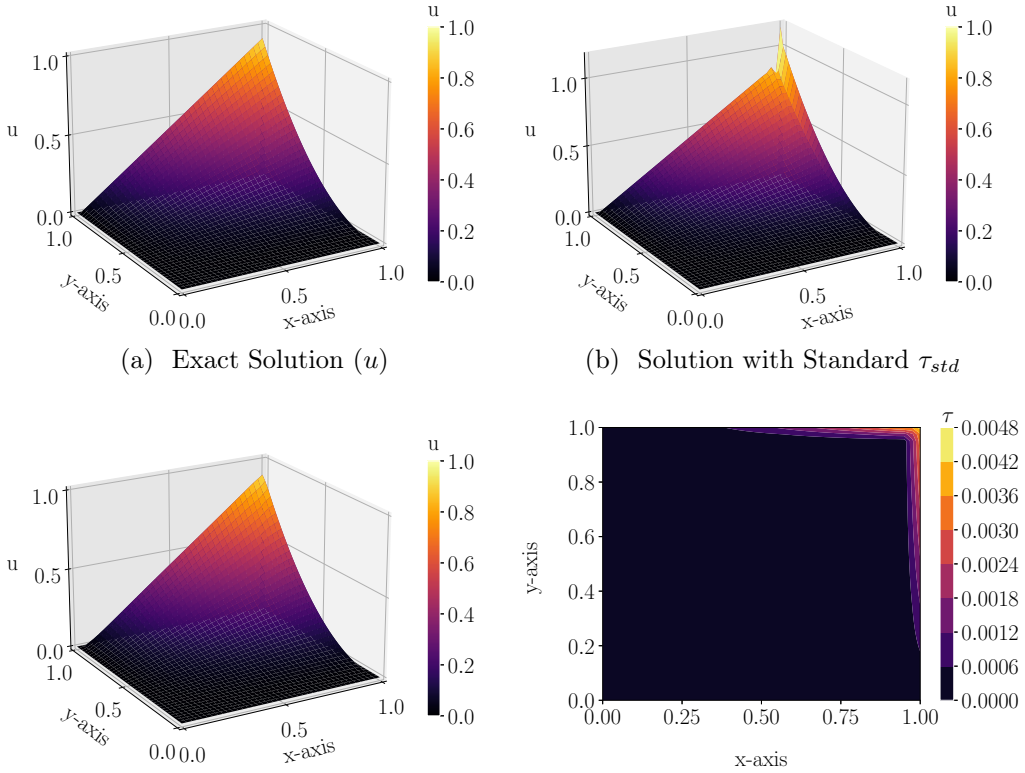
4.2. Example 2:

We consider the convection diffusion equation (1) with following equation coefficients and boundary conditions as given in [34]:

$$\varepsilon = 10^{-8},\ \mathbf{b} = (2,3)^{\mathrm{T}},\ u_b = 0 \tag{13}$$

The source term f is calculated by substituting the following analytical solution (u) in the equation (13).

$$\begin{aligned} u(x,y) =& xy^2 - x\exp\left(\frac{3(y-1)}{\varepsilon}\right) - y^2\exp\left(\frac{2(x-1)}{\varepsilon}\right) \\ &+ \exp\left(\frac{2(x-1)+3(y-1)}{\varepsilon}\right). \end{aligned} \tag{14}$$



(a) Exact Solution (u)

(b) Solution with Standard τ_{std}

(c) Solution with τ_K obtained from SPDE-NetII

(d) τ_K predicted from SPDE-NetII

Figure 5: Results for Example 2: Presenting the comparison of exact solution with the τ_{std} obtained from Standard technique and SPDE-NetII, additionally we show the heatmap of the τ_K predicted from the proposed technique.

The solution exhibits two outflow boundary layers located near $x = 1.0$

and $y = 1.0$, as illustrated in Figure 5(a). This characteristic makes it an appropriate test case for evaluating the performance of the SPDE-NetII method, as discussed in [34]. Figure 5(d) depicts the values of τ_K otained from SPDE-NetII, while Table 9 presents the corresponding τ_{std} value, which is measured to be 0.0177. Notably, all the predicted values are observed to be smaller than τ_{std} as given in Table 9. The heat map of τ_K reveals a distinct pattern corresponding to the presence of boundary layers, with the highest τ_K values occurring around these boundary layer regions. Importantly, SPDE-NetII demonstrates superior performance across all error metrics for this example as shown in Tables 5, 7, 8, 6, mirroring its performance in Example 1.

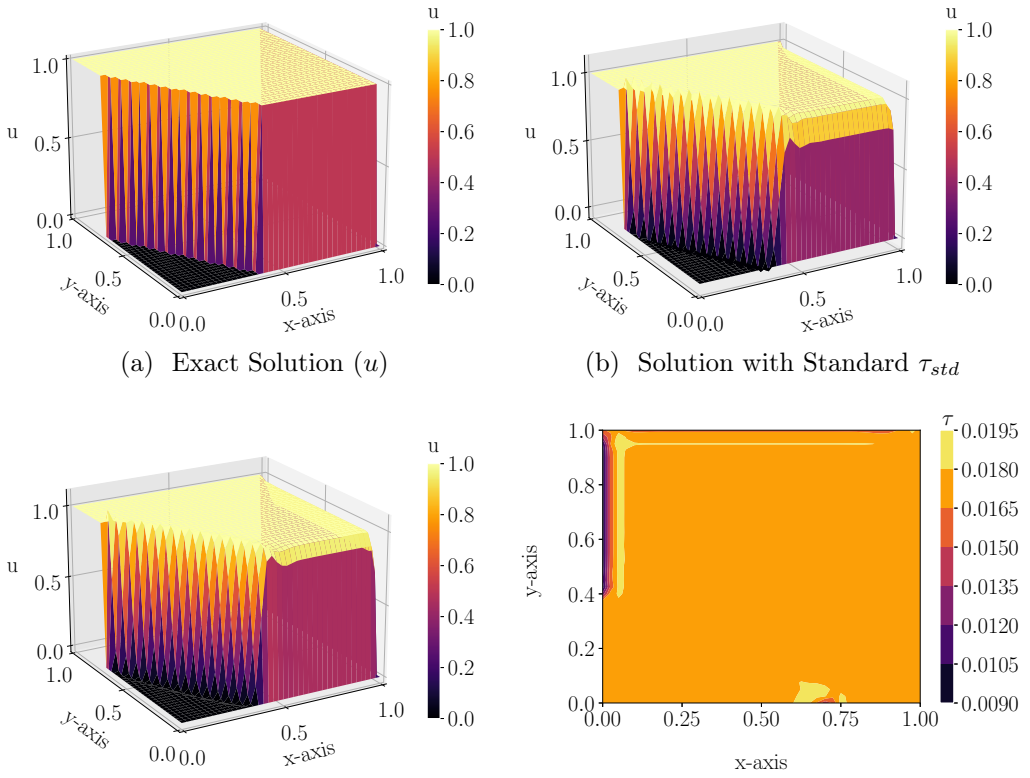
4.3. Example 3:

Next, we consider the convection-diffusion equation (1) with the following equation coefficients and boundary conditions:

$$\begin{aligned} &\varepsilon = 10^{-8},\ \mathbf{b} = (\cos(\theta), \sin(\theta))^{\mathrm{T}},\ \theta = -\pi/3,\ f = 0, \\ &u_b(x,y) = \begin{cases} 0, & \text{for } x = 1 \text{ or } y \leq 0.7 \\ 1, & \text{otherwise.} \end{cases} \end{aligned} \tag{15}$$

The example discussed in this context involves a solution that exhibits exponential and boundary layers, as referenced in [30]. The standard technique yields a τ_{std} value of 0.0049, whereas τ_K remains consistently below 0.002 throughout, as demonstrated in Figure 6. Comparing the two solutions, it is evident that the numerical solution with τ_K exhibits fewer oscillations compared to the standard technique. The heatmap of τ_K in Figure 6(d) illustrates the successful capture of both the interior and exponential layers.

To further evaluate the performance, we compare the solutions obtained using the τ_{std} technique and τ_K. Notably, both solutions display noticeable oscillations near the interior layer. However, upon comparing Figures 6(b) and 6(c), it becomes apparent that SPDE-NetII effectively captures the sharp interior layer in contrast to τ_{std}. It is worth mentioning that among all the baseline techniques described in section 3.3 for this example, SPDE-NetII demonstrates the least errors across all the error metrics detailed in section 2.4.



(a) Exact Solution (u)

(b) Solution with Standard τ_{std}

(c) Solution with τ_K obtained from SPDE-NetII

(d) τ_K predicted from SPDE-NetII

Figure 6: Results for Example 3: Presenting the comparison of exact solution with the τ_{std} obtained from Standard technique and SPDE-NetII, additionally we show the heatmap of the τ_K predicted from the proposed technique.

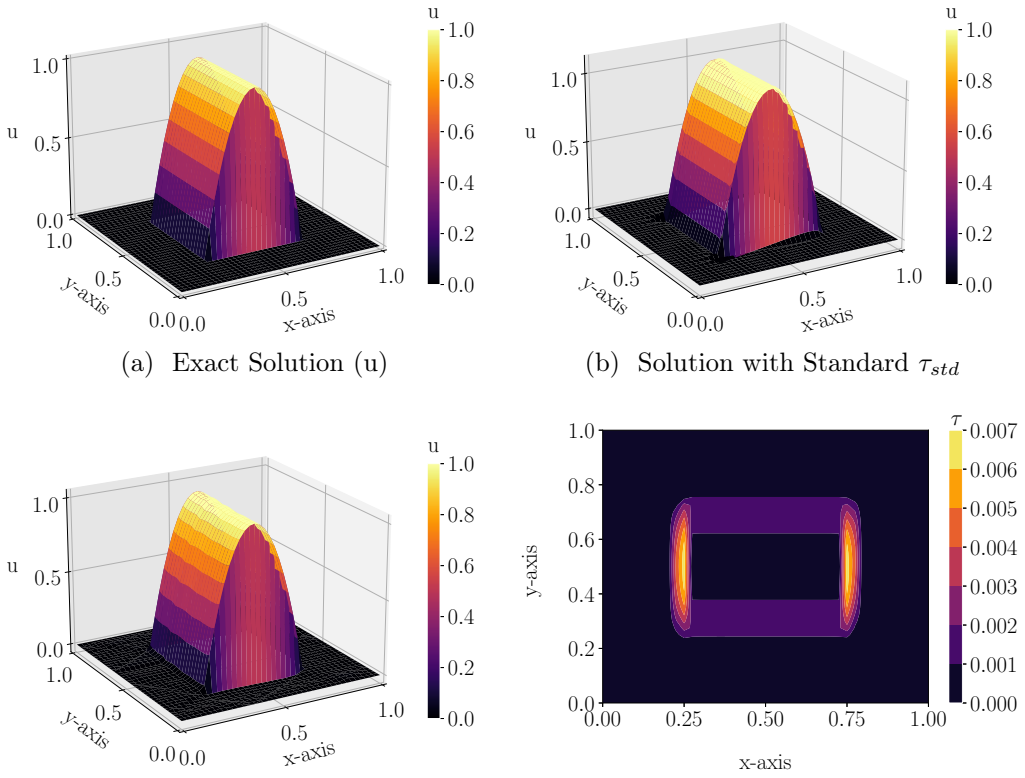
4.4. Example 4:

We consider the convection-diffusion equation (1) with the following equation coefficients and boundary conditions:

$$\varepsilon = 10^{-8},\ \mathbf{b} = (1,0)^{\mathrm{T}},\ u_b = 0$$

$$f = \begin{cases} 0, & \text{if } |x-0.5| \geq 0.25 \cup |y-0.5| \geq 0.25 \\ -32(x-0.5), & \text{otherwise,} \end{cases}$$

$$u = \begin{cases} 0, & \text{if } |x-0.5| \geq 0.25 \cup |y-0.5| \geq 0.25 \\ -16(x-0.25)(y-0.75), & \text{otherwise.} \end{cases} \tag{16}$$



(a) Exact Solution (u)

(b) Solution with Standard τ_{std}

(c) Solution with τ_K obtained from SPDE-NetII

(d) τ_K predicted from SPDE-NetII

Figure 7: Results for Example 4: Presenting the comparison of exact solution with the τ_{std} obtained from Standard technique and SPDE-NetII, additionally we show the heatmap of the τ_K predicted from the proposed technique.

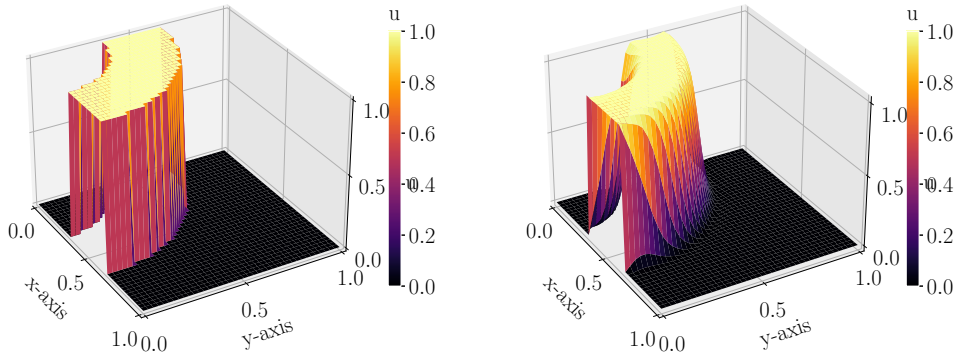
In this example, we introduce a variation compared to example 1 by considering a different source function f, as previously used in [35]. The key characteristic of this example is the presence of two interior characteristic layers in the convection direction, located between the spatial points $(0.25,\ 0.25)$ and $(0.25,\ 0.75)$. The Péclet number associated with this case is $1.77e+06$. To assess the generalizability of SPDE-NetII, we solve this example for various levels of mesh refinement. Figure 7 presents the numerical solution and heat map of τ_K corresponding to a mesh size $h = \sqrt{2}/40$. In the subsequent section, we provide a grid-convergence study. Notably, for this example, the solution obtained from SPDE-NetII exhibits the closest

agreement with the exact solution and displays minimal spurious oscillations. Furthermore, when evaluating error metrics, SPDE-NetII outperforms other techniques. Both SPDE-NetII and Standard τ_{std} effectively capture the interior layers.

4.5. Example 5: Variable convection velocity

In Section 3.2, we introduced SPDE-NetII(local), a method capable of effectively handling SPPDEs with variable coefficients (i.e. coefficients which are a function of the independent variable x and/or y). Unlike the SPDE-NetII, SPDE-NetII(local) considers the local convective velocity ($\mathbf{b}(x, y)$) as an input, allowing for accurate solutions in such cases. To demonstrate the capabilities of SPDE-NetII(local), we applied it to solve equation (1) with variable convection velocity case as given below:

$$\begin{aligned} \epsilon =& 10^{-8}, \quad \mathbf{b} = (-y, x)^{\mathrm{T}}, \quad f = 0 \\ u_b(x, y) =& \begin{cases} 1 & \text{if } \frac{1}{3} \leq x \leq \frac{2}{3} \text{ and } y = 0 \\ 0 & \text{else.} \end{cases} \end{aligned} \tag{17}$$



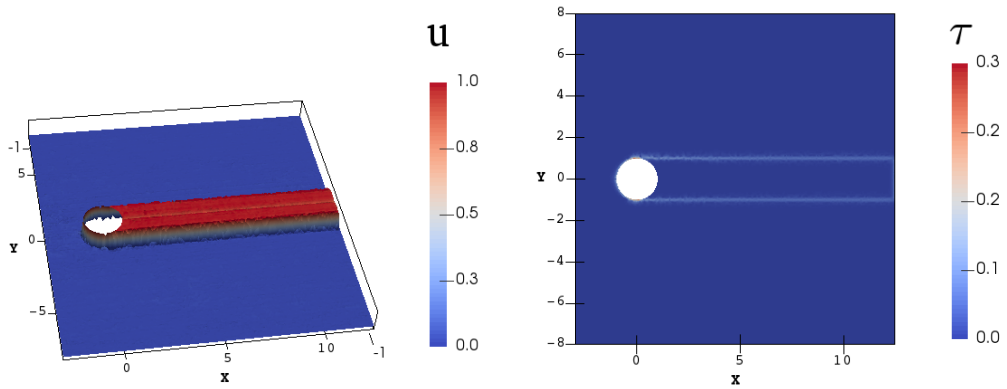
(a) Exact Solution (u)

(b) u_h obtained with $\hat{\tau}$ from SPDE-NetII(local)

Figure 8: SPDE-NetII(local) for Example 5 (equation 17) with variable coefficients.

It is important to note that the standard technique, which is applicable only to constant coefficient cases, could not be utilized for this scenario.

SPDE-NetII (local) fills this gap by accommodating variable coefficients and providing reliable solutions. The obtained solution is presented in Figure 8. The u_h obtained from the τ_K obtained with SPDE-NetII is over-smeared near the interior layer. We couldn't compare the numerical solution of SPDE-NetII with SPDE-NetII (local) as SPDE-NetII is not applicable for variable coefficients



(a) Solution u_h obtained with τ_K predicted from SPDE-NetII (b) τ_K predicted from SPDE-NetII

Figure 9: Results for Example 6: Hemker's Example (Eq. 18): (a) Presenting the solution with the τ_K obtained from SPDE-NetII, (b) additionally we show the heatmap of the τ_K predicted from the proposed technique.

4.6. Example 6: Hemker's Example

Finally, we consider Hemker's example, which has garnered significant attention in the literature due to its direct relevance to real-world applications. Unlike conventional test examples, this equation closely mirrors the situations frequently encountered in practical scenarios. We consider the equation (1) with

$$\begin{aligned} \Omega &= \{(-3, 12) \times (-8, 8)\} \setminus \{(x, y); x^2 + y^2 \leq 1\} \\ \epsilon &= 10^{-8}, \mathbf{b} = (1, 0)^T, \text{and } f = 0. \end{aligned} \tag{18}$$

In this study, we impose homogeneous Dirichlet boundary conditions at the inlet $x = -3$ while setting $u = 1$ on the circle and applying Neumann boundary conditions to the remaining parts of $\partial\Omega$. This example serves as a depiction of heat transfer from a heated column in the presence of convection.

However, when employing this approach, undesired oscillations become apparent at the initial points of the interior layers. These spurious oscillations have a detrimental effect on the accuracy of the results. Acknowledging and addressing these oscillations is crucial to ensure the outcomes' reliability and precision.

Table 2: Perforance comparison for Example 6: Hemker's example (Eq. 18)

	L^2-error	Relative l^2-error	H^1 seminorm-error	L^∞-error
Standard τ_{std}	9.32e-6	1.75e-1	5.97e-4	8.18e-5
SPDE-NetII (local)	5.35e-7	2.57e-2	7.23e-5	7.48e-7

Furthermore, it is important to note that the standard τ_{std} values near the circle are relatively small due to the small cell diameters in that specific mesh region. We have visualized the predicted τ_K from SPDE-NetII for this example in Figure 9b, along with the corresponding numerical solution depicted in Figure 9a. However, it is worth mentioning that the solutions obtained using this approach exhibit significant undesirable oscillations at the initial points of the interior layers. To evaluate the performance of SPDE-NetII (local) compared to τ_{std}, we have conducted a comprehensive analysis and summarized the results in Table 2, considering various metrics defined in Section 2.4. The comparison reveals that SPDE-NetII (local) performs reasonably well in comparison to τ_{std}.

5. Consistency Analysis: Verifying the Numerical Scheme

This section showcases the results of various experiments conducted to assess the consistency of the proposed scheme. Firstly, we present the optimal order of convergence achieved by the proposed scheme. Next, we demonstrate the effectiveness of incorporating an additional crosswind term in the loss function, highlighting its impact on improving the solution. Lastly, we showcase the performance of SPDE-NetII in cases involving higher Pe numbers, demonstrating its efficacy in complex scenarios.

5.1. Mesh Refinement Analysis

Studying grid convergence is essential for ensuring any mesh-based solver's numerical stability and accuracy. The standard τ_{std} scheme, as discussed in

Section 3.3, demonstrates an optimal order of convergence away from layers. However, as we move closer to layers, the associated error for this scheme tends to increase with a decrease in the mesh size (h). On the other hand, it is important to note that all other ANN-based baselines are meshless methods. Consequently, it is not possible to define an order of convergence for these methods in the traditional sense, as they do not rely on a mesh-based discretization. For the SPDE-NetII, we present the numerical errors obtained on uniformly refined mesh, specifically for the case of example 4. By computing the order of error convergence for the L^2-error, we determine the optimal convergence rate for the P_2 finite element used to approximate the solution u. The corresponding results are illustrated in Table 3.

Table 3: Mesh refinement analysis for example 4

N_{cells}	h	L^2-error	Order of convergence
10	1.41e-1	4.75e-4	
20	7.07e-2	2.93e-5	4.02
40	3.54e-2	3.63e-6	3.01
80	1.77e-2	4.51e-7	3.01

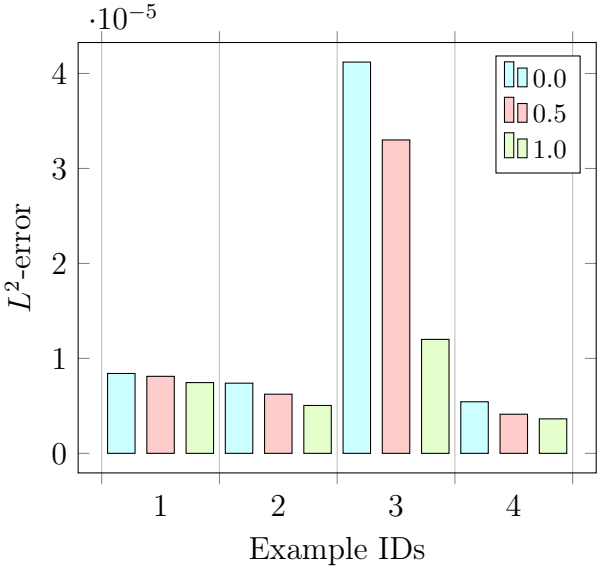
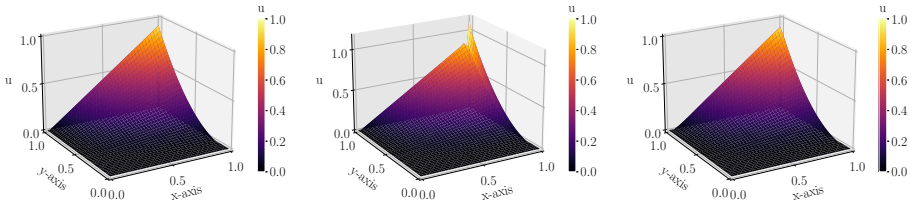


Figure 10: Comparison of L^2-error for different weights of the crosswind-derivative term.

5.2. Impact of crosswind derivative term in the loss function

One major contribution of this manuscript is using the aposteriori error indicator term consisting of the crosswind derivative term in the loss function. This term controls the smearing in the solution. Figure 10 compares the L^2-error in the numerical solution obtained by minimizing the loss function with different weights of the cross-wind derivative term. We clearly saw a drop in L^2-error for all the examples when we used crosswind derivative terms in the loss function. L^2-error corresponding to weight 1.0 is least among the all and hence the same is used for all the experiments in this manuscript.



(a) Exact Solution (u) (b) u_h with τ_{std} (c) u_h with $\hat{\tau}$ from *SPDE-NetII*

Figure 11: Example 2 with $\varepsilon = 10^{-9}$, $\mathbf{b} = (60, 90)$, $|b| = 108.16$.

5.3. Robustness check for higher Peclet number

In this section, we wanted to check the performance of SPDE-NetII for testing examples beyond the range of coefficients considered for training the network. We evaluated the performance of the proposed method, SPDE-NetII, for cases where the value of ϵ is smaller than 10^{-8} and while the absolute value of $|\mathbf{b}|$ exceeds 100. To establish the benchmark, we compared the outcomes obtained from our proposed method with those from the Standard τ_{std} solution.The graphical representation of these results can be observed in Figure 11. In table 4, we can see that SPDE-NetII has lesser error than Standard τ_{std}.

Table 4: Comparison of performance of *SPDE-NetII* with Std. τ for high Pe

	L^2-error	Relative l^2-error	H^1 seminorm-error	L^∞-error
Standard τ_{std}	6.77e-6	1.36e-1	6.74e-4	7.29e-5
SPDE-NetII	5.04e-6	9.73e-2	4.80e-4	4.05e-5

6. Comparative Analysis: Evaluating against Baselines

In this section, we conduct a comprehensive evaluation of SPDE-NetII's performance by comparing it with various baseline methods, considering diverse error metrics. Subsequently, we assess the range of the predicted $\hat{\tau}$ in comparison to the values obtained from τ_{std}.

6.1. Performance

We compared the performance of VarNet, the standard technique, and the method of standard τ_{std} normalized with $\|\nabla u_h\|_0$ with SPDE-NetII for all four benchmark examples in terms of L^2-error, H^1-seminorm, L^∞-error, and relative l^2-error. The performance of PINN is reported in terms of relative l^2 error only since other metrics cannot be computed for PINNs solutions. The tables below display different error metrics for numerical solutions generated by the considered numerical techniques. For all the examples, SPDE-NetII performs better than all the baselines (section 3.3) in terms of various metrics defined in section 2.4.

Table 5: L^2-error ($\|e_h\|_0$)

	Examples			
Techniques	1	2	3	4
Standard τ_{std}	1.32e-5	6.77e-6	1.41e-5	3.63e-6
Standard τ_{std} normalized with $\|\nabla u_h\|_{0,K}$	4.04e-5	1.63e-5	1.76e-5	3.85e-6
VarNet	1.70e-4	2.37e-4	3.00e-4	2.80e-4
SPDE-NetII	7.45e-6	5.04e-6	1.20e-5	3.63e-6

Table 6: Relative l^2-error ($\|e_h\|_{0,\ell}$)

	Examples			
Techniques	1	2	3	4
PINN	4.85e-1	4.85e+1	8.69e-1	1.01e
Standard τ_{std}	1.17e-1	1.36e-1	8.02e-2	4.63e-2
Standard τ_{std} normalized with $\|\nabla u_h\|_{0,K}$	3.35e-1	3.06e-1	9.68e-2	4.90e-2
VarNet	5.17e-1	1.62e	5.39e-1	1.25e
SPDE-NetII	7.41e-2	9.73e-2	6.94e-2	4.60e-2

Table 7: H^1 seminorm-error ($|e_h|_1$)

	Examples			
Techniques	1	2	3	4
Standard τ_{std}	1.30e-3	6.74e-4	1.43e-3	3.28e-4
Standard τ_{std} normalized with $\|\nabla u_h\|_{0,K}$	3.65e-3	1.51e-3	1.72e-3	3.69e-4
VarNet	1.80e-3	1.87e-3	2.26e-3	2.62e-3
SPDE-NetII	7.10e-4	4.80e-4	1.23e-3	3.29e-4

Table 8: L^∞-error ($\|e\|_{L^\infty(\Omega)}$)

	Examples			
Techniques	1	2	3	4
Standard τ_{std}	9.07e-5	7.29e-5	1.13e-4	7.60e-6
Standard τ_{std} normalized with $\|\nabla u_h\|_{0,K}$	1.32e-4	8.70e-5	1.47e-4	1.06e-5
VarNet	3.27e-4	3.55e-4	3.53e-4	3.54e-4
SPDE-NetII	6.91e-5	4.05e-5	9.08e-5	7.35e-6

6.2. Comparison of τ obtained from SPDE-NetII and classical stabilization scheme

In table 9, we present a comprehensive comparison between the values of τ_{std} obtained through the classical stabilization scheme and the values derived using the SPDE-NetII approach for various testing examples. The traditional method yields a single τ_{std} value, whereas, with SPDE-NetII, we observe a range of τ_K values due to its localized definition. By examining the results in table 9, we can observe the distinct differences between the two approaches. The τ_{std} values obtained from the classical stabilization scheme provide a baseline for comparison, representing a global stabilization parameter across all testing examples. On the other hand, the SPDE-NetII approach introduces the concept of τ_K, which varies locally and adapts to the specific characteristics of each example. The range of τ_K values for SPDE-NetII signifies its ability to capture the intricacies and heterogeneity within the dataset. This localized approach allows for finer control and adaptability in stabilizing the system, resulting in more accurate and tailored solutions for individual examples.

Table 9: Value of τ_K by standard technique for different examples.

Example	Internal Layer	Boundary Layer	N_{cells}	τ_{std}	Min. of τ_K	Max of τ_K
1	✗	✓	40	1.77e-2	0.0	2.10e-01
2	✗	✓	40	1.77e-2	0.0	4.80e-03
3	✓	✓	40	1.77e-2	0.0	1.95e-02
4	✓	✗	40	4.90e-3	0.0	7.00e-03

Comparing the two methods, we can observe cases where the τ_K values obtained from SPDE-NetII closely align with the global τ_{std} value derived from the classical stabilization scheme. These instances indicate consistency between the localized and global stabilization approaches, suggesting that the classical scheme captures the dominant stability characteristics of those examples adequately. However, there are also scenarios where the τ_K values significantly deviate from the τ_{std} value. These deviations illustrate the limitations of the classical scheme, which assumes a uniform stability parameter across all instances. In contrast, SPDE-NetII's ability to adapt and adjust τ_K locally provides a more nuanced perspective on the stability requirements, enabling improved accuracy and performance for challenging cases. Overall, the comparison between τ_{std} obtained from the classical stabilization scheme and the range of τ_K values from SPDE-NetII reveals the benefits of a localized approach. While the classical scheme offers a global perspective on stabilization, SPDE-NetII introduces a more flexible and adaptive framework that accounts for the intricacies and variations within the dataset.

7. Summary and Future Work

This paper proposes an ANN-based technique to predict an optimal stabilization parameter for the SUPG method in solving SPPDEs. The technique utilizes equation coefficients and mesh size as input features for the neural network, and the gradient of the solution normalizes the predicted parameter. Using the strong residual form of the equation and considering the cross-wind derivative term, a cost for the back-propagation algorithm is calculated. The proposed technique surpasses existing Neural Network-based PDE solvers like PINNs and VarNet for SPPDE applications and outperforms standard techniques for benchmark problems. Additionally, it achieves optimal convergence when refining the computational grid. Future work can expand on

these findings by exploring time-dependent convection-diffusion and investigating its intricacies and potential applications. Theoretical investigations can be conducted on SPDE-Net, SPDE-NetII, and SPDE-NetII(local) to understand their empirical performance compared to existing methods, contributing to their effectiveness justification. Furthermore, enhancing SPDE-NetII with adaptive mesh refinement in unstable regions can lead to more accurate equations by dynamically refining the mesh during the neural network's training process, resulting in improved results.

Acknowledgements

We would like to express our sincere gratitude to Shell, India for their generous support and funding provided for the completion of this research project.

References

[1] R. J. LeVeque, Numerical Methods for Conservation Laws, Birkhauser-Verlag, 1990.

[2] H. Roos, M. Stynes, L. Tobiska, Numerical methods for singularly perturbed differential equations, Springer-Verlag, 2008.

[3] A. N. Brooks, T. J. R. Hughes, Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations, Computer Methods in Applied Mechanics and Engineering 32 (1982) 199–259.

[4] E. Burman, Consistent supg-method for transient transport problems: Stability and convergence, Computer Methods in Applied Mechanics and Engineering 199 (17-20) (2010) 1114–1123. doi:10.1016/j.cma.2009.11.023.

[5] S. Ganesan, An operator-splitting Galerkin/SUPG finite element method for population balance equations: Stability and convergence, ESAIM Mathematical Modelling and Numerical Analysis 46 (2012) 1447–1465. doi:10.1051/m2an/2012012.

[6] V. John, J. Novo, Error analysis of the supg finite element discretization of evolutionary convection-diffusion-reaction equations, SIAM Journal on Numerical Analysis 49 (3) (2011) 1149–1176. doi:10.1137/100789002.

[7] B. Faranak, N. J. Charles, A semi-discrete SUPG method for contaminant transport in shallow water models, Procedia Computer Science 80 (2016) 1313–1323. doi:10.1016/j.procs.2016.05.476.

[8] D. F. Javier, G.-A. Bosco, J. Volker, N. Julia, An adaptive SUPG method for evolutionary convection-diffusion equations, Computer Methods in Applied Mechanics and Engineering 273 (2014) 219–237. doi:10.1016/j.cma.2014.01.022.

[9] J. Claes, N. Uno, P. Juhani, Finite element methods for linear hyperbolic problems, Computer Methods in Applied Mechanics and Engineering 45 (1-3) (1984) 285–312. doi:10.1016/0045-7825(84)90158-0.

[10] G. Swetlana, I. Traian, J. Volker, W. David, SUPG reduced order models for convection-dominated convection-diffusion-reaction equations, Computer Methods in Applied Mechanics and Engineering 289 (2015) 454–474. doi:10.1016/j.cma.2015.01.020.

[11] L. Richen, W. Qingbiao, Z. Shengfeng, Proper orthogonal decomposition with SUPG-stabilized isogeometric analysis for reduced order modelling of unsteady convection-dominated convection-diffusion-reaction problems, Journal of Computational Physics 387 (18) (2019) 280–302. doi:10.1016/j.jcp.2019.02.051.

[12] P. Knobloch, P. Lukáš, P. Solin, On error indicators for optimizing parameters in stabilized methods, Advances in Computational Mathematics 45 (4) (2019) 1853–1862. doi:10.1007/s10444-019-09662-4.

[13] P. Knobloch, P. Lukáš, P. Solin, Importance of parameter optimization in a nonlinear stabilized method adding a crosswind diffusion, Journal of Computational and Applied Mathematics 393 (2021) 113527. doi:10.1016/j.cam.2021.113527.

[14] 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 378 (2019) 686–707.

[15] E. Kharazmi, Z. Zhang, G. E. Karniadakis, Variational physics-informed neural networks for solving partial differential equations, arXiv preprint arXiv:1912.00873 (2019).

[16] R. Khodayi-Mehr, M. Zavlanos, Varnet: Variational neural networks for the solution of partial differential equations, in: A. M. Bayen, A. Jadbabaie, G. Pappas, P. A. Parrilo, B. Recht, C. Tomlin, M. Zeilinger (Eds.), Proceedings of the 2nd Conference on Learning for Dynamics and Control, Vol. 120 of Proceedings of Machine Learning Research, PMLR, 2020, pp. 298–307.

[17] E. Kharazmi, Z. Zhang, G. E. Karniadakis, hp-vpinns: Variational physics-informed neural networks with domain decomposition, arXiv preprint arXiv:2003.05385 (2020).

[18] S. Lukas, R. Deep, H. J. S., Controlling oscillations in spectral methods by local artificial viscosity governed by neural networks, Journal of Computational Physics 431 (2021) 110144. doi:10.1016/j.jcp.2021.110144.

[19] D. Niccolò, H. J. S., R. Deep, Controlling oscillations in high-order discontinuous Galerkin schemes using artificial viscosity tuned by neural networks, Journal of Computational Physics 409 (2020) 109304. doi:10.1016/j.jcp.2020.109304.

[20] V. M. Han, A. Rémi, Towards a general stabilisation method for conservation laws using a multilayer perceptron neural network: 1D scalar and system of equations, in: Proceedings of the 6th European Conference on Computational Mechanics: Solids, Structures and Coupled Problems, ECCM 2018 and 7th European Conference on Computational Fluid Dynamics, ECFD 2018, 2020, pp. 2525–2539.

[21] R. Deep, H. J. S., An artificial neural network as a troubled-cell indicator, Journal of Computational Physics 367 (2018) 166–191. doi:10.1016/j.jcp.2018.04.029.

[22] R. Deep, H. J. S., Detecting troubled-cells on two-dimensional unstructured grids using a neural network, Journal of Computational Physics 397 (2019) 108845. doi:10.1016/j.jcp.2019.07.043.

[23] L. K. O., M. Siddhartha, R. Deep, Deep learning observables in computational fluid dynamics, Journal of Computational Physics 410 (2020) 109339. arXiv:1903.03040, doi:10.1016/j.jcp.2020.109339.

[24] S. Yadav, S. Ganesan, Spde-net: Neural network based prediction of stabilization parameter for supg technique, in: 13th Asian Conference

on Machine Learning, no. Proceedings of Machine Learning Research, 2021, pp. 268–283.

[25] T. Tassi, A. Zingaro, L. Dede', A machine learning approach to enhance the SUPG stabilization method for advection-dominated differential problems, Mathematics in Engineering 5 (2) (2022) 1–26. doi:10.3934/mine.2023032.

[26] L. Schwander, J. Hesthaven, D. Ray, Controlling oscillations in spectral methods by local artificial viscosity governed by neural networks (2020).

[27] J. Yu, J. S. Hesthaven, A data-driven shock capturing approach for discontinuous galekin methods, Computers & Fluids 245 (2022) 105592. doi:https://doi.org/10.1016/j.compfluid.2022.105592.

[28] R. Verfürth, Robust a posteriori error estimates for stationary convection-diffusion equations, SIAM Journal on Numerical Analysis 43 (4) (2005) 1766–1782. arXiv:https://doi.org/10.1137/040604261, doi:10.1137/040604261.

[29] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Networks 2 (5) (1989) 359–366. doi:https://doi.org/10.1016/0893-6080(89)90020-8.

[30] V. John, P. Knobloch, S. B. Savescu, A posteriori optimization of parameters in stabilized methods for convection–diffusion problems – part i, Computer Methods in Applied Mechanics and Engineering 200 (41) (2011) 2916–2929. doi:https://doi.org/10.1016/j.cma.2011.04.016.

[31] M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations, arXiv preprint arXiv:1711.10561 (2017).

[32] S. Ganesan, L. Tobiska, Finite elements: Theory and algorithms, Cambridge-IISc press, 2017.

[33] V. John, P. Knobloch, On spurious oscillations at layers diminishing (sold) methods for convection-diffusion equations: Part ia review, Computer Methods in Applied Mechanics and Engineering 196 (2007) 2197–2215. doi:10.1016/j.cma.2006.11.013.

[34] P. Knobloch, On the choice of the supg parameter at outflow boundary layers, Adv. Comput. Math. 31 (2009) 369–389. doi:10.1007/s10444-008-9075-6.

[35] P. Knobloch, On the definition of the supg parameter, Electronic Transactions on Numerical Analysis. Volume 32 (2008) 76–89.

Appendix A. Derivation: Analytical solution of Example 1

Let's say we have a one-dimensional convection-diffusion equation given as follows:

$$-\epsilon u'' + bu' = f \tag{A.1}$$

Here b and f are constants, and the value of u on the boundary is

$$u(0) = L, u(1) = R \tag{A.2}$$

$$\text{Let } u(x) = u_0(x) + u_p(x) \tag{A.3}$$

where u_0 is the solution of the homogeneous equation and u_p is the particular solution. Let's solve the homogeneous equation. Let say $u_0 = Ae^{\lambda x}$ is the homogeneous solution for

$$\epsilon u'' + bu' = 0 \tag{A.4}$$

Substitute the value of $u_0 = Ae^{\lambda x}$ in the equation A.1, and we get

$$\begin{aligned}
-A\epsilon\lambda^2 + Ab\lambda =& 0, \text{this equation has two roots } \lambda_1 = 0, \lambda_2 = \frac{b}{\epsilon} \\
=> u_0(x) =& A_1 + A_2 e^{\beta x} \text{ where } \beta = \frac{b}{\epsilon} \\
u(x) =& u_0(x) + u_p(x) = A_1 + A_2 e^{\beta x} + \alpha x \\
u(0) =& L = A_1 + A_2 \\
u(1) =& R = A_1 + A_2 e^{\beta} + \alpha \\
& R = L - A_2 + A_2 e^{\beta} + \alpha \\
=> A_2 =& \frac{R - L - \alpha}{e^{\beta} - 1} \\
u(x) =& L - A_2 + A_2 e^{\beta x} + \alpha x \\
=& L + A_2(e^{\beta x} - 1) + \alpha x \\
=& L + (R - L - \alpha)(\frac{e^{\beta x} - 1}{e^{\beta} - 1}) + \alpha x \\
=& L + (R - L - \alpha)\frac{\frac{e^{\beta x}}{e^{\beta}} - \frac{1}{e^{\beta}}}{1 - e^{-\beta}} + \alpha x \\
u(x) =& \alpha x + \frac{(R - L - \alpha)[e^{(-\beta(1-x))} - e^{-\beta}]}{1 - e^{-\beta}} + L
\end{aligned}$$