Convolutional Neural network for local stabilization parameter prediction for Singularly Perturbed PDEs

Sangeeta Yadav [1] **Sashikumaar Ganesan** [1]

Abstract

Singularly Perturbed Partial Differential Equations are challenging to solve with conventional numerical techniques such as Finite Element Methods due to the presence of boundary and interior layers. Often the standard numerical solution has spurious oscillations in the vicinity of these layers. Stabilization techniques are employed to eliminate these spurious oscillations in the numerical solution. The accuracy of the stabilization technique depends on a user-chosen stabilization parameter, where an optimal value is challenging to find. In this work, we focus on predicting an optimal value of the stabilization parameter for a stabilization technique called the Streamline Upwind Petrov Galerkin technique for solving singularly perturbed partial differential equations. This paper proposes *SPDE-ConvNet*, a convolutional neural network for predicting stabilization parameters by minimizing a loss based on the cross-wind derivative term. The proposed technique is compared with the state-of-the-art variational form-based neural network schemes.

1. Introduction

1.1. Singularly Perturbed Partial Differential Equations

We use Singularly Perturbed Partial Differential Equations (SPPDEs) to model many scientific phenomena (Tobiska & Verfurth, 1996) such as chromatography, fluid flow and continuity of electrons in semiconductors. The significant characteristic of these equations is the diffusion parameter $\epsilon > 0$ multiplied by the second-order derivative term. Solving these equations with finite element or finite volume method is challenging as it will possess boundary and interior layers, and often we get spurious oscillations in the numerical solution. Thus stabilization techniques are proposed to eliminate these spurious oscillations in the numerical solution (Yadav & Ganesan, 2021). The accuracy of these stabilization techniques depends on a user-chosen stabilization parameter (τ). Finding τ is challenging since any generalized expression for τ doesn't exist. This work attempts to solve SPPDEs with the neural network by leveraging its function approximation capabilities by predicting τ.

Recently, deep learning has made its way into scientific computing (Ee & Yu, 2017). Its universal approximation capabilities (Cybenkot, 2006) are utilized for designing alternate methods of solving partial differential equations (PDE). Unlike image classification or segmentation, PDE solving is an unsupervised task as we do not have access to the analytical solution of the given PDE at the training time. Thus implicit information from the equation itself is used to solve the equation. One such method is Physics Informed Neural Networks (PINN) which minimizes the residual of the equation (Raissi et al., 2017). PINNs show limited accuracy for solving SPPDEs; thus, we attempt to aid conventional FEM with deep learning-based τ prediction.

1.2. Outline

The paper is organized as follows: In section 1, we introduce a SPPDE and the challenges associated with solving them. In section 2, a brief description of a few contemporary deep learning-based techniques for solving PDEs is given, followed by stabilization techniques such as SUPG. Section 3 provides the mathematical preliminaries required to understand the proposed technique. In section 4, we provide details of the proposed method with network architecture and the loss function. We discuss two major neural network-based PDE solvers in the following section. Finally, the work is concluded in section 5.

2. Literature Review

SPPDEs are a particular class of PDEs; therefore, any technique developed for general PDEs can be modified for SPPDEs. Keeping this in mind, we shall first discuss the avail-

[*]Equal contribution [1]Department of Computational and Data Science, Indian Institute of Science, Bangalore, India. Correspondence to: Sangeeta Yadav <sangeetay@iisc.ac.in>.

Accepted after peer-review at the 1st workshop on Synergy of Scientific and Machine Learning Modeling, SynS & ML ICML, Honolulu, Hawaii, USA. July, 2023.

able deep learning solvers for PDEs and how we can modify these solvers for SPPDEs. Much research has been done on solving PDEs with deep learning in recent years (Beguinet et al., 2022). Deep learning can be used in two ways, either as a direct PDE solver or as a helper for conventional PDE solving techniques such as FEM, FDM, and FV. While using DL as a PDE solver, the given problem is modelled as an optimization problem since the labelled data is unavailable for supervised training.

2.1. Physics-Informed Neural Network (PINN)

PINN is a standard neural network-based PDE solving technique. In this subsection, we elaborate on the methodology of PINN (Raissi et al., 2017). Unlike neural network-based supervised learning, where we have access to the ground truth, PINN uses a data-driven approach that considers the physical laws of the data for computing loss function for training the neural network. The PINN approximate the numerical solution by minimizing the residual of the equation constrained with boundary conditions. For example, let us say we are given the following PDE defined on a bounded domain $\Omega \in R^d$ for $d = 1, 2$ with boundary conditions as given below:

$$\begin{aligned} \mathcal{D}(u(x)) &= 0 \quad x \in \Omega \\ \mathcal{B}(u(x)) &= 0 \quad x \in \partial\Omega \end{aligned} \tag{1}$$

where u is the unknown solution, $\partial\Omega$ is the domain boundary, $\mathcal{D}$ denotes a linear or nonlinear differential operator, and the operator $\mathcal{B}$ denotes the boundary condition of the given PDE(e.g., Dirichlet, Neumann and Robin boundary condition). In PINN, an approximate solution $\hat{u}$ to equation (1) is estimated by a feed-forward network M as follows:

$$\begin{aligned} \hat{u} &:= M(x;\theta) \\ Loss(\theta) &= MSE_u + MSE_f, \\ MSE_u &= \frac{1}{N_u}\sum_{i=1}^{N_u} |\hat{u}^i - u(x_u^i)|^2, \\ MSE_f &= \frac{1}{N_f}\sum_{i=1}^{N_f} |f(x_f^i)|^2 \end{aligned} \tag{2}$$

where x_u^i, x_f^i are the spatially collocated points in Ω and $\partial\Omega$ respectively, N_f, N_u is the number of boundary and interior points, respectively. Loss(2) is minimized to obtain an optimal θ. PINN was the foremost neural network architecture proposed for solving PDEs, but its accuracy was limited; hence, many advancements have been made after PINN. One such advancement is Variational Neural Networks(VarNet)(Khodayi-mehr & Zavlanos, 2020) for the solution of PDEs which will be explained in the next section.

2.2. Variational Neural Networks for the Solution of Partial Differential Equations (VarNet)

VarNet (Khodayi-mehr & Zavlanos, 2020) is a neural network-based PDE solver. Its novel loss function depends on PDE's variational (integral) form rather than its differential form used in PINN. This loss function effectively approximates the solution as it uses lower-order derivatives. The performance of both PINNs and VarNet is limited for SPPDEs, so we attempt to add the classical stabilization technique in the loss function of the neural-network-based PDE solvers to enhance its accuracy for solving SPPDEs. We present the SUPG stabilization technique in the next section.

2.3. Stabilization technique: Streamline Upwind Petrov Galerkin

Many stabilization techniques exist in the literature; one widely used technique is the streamline upwind Petrov–Galerkin(SUPG) technique (Brooks & Hughes, 1982; Hughes et al., 1989). It stabilizes the given weak form of the PDE by adding the extra diffusion in the upwind direction. In SUPG, the amount of stabilization is controlled by the value of a user-chosen stabilization parameter (τ). A significant value of τ can smear the oscillations, whereas a small value will not remove the oscillation adequately. Thus finding an optimal value of τ is essential for good performance with the SUPG technique. In this paper, we propose to predict the value of τ with a convolutional neural network by minimizing an error functional proposed in (John et al., 2011). It will enhance its accuracy for SPPDEs.

2.4. Contributions

The major contributions are:

- A convolutional neural network is proposed for predicting stabilization parameters for SUPG for two-dimensional SPPDEs.
- We effectively use an error functional proposed in (John et al., 2011) based on the cross-wind derivative term as the loss function for the proposed *SPDE-ConvNet*.
- The proposed technique is compared with the following contemporary ideas:
  - VarNet (Kharazmi et al., 2019)
  - SUPG Stabilized Finite Element Method (N. & R., 1982)

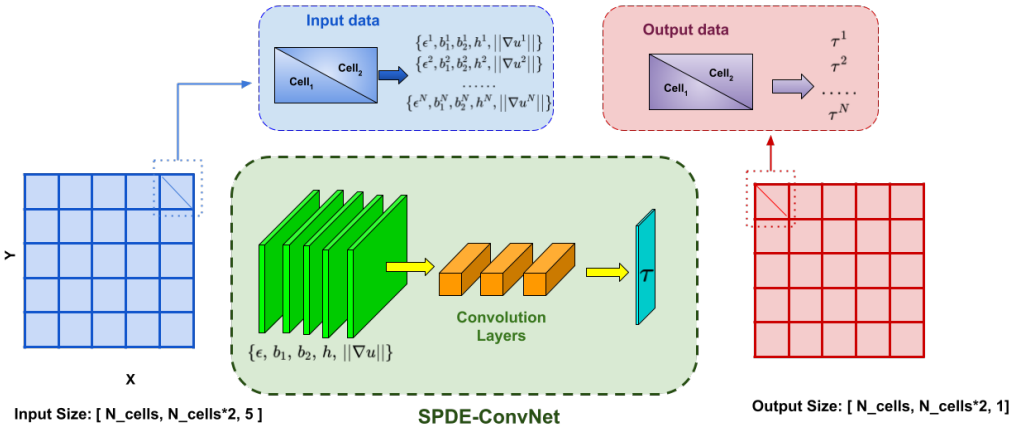

Figure 1. Network Architecture of *SPDE-ConvNet*

3. Mathematical Preliminaries

3.1. Singularly Perturbed Partial Differential Equation (SPPDE)

SPPDEs are a class of differential equations with a small diffusion parameter ($\epsilon > 0$) multiplied with the second order differential term. A small value of ϵ often induces spurious oscillations in the standard Galerkin solution. For a given bounded domain $\Omega \subset \mathbb{R}^d$, where $d \in \mathbb{N}$, an SPPDE is given as follows:

$$\begin{aligned} -\epsilon\Delta u + \mathbf{b}.\nabla u = f(x), \quad & x \in \Omega \subset \mathbb{R}^d, \\ u = g, \quad & on \partial\Omega \end{aligned} \tag{3}$$

$\epsilon > 0$ is the diffusion coefficient and is called perturbation parameter, $\mathbf{b} = [b_1, b_2]^T$ is the convective velocity, $f \in L^2(\Omega)$ is the external source term, u is the unknown scalar term, g is the Dirichlet boundary value. For smooth $\mathbf{b}$ and $f(x)$, equation (3) has a unique solution. We will consider the convection-dominated problems where $\epsilon << |\mathbf{b}|$.

3.2. Weak Formulation

In this work, we use FEM, and the first step in FEM is to derive the weak form for the given equation. The weak form equation of (3) is derived by multiplying it by a function $v \in V := H_0^1(\Omega)$, followed by integrating on Ω and subsequently applying integration by parts. Now, we find $u \in H^1(\Omega)$ such that for all $v \in V$

$$a(u, v) = (f, v) \tag{4}$$

where, the bilinear form $a(\cdot, \cdot) : H^1(\Omega) \times H_0^1(\Omega) \to \mathbb{R}$ and the linear form $f(v) : H_0^1(\Omega) \to \mathbb{R}$ are defined as:

$$a(u, v) = \int_\Omega \epsilon \nabla u \cdot \nabla v \, dx + \int_\Omega \mathbf{b} \cdot \nabla u v \, dx \tag{5}$$

$$f(v) = \int_\Omega f v \, dx \tag{6}$$

Let Ω_h be an admissible decomposition of Ω and let K represent a single cell in Ω_h. The weak formulation for this discretized domain requires choosing a finite-dimensional space $V_h \subset H_0^1(\Omega)$ comprising continuous piece-wise polynomials and finding $u_h \in H^1(\Omega)$ such that for all $v_h \in V_h$ we have

$$\begin{aligned} &a_h(u_h, v_h) = (f, v_h), \\ &a_h\left(u_h, v_h\right) := \varepsilon\left(\nabla u_h, \nabla v_h\right) + \left(\mathbf{b} \cdot \nabla u_h, v_h\right) = \left(f, v_h\right) \end{aligned} \tag{7}$$

3.3. Stabilized weak formulation of the SPPDE using SUPG

In SUPG, we add a residual term to the weak form in the direction of streamline. Let $R(u)$ be the residual of the equation (3) defined as:

$$R(u) = -\epsilon\Delta u + b \cdot \nabla u - f \tag{8}$$

The term $R(u)$ is added to the discretized weak formulation given in equation (7). Now, the modified discrete weak form is to find $u_h \in V_h$ such that:

$$\begin{aligned} &a\left(u_h, v_h\right) + \left(R_h\left(u_h\right), \tau\mathbf{b} \cdot \nabla v_h\right) = \left(f, v_h\right), \\ &a(u_h, v_h) = \epsilon(\nabla u_h, \nabla v_h) + (\mathbf{b} \cdot \nabla u_h, v_h) \\ &+ \sum_{K \in \Omega_h} \tau_K(-\epsilon\Delta u_h + \mathbf{b} \cdot \nabla u_h - f_h, \mathbf{b} \cdot \nabla v_h)_{\Omega_h} \end{aligned} \tag{9}$$

τ_K is a non-negative stabilization parameter. Its value plays an essential role in the quality of the approximated solution; hence, it is the focus of this work. A very large value can show unexpected smearing, whereas a very small value will not remove the spurious oscillations. We aim to predict τ with a convolutional neural network.

3.4. Standard stabilization parameter

In literature, a few expressions exist for stabilization parameter (τ) given as follows:

$$\begin{aligned} Pe_K &= \frac{bh}{2\epsilon}, \\ \tau_{std}|_K &= \frac{h_K}{2|\mathbf{b}|}(\coth(Pe_K) - \frac{1}{Pe_K}) \end{aligned} \tag{10}$$

where h_K is the diameter of the cell K, Pe_K is the local Peclet number. The numerical accuracy of τ_{std} is inadequate for all the SPPDEs, and this expression is not extendable to higher dimensional problems. Thus we attempt to develop a generalizable τ prediction technique using convolutional neural networks wherein we aim to get a lower numerical error in the solution than what is provided by standard τ.

4. Proposed Method: SPDE-ConvNet

We propose *SPDE-ConvNet*, a convolutional neural network for predicting the value of τ. The loss of the neural network is inspired by the error indicators proposed in (John et al., 2011). It consists of SUPG stabilized weak form(equation (9)) of the equation.

$$\begin{aligned} \tau_K(\theta) &= \textit{SPDE-ConvNet}(\epsilon^K, b_1^K, b_2^K, h^K, ||\nabla u_h^K||) \\ &\quad \text{where} K \in \Omega_h \\ Loss(\tau(\theta)) &= [-\epsilon\Delta u_h + \mathbf{b}\cdot\nabla u_h - f]^2 + q(\mathbf{b}^\perp\cdot\nabla u_h) \\ \text{where, } q(s) &= \begin{cases} \sqrt{s} & s > 1 \\ 2.5s^2 - 1.5s^3 & \text{otherwise} \end{cases} \\ \text{and, } \mathbf{b}^\perp(\mathbf{x}) &= \begin{cases} \frac{[b_2(\mathbf{x}), -b_1(\mathbf{x})]}{|\mathbf{b}(\mathbf{x})|} & \text{if } |\mathbf{b}(\mathbf{x})| \neq 0 \\ 0 & \text{else } |\mathbf{b}(\mathbf{x})| = 0 \end{cases} \\ \theta^* &= \arg\min_\theta (Loss(\tau(\theta))) \end{aligned} \tag{11}$$

u_h is the numerical solution of equation (9) with predicted τ_K, the function $q(s)$ represents the cross-wind derivative term. *SPDE-ConvNet* consists of three one-dimensional convolutional layers of size [64, 32, 32]. Each layer consists of a convolutional filter which strides by 1. The input to the network consists of diffusion coefficient (ϵ), convective velocity in x and y direction (b_1, b_2), mesh size (h) and the norm of the gradient of the Galerkin solution. It is implemented from scratch using *PyTorch* (Paszke et al., 2019) and *FEniCS*(A. Logg, 2012), (Logg & Wells, 2010). The network is shown schematically in figure 1.

4.1. Example :

For testing the accuracy of *SPDE-ConvNet*, we consider the convection-diffusion equation (3) with following equation coefficients and boundary conditions. This example is taken from (Knobloch, 2009):

$$\begin{aligned} &\epsilon = 10^{-8}, \quad \mathbf{b} = (2,3)^T, \quad \Omega = (0,1)^2, \\ &u = 0 \quad \text{on} \quad \partial\Omega \end{aligned} \tag{12}$$

Source term f is calculated by substituting the following analytical solution(u_{exact}) in equation (12).

$$\begin{aligned} u_{exact}(x,y) &= xy^2 - x\exp\left(\frac{3(y-1)}{\epsilon}\right) \\ &- y^2\exp\left(\frac{2(x-1)}{\epsilon}\right) + \exp\left(\frac{2(x-1)+3(y-1)}{\epsilon}\right) \end{aligned} \tag{13}$$

It contains two boundary layers near $x = 1.0$ and $y = 1.0$ as shown in figure 2(a) and hence makes a suitable test case for checking the performance of the *SPDE-ConvNet* as mentioned in (Knobloch, 2009). The $\tau(x)$ predicted from *SPDE-ConvNet* is shown in figure 2.

4.2. Performance

We compare the error metrics given by *SPDE-ConvNet*, VarNet and standard τ_{std}. The results are shown in table 1. All the error metrics for *SPDE-ConvNet* is less than VarNet and standard τ_{std}. It shows it performs better than the other two techniques.

Table 1. Comparison of *SPDE-ConvNet* with other techniques

	L^2 Error	Relative l^2 error	H^1 error	l^∞ error
Standard τ	6.77e-6	1.36e-1	6.74e-4	7.29e-5
VarNet	2.37e-4	1.62e+0	1.87e-3	3.55e-4
SPDE-ConvNet	3.04e-6	8.36e-2	3.20e-4	4.03e-5

5. Summary

We proposed *SPDE-ConvNet*, a convolutional neural network for predicting stabilization parameters for solving two-dimensional SPPDEs with the SUPG technique. The proposed technique is tested in terms of four different error metrics. The comparison shows that the proposed CNN-based *SPDE-ConvNet* technique outperforms both VarNet and the standard τ_{std}.

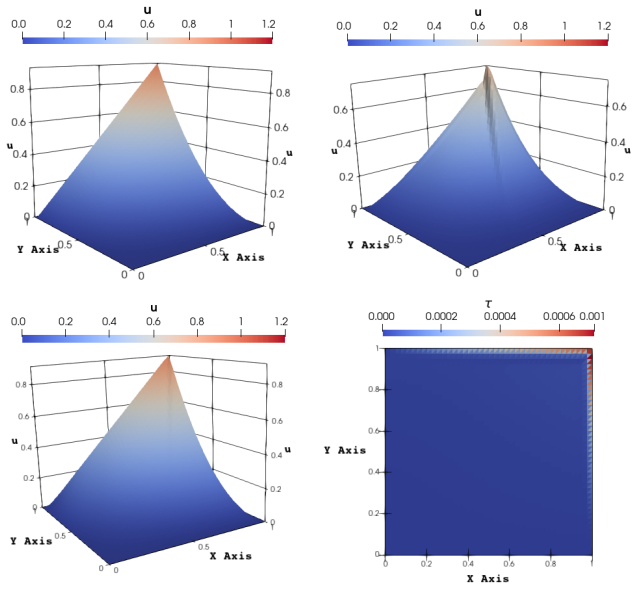

Figure 2. (a) Exact Solution, (b) Solution with standard τ, (c) Solution from *SPDE-ConvNet*

References

A. Logg, K.-A. Mardal, G. N. W. e. a. *Automated Solution of Differential Equations by the Finite Element Method*. Springer, 2012. doi: 10.1007/978-3-642-23099-8.

Beguinet, A., Ehrlacher, V., Flenghi, R., Fuente, M., Mula, O., and Somacal, A. Deep learning-based schemes for singularly perturbed convection-diffusion problems. *arxiv*, 05 2022.

Brooks, A. N. and Hughes, T. J. R. 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(1):199–259, September 1982. ISSN 0045-7825. doi: 10.1016/0045-7825(82)90071-8. URL http://www.sciencedirect.com/science/article/pii/0045782582900718.

Cybenkot, G. Approximation by Superpositions of a Sigmoidal Function *, 2006. URL /paper/Approximation-by-Superpositions-of-a-Sigmoidal-*-Cybenkot/05ceb32839c26c8d2cb38d5529cf7720a68c3fab.

Ee, W. and Yu, B. The deep ritz method: A deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6, 09 2017. doi: 10.1007/s40304-018-0127-z.

Hughes, T. J. R., Franca, L. P., and Hulbert, G. M. A new finite element formulation for computational fluid dynamics: VIII. The galerkin/least-squares method for advective-diffusive equations. *Computer Methods in Applied Mechanics and Engineering*, 73(2):173–189, May 1989. ISSN 0045-7825. doi: 10.1016/0045-7825(89)90111-4. URL http://www.sciencedirect.com/science/article/pii/0045782589901114.

John, V., Knobloch, P., and Savescu, S. B. A posteriori optimization of parameters in stabilized methods for convection–diffusion problems – part i. *Computer Methods in Applied Mechanics and Engineering*, 200(41):2916–2929, 2011. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2011.04.016. URL https://www.sciencedirect.com/science/article/pii/S0045782511001496.

Kharazmi, E., Zhang, Z., and Karniadakis, G. E. Variational physics-informed neural networks for solving partial differential equations. *arXiv preprint arXiv:1912.00873*, 2019.

Khodayi-mehr, R. and Zavlanos, M. M. Varnet: Variational neural networks for the solution of partial differential equations. *ArXiv*, abs/1912.07443, 2020.

Knobloch, P. On the choice of the supg parameter at outflow boundary layers. *Adv. Comput. Math.*, 31:369–389, 10 2009. doi: 10.1007/s10444-008-9075-6.

Logg, A. and Wells, G. N. DOLFIN: automated finite element computing. *ACM Transactions on Mathematical Software*, 37, 2010. doi: 10.1145/1731022.1731030.

N., B. A. and R., H. T. J. 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*, 32:199–259, 1982.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32, pp. 8026–8037. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf.

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

Tobiska, L. and Verfurth, R. Analysis of a Streamline Diffusion Finite Element Method for the Stokes and Navier-Stokes Equations. *SIAM Journal on Numerical Analysis*, 33(1):107–127, 1996. ISSN 0036-1429. URL https://www.jstor.org/stable/2158427.

Yadav, S. and Ganesan, S. Spde-net: Neural network based prediction of stabilization parameter for supg technique. In *13th Asian Conference on Machine Learning*, number Proceedings of Machine Learning Research, pp. 268–283. https://proceedings.mlr.press/v157/yadav21a.html, 2021.

.1. Error Metrics

We use the following metrics to calculate numerical errors in the solution obtained with the τ predicted from *SPDE-ConvNet*. We use them for comparison against the standard τ(equation (10)) and VarNet as explained in section 2.

$$L^2 \text{ error: } ||e_h||_{L^2(\Omega)} = ||\hat{u}(\hat{\tau}) - u_{exact}||_{L^2(\Omega)}$$

$$\text{Relative } l^2 \text{ error: } \sum_{i=1}^{i=N_d} \frac{||\hat{u}_{\hat{\tau}}(x_i) - u_{exact}(x_i)||_2}{||u||_2}$$

$$H^1 \text{ error: } ||e||_{H^1(\Omega)} = = \sum_{\alpha<=1} \int_{\Omega} D^{\alpha}(\hat{u}(\hat{\tau}) - u_{exact}) D^{\alpha}(\hat{u}(\hat{\tau}) - u_{exact}) dx$$

$$L^{\infty} \text{ error: } ||e||_{L^{\infty}(\Omega)} = ess \quad sup\{|\hat{u}(\hat{\tau}) - u_{exact}| : x \in \Omega\} \tag{14}$$

Here N_d is the number of degrees of freedom, D^{α} is the weak derivative up to order α, $\hat{\tau}$ is the stabilization parameter predicted by SPDE-NetII, u_{exact} is the analytical solution, $\hat{u}(\hat{\tau})$ is the SUPG solution calculated with $\hat{\tau}$.