SPDE-Net: Neural Network based prediction of stabilization parameter for SUPG technique

Sangeeta Yadav SANGEETAY@IISC.AC.IN
Department of Computational and Data Sciences, Indian Institute of Science, Bangalore, India

Sashikumaar Ganesan SASHI@IISC.AC.IN
Department of Computational and Data Sciences, Indian Institute of Science, Bangalore, India

Editors: Vineeth N Balasubramanian and Ivor Tsang

Abstract

We propose *SPDE-Net*, an artificial neural network (ANN), to predict the stabilization parameter in the streamline upwind/Petrov-Galerkin (SUPG) stabilization technique for solving singularly perturbed differential equations (SPDEs). The prediction task is modeled as a regression problem and is solved using Artificial Neural Network (ANN). Three training strategies, i.e., supervised, L^2 error minimization (global), and L^2 error minimization (local), for the ANN are proposed. It is observed that the proposed method yields accurate results and even outperform the existing state-of-the-art ANN-based partial differential equation (PDE) solvers such as Physics Informed Neural Network (PINN) for one-dimensional SPDEs.

Keywords: Deep Learning; Singularly Perturbed Partial Differential Equations; SUPG; Finite element methods;

1. Introduction

The species concentration scalar equation is used to model the dynamics of transport of quantities like energy, the concentration of particles, etc. Roos et al. (2008). Different numerical methods such as Finite Element Method (FEM), Finite Difference (FD), and Finite Volume (FV) exist for the solution of the scalar convection-diffusion equation describing the dynamics of the species concentration. The scalar convection-diffusion equation with a large Peclet number (ratio of convection and diffusion) is called a singularly perturbed partial differential equation (SPDE). The solution of SPDE could contain interior and boundary layers, which are challenging to capture accurately by standard numerical methods Stynes (2005). For instance, the standard Galerkin FEM induces spurious oscillations in the numerical solution. Therefore, stabilization methods are often employed with FEM to reduce the spurious oscillations in the numerical solution and to capture the interior and boundary layers accurately Braack and Burman (2006), John and Knobloch (2007). Several methods such as SUPG Brooks and Hughes (1982b), Local Projection Stabilization (LPS) Braack and Burman (2006) have been proposed in the context of the finite element method. The key idea in the stabilization mentioned above is to add artificial diffusion, controlled via a stabilization parameter, to suppress spurious oscillations in the numerical solution. However, this approach poses another challenge: how much artificial diffusion should be added?

It is challenging to find an optimal value of the stabilization parameter; for further details on existing stabilization techniques, read the review John and Knobloch (2007).

Recently many neural network-based PDE solvers have been introduced Raissi et al. (2017), Han et al. (2018). However, applying these solvers for SPDEs to capture interior and boundary layers is challenging and induces significant numerical errors. It motivates the current work, where we utilize the neural network to identify an optimal stabilization parameter. In this context, we propose the "SPDE-Net": an artificial neural network to predict stabilization parameters for solving SPDEs using the streamline upwind/Petrov-Galerkin (SUPG) technique.

The paper is organized as follows: Section 2 reviews the existing stabilization and machine learning techniques for FEM. Further, the contributions and the novelty of the proposed technique are also highlighted. Section 3 describes the mathematical preliminaries required for the understanding of current work, such as convection-diffusion equation, its variational formulation, and SUPG stabilization. Section 4 presents the proposed technique, network architecture, and the learning method. Section 5 provides details of the experiments conducted and the metrics used for evaluation. Section 6 presents the results and analysis and comparison of different techniques. Finally, section 7 concludes the current research, and the scope for future extension of this work is discussed.

2. Related Work

2.1. Streamline Upwind Petrov Galerkin(SUPG)

Many stabilization techniques such as SUPG Brooks and Hughes (1982b), Local Projection Stabilization Braack and Burman (2006), Continuous Interior Penalty Burman (2009), Least-Square method and Subgrid Viscosity have been proposed in the literature for the solution of SPDEs. SUPG is a residual-based, prevalent stabilization technique and is the focus of this article. In this technique, we generally add a residual term in the direction of streamline to the weak form of the equation, and there remains no crosswind diffusion John and Knobloch (2007). Moreover, the addition of the residual term is controlled by a user-chosen parameter called the stabilization parameter, which significantly impacts the accuracy of the numerical solution. Adding an optimal stabilization is very important since adding more diffusion can result in smearing of layers, and adding less will not suppress spurious oscillations in the numerical solution. Lower numerical accuracy of the SUPG scheme for SPDEs boils down to the unavailability of an optimal stabilization parameter. In literature, several expressions for the stabilization parameter exist, and a comparison of existing techniques is given in John and Knobloch (2007). Although there is a closed-form expression for stabilization parameter for 1-dimensional problems, it is hard to derive a closed-form expression in higher dimensions.

2.2. Machine Learning in FEM

In recent times, machine learning has been quite useful in scientific computing. Many researchers have stated ANN as a strong candidate for solving unsolved problems of scientific computing leveraged by its universal approximation abilities Cybenkot (2006), Yadav et al. (2016) and free control over the hyperparameters. ANNs can successfully be used for solving

PDEs Lagaris et al. (1998), Sirignano and Spiliopoulos (2018) and learning PDEs from data by injecting data observations in PDE models Long et al. (2018), Raissi et al. (2017), Raissi et al. (2019). Recently, many researches have used ANN for aiding traditional numerical methods Brevis et al. (2020), Hesthaven and Ubbiali (2018). One such effort is made by Discacciati et al. (2020), to predict the optimal value of artificial dissipation to get rid of discontinuities from Gibbs phenomenon in higher-order numerical solvers and also to detect the location of these discontinuities in the solution. It has been shown at par performance of the ANN-based models with the traditional models. In another work, Hesthaven and Ubbiali (2018) has proposed POD-NN, which uses ANN in the interpolation step to develop an ANN aided non-intrusive Radial Basis(RB) method using proper orthogonal decomposition. In Capuano and Rimoli (2019), a smart finite element method is proposed which directly maps the element-based input and output to reduce the computational costs involved in solving large-scale systems of equations. However, in both of these papers, FEM was not part of the deep learning architecture but was only used to generate the dataset or speed up one or more numerical steps of FEM.

2.3. Current Work

The proposed SPDE-Net framework in this work has two components: one uses ANN to predict the stabilization parameter(τ), and the other uses FEM to solve the partial differential equations (PDE) using that predicted τ. In this study, we have used an interface to stitch the computational graphs generated by an FEM package and a deep learning framework (*PyTorch* Paszke et al. (2019)). We have considered the coefficients of the SPDE as the input features. In another work Yadav and Ganesan (2019), the stabilization parameter was predicted by reducing the residual of the equation, and here we have attempted to minimize the L^2-error instead to get better performance.

To summarize, we make the following contributions through this work:

- Developed an ANN-based supervised and L^2-error minimizing (L^2 EM) techniques for predicting the stabilization parameter in the SUPG method.
- Developed a training dataset based on the equation coefficients and demonstrated the prediction of global and local variants of stabilization parameter τ with ANN.
- Showed that ANN-aided FEM solvers solve 1-dimensional SPDE with lesser numerical error than that with pure neural network solvers such as PINNs

3. Preliminaries

In this section, we present a convection-diffusion problem with a boundary layer. Then, we derive the SUPG finite element formulation. We use standard notations $L^p(\Omega)$, $W^{k,p}(\Omega)$, $H^k(\Omega) = W^{k,2}(\Omega)$, where $1 \leq p < \infty$, $k \geq 0$, to denote the Sobolev space. Further, $(\cdot, \cdot)$ denotes the inner product in the $L^2(\Omega)$ space and $|v|$ stands for the Euclidean norm of $v \in \mathbb{R}$.

3.1. Convection Diffusion Problem

Let Ω be a bounded domain in $\mathbb{R}$. Consider a convection diffusion equation: find u(x): $\Omega \to \mathbb{R}$ subject to

$$\begin{aligned} -\epsilon u''(x) + bu'(x) &= f \text{ for } x \in (0,1) \\ \text{with} \quad u(0) &= L, \quad u(1) = R, \end{aligned} \tag{1}$$

where $u(x)$ is the unknown scalar function, ϵ is a small positive diffusion coefficient, b is convection coefficient and f is a given source term, L and R are given boundary values. The analytical solution of the problem (1) is given by:

$$\begin{aligned} u(x) &= \alpha x + \frac{(R - L - \alpha)[\exp^{(-\beta(1-x)}) - \exp^{-\beta}]}{1 - \exp^{-\beta}} + L \\ \text{where } \alpha &= \frac{f}{b} \quad \beta = \frac{b}{\epsilon} \end{aligned} \tag{2}$$

The given problem (1) has sharp layers near the boundary, which possesses the challenge to approximate the solution with Galerkin FEM. Hence, stabilization technique is often used.

3.2. Weak Formulation

In this section, the weak formulation of (1) is derived. Let $V = \{v \in H_0^1(\Omega)\}$. Multiply the equation (1) with a test function $v \in V$ and apply integration by parts to the higher order term. It results in the variational form of (1) as

Find $u \in H^1(\Omega)$ such that for all v

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

where

$$a(u, v) = \int_\Omega \epsilon u' v' \, dx + \int_\Omega bu' v \, dx \tag{4}$$

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

3.3. Galerkin's finite element discretization

Let T_h be an admissible decomposition of the domain Ω and K be a cell in T_h. Here, h is the mesh-size. Denote Ω_h as a collection of all cells. Let $V_h \subset V$ be a finite-dimensional subspace comprising continuous piecewise linear polynomial. Using this finite dimensional space, the discrete form of the model problem reads:

Find $u_h \in H^1(\Omega_h)$ such that for all $v_h \in V_h$ we have

$$a_h(u_h, v_h) = (f, v_h), \tag{6}$$

where

$$a_h(u, v) = \int_{\Omega_h} \epsilon u' v' \, dx + \int_\Omega bu' v \, dx, \quad (f, v) = \int_{\Omega_h} fv \, dx.$$

3.4. SUPG Stabilization

The standard Galerkin approximation of convection dominated problems induces spurious oscillations. Therefore, the SUPG stabilization Brooks and Hughes (1982a) is added to the variational form. Denote the residual $R(u)$ of the equation (1) as

$$R(u) = -\epsilon u'' + bu' - f.$$

This residual is added to the discrete form (6). Now, the SUPG form reads:
Find $u_h \in V_h$ for all $v_h \in V_h$ such that:

$$a_h(u_h, v_h) = (f, v_h) \tag{7}$$

where

$$a_h(u_h, v_h) = \epsilon(u_h', v_h') + (bu_h', v_h) + \sum_{K \in \Omega_h} \tau_K (R(u), bv_h')_K.$$

Here, τ_K is a user-chosen non-negative stabilization parameter, and plays an important role in the quality of the approximated solution. A very large value of τ_K can lead to unexpected smearing, whereas a low value will not suppress the spurious oscillations. Therefore, an optimal value of stabilization parameter is required, which can control oscillations and smearing both appropriately. As mentioned in Madden and Stynes (1997), no standard formula exists for τ for higher-dimensional problems. Hence, we attempt to predict the value of τ using deep learning for a one-dimensional problem to develop the theory and experimentation as a base for solving higher-dimensional problems later. For a local Peclet number, $Pe_K = \frac{bh}{2\epsilon}$, the existing expression for τ_K is given by

$$\tau_K = \frac{h}{2b}(\coth(Pe_K) - \frac{1}{Pe_K}); \qquad \text{where } \coth(x) = \frac{\exp(x) + \exp(-x)}{\exp(x) - \exp(-x)}. \tag{8}$$

This expression is only applicable to SPDEs with a constant convection coefficient. Furtehr, it is not directly extendable to the higher dimensions, but it is informative for developing a more generalized technique using ANNs. From this expression, we could deduce that the dependent variables used as input features to the proposed neural network, essentially the stabilization parameter τ depends on $(\epsilon,\ b,\ h)$.

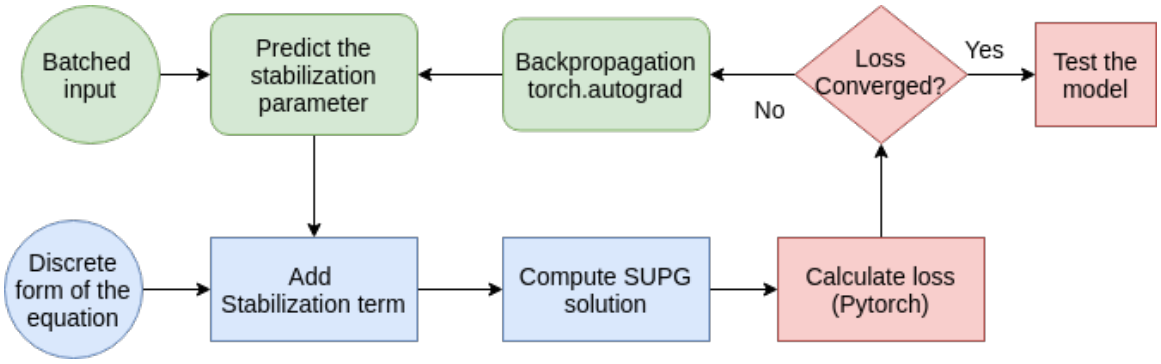

Figure 1: Overview of the training algorithm

3.5. Evaluation Metrics

Root Mean Square Error (RMSE) with standard τ (eq. 8) and L^2-error between $\hat{u}(\hat{\tau})$ and u (eq. 2) are the two metrics used to quantitatively evaluate the performance of the proposed technique.

$$RMSE = \frac{\sqrt{\sum_{i=1}^{N}(\hat{\tau} - \tau)^2}}{N} \tag{9}$$

$$L^2\text{-error: } ||e_h||_0 = ||u - u_h||_{L^2(\Omega)} = \left(\int_\Omega (\hat{u}(\hat{\tau}) - u)^2 dx\right)^{\frac{1}{2}}. \tag{10}$$

Here, N is the number of examples in the dataset.

4. SPDE-Net: Predicting SUPG stabilization parameter

The problem of approximating stabilization parameter (τ) is modeled as a regression task. For this, the τ is predicted using ANN with input features $\{\epsilon,\ b,\ h\}$. Two types of loss functions are considered for training the network. One is the supervised loss (calculated from the value of τ) used as a baseline, and the other is a L^2-error (calculated from the value of u) loss. For an i^{th} training sample,

$$\begin{aligned}
\hat{\tau}_i(\theta) &= G_\theta(\epsilon_i, b_i, h_i), \\
\hat{u}_i(\theta) &= u_{\text{SUPG}}(\epsilon_i, b_i, h_i, \hat{\tau}_i(\theta)), \\
\theta^*_{supervised} &= argmin \sum_{i=1}^{N} loss\left(\hat{\tau}_i(\theta), \tau_i\right), \\
\theta^*_{L^2EM} &= argmin \sum_{i=1}^{N} loss\left(\hat{u}_i(\theta), u_i\right),
\end{aligned}$$

where, G_θ is θ parametrized SPDE-Net, $\hat{u}_i(\theta)$ is the SUPG solution of eq. (7), τ_i is stabilization parameter (eq. (8)), u is the analytical solution (eq. (2)) and N is the number of training examples. We have to learn G by finding optimal θ^* through iterative back-propagation of loss. The end to end approximation process is shown schematically in Fig. (1).

4.1. Network architecture

The network architecture is shown in Fig. (2). The neural network model consists of three fully connected layers with tanh non-linearity as given below:.

$$\tanh(x) = \frac{exp(x) - exp(-x)}{exp(x) + exp(-x)} \tag{11}$$

As part of pre-processing, Pe is calculated for each input $\{\epsilon$, b and h$\}$ in the input normalization step given as below. The network outputs normalized $\hat{\tau}_{norm}$ which is re-scaled to

get the actual $\hat{\tau}$.

$$\text{Input to the neural network: } Pe = \frac{bh}{2\epsilon}$$
$$\text{Output of the neural network: } \hat{\tau}_{norm}$$
$$\text{Re-scaled output: } \hat{\tau} = \hat{\tau}_{norm}\frac{h}{b}$$

After that, we have used the obtained $\hat{\tau}$ to run testing experiments, as explained in the following sections. Normalization of input improves the result, as was also observed by Discacciati et al. (2020).

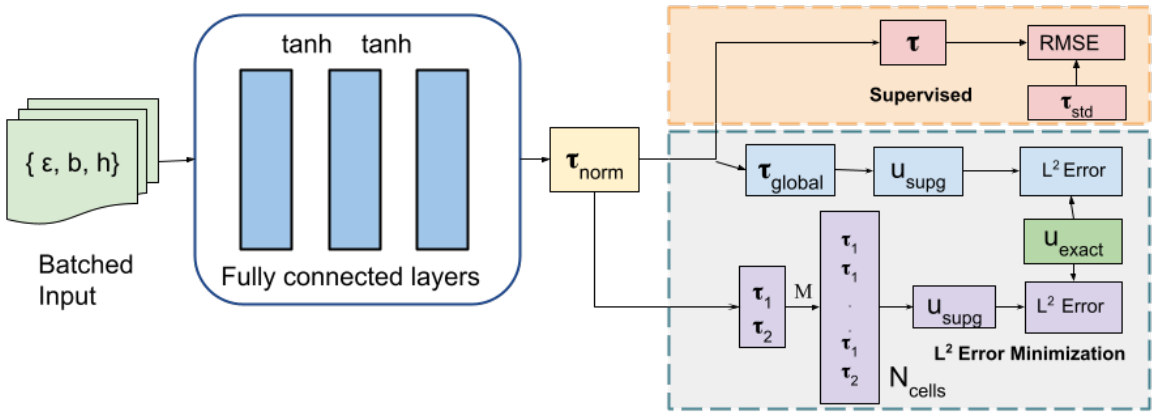

Figure 2: Schematic diagram of *SPDE-Net*: An end-to-end deep learning+FEM framework for solving SPDE

4.2. Supervised learning

Supervised learning model as given in Algorithm (1), uses RMSE between predicted $\hat{\tau}$ and the standard τ (eq. (8)) as a loss function. This model cannot be extended to the higher dimensions as the value of τ is not known in general, especially in higher dimensions. Nevertheless, it serves as an important baseline to assess the performance of the proposed L^2-error minimization technique.

Algorithm 1 SPDE-Net: Supervised Training

x: Input, BS: batchsize, η: learning rate, n_{epochs}: no. of training epochs, $n_{batches}$: no. of batches
Initialize the network parameters with random parameters θ_0
Initialize the Adam optimizer and stepLR scheduler
for $k = 1$ to n_{epochs} **do**
  $loss(\theta_k) = 0$
  for $m = 1$ to n_{batches} **do**
    $\hat{\tau}(\theta_k) = \text{SPDE-Net}(x_{(m-1)\times BS:m\times BS}; \theta_k)$
    $\tau = \frac{h}{2b}\left(\coth Pe - \frac{1}{Pe}\right)$
    $loss(\theta_k) += RMSE(\hat{\tau}, \tau)$
  end for
  $\theta_{k+1} = \theta_k - \eta_k \frac{\nabla loss(\theta_k)}{N}$
end for

4.3. L^2 error minimization((L^2 EM)

L^2-error minimization as given in Algorithm (2), is used in situations where the expression for τ is unknown but the analytical solution for the equation is known to train the network. Here, the L^2-error between the $\hat{u}(\hat{\tau})$ and the solution u (eq. (2)) is used as the loss function. We propose two variants of the stabilization parameter for this case as following:

- Global stabilization parameter (τ_g): A single predicted $\hat{\tau}_g$ is considered for the whole domain Ω_h.
- Local stabilization parameter (τ_{loc}): Each cell of the domain will have a local τ_K. It is to understand if having a non-uniform τ improves the accuracy of the numerical scheme. Two values $\hat{\tau}_{loc_1}$ and $\hat{\tau}_{loc_2}$ are predicted by the network for non-boundary and boundary layer region respectively. These values are then localised across the entire domain using matrix transformation. For a particular one-dimensional case (eq. (1)), the boundary region is either near $x = 0$(for $b < 0$) or $x = 1$(for $b > 0$). Hence, $\hat{\tau}_{loc_2}$ is used for the extremity, as given below (for $b > 0$):

$$M = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ \vdots & \vdots \\ 0 & 1 \end{bmatrix}_{N_{\text{cells}},2} \tag{12}$$

$$\hat{\tau} = [\hat{\tau}_{loc_1}(\theta), \hat{\tau}_{loc_2}(\theta)]^T \tag{13}$$

$$\tau_{loc} = M\hat{\tau} \tag{14}$$

$M(i, j) = 1$ means τ_j will be used in i_{th} cell of the discretized domain Ω_h. M is used to obtain $\hat{\tau}_K$ for each cell from the output of the neural network.

Algorithm 2 SPDE-Net: L^2 error minimization (L^2 EM)

x: Input, $\hat{u}$: solution with predicted τ, u: analytical solution, η: learning rate, bs: batchsize
Initialize network parameters with θ_0 randomly initialized parameters
Initialize the Adam optimizer and stepLR scheduler
for $k = 1$ to n_{epochs} **do**
  $loss(\theta_k) = 0$
  for $m = 1$ to n_{batches} **do**
    $\hat{\tau}(\theta_k) = \text{SPDE-Net}(x_{(m-1)\times bs:m\times bs}; \theta_k)$
    $loss(\theta_k) += (\int_\Omega (\hat{u}(\hat{\tau}(\theta_k)) - u)dx)^{\frac{1}{2}}$
  end for
  $\theta_{k+1} = \theta_k - \eta_k \frac{\nabla loss(\theta_k)}{N}$
end for

5. Experiments

This section represents the constituents of the experimental setup. It details the training, testing and validation dataset. Hyper-parameters obtained from the ablation study are given for training the *SPDE-NET*.

5.1. Dataset

We have considered the examples of steady singularly perturbed partial differential equations as given in eq. (1). A dataset has been generated by sampling the values of ϵ, b and h from the ranges mentioned in Table (1) and $f = 1$ for supervised training and L^2 EM. For each example, the solution is approximated using P_1 finite elements. Further, 20% samples from the training set are separated for validation. Held-out test dataset details are also given in Table (1).

Table 1: Dataset properties

Supervised/L^2 EM		
Input features	Training+validation dataset	Testing
ϵ	33 samples in $\left[10^{-16}, 10^{4}\right]$	32 samples in $\left[10^{-16}, 10^{-1}\right]$
b	$\{1.0, 1.1, 1.2, 1.3, 1.4\}$	$\{1.5, 1.6, 1.7\}$
h^{-1}	{30,35,40,45,60,70,80, 90,100,500}	{50}
Boundary (L, R)	$\{-1, 0, 1\}$	$\{-1, 0, 1\}$
Total data samples	4950	288

5.2. Hyper-parameters

The base model is trained from scratch using PyTorch with hyper-parameters as mentioned in Table (2). The network is trained using mini-batch gradient descent and Adam optimizer. The learning rate is controlled with stepLR scheduler by starting with a high initial learning

rate, which is gradually reduced by a factor of gamma after a specified step size depending on the type of training.

Table 2: Training hyperparameters

	Supervised	**L^2 EM**	
	τ	τ_g	τ_{loc}
Nonlinearity	tanh	tanh	tanh
Epochs	3200	80	100
Batch Size	32	64	32
Optimizer	Adam	Adam	Adam
Initial LR	0.0001	0.01	0.001
Scheduler	StepLR	StepLR	StepLR
Step size	1000	25	15
Gamma	0.1	0.2	0.5
Layer sizes	64,32,16	64,32,16	128,128,128

5.3. Split the data in critical and non-critical parts

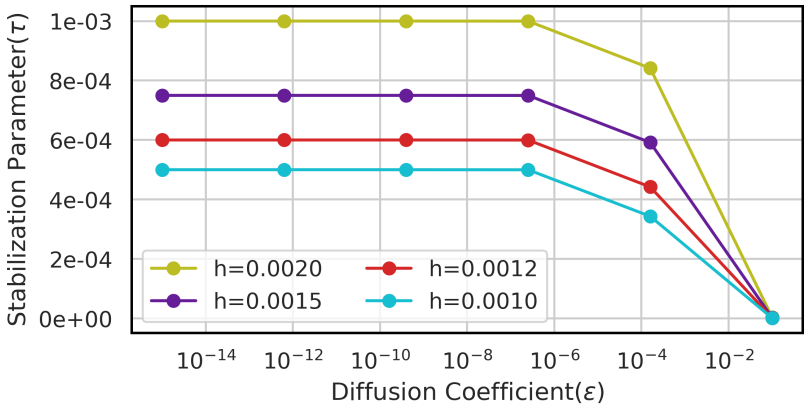

Figure 3: Variation of Standard(analytical) τ with ϵ and h

The dataset for 1D problems is divided into critical and non-critical parts for testing. It is based on the fact that as the value of ϵ reduces the problem becomes more perturbed. We have observed the variation of standard τ for b and h as shown in Fig. (3). We can see that the value of τ for ϵ in $[10^{-14}, 10^{-7}]$ is almost constant and it doesn't vary much with the input features, thus it needs to be tested separately so that it doesn't influence the testing error much. We call this data category to be critical, and the rest of the data points will fall into the non-critical category.

6. Results

In this section, the performance of each technique is shown on the test and validation dataset. Supervised technique, L^2 EM (τ_g) and L^2 EM (τ_{loc}) technique are compared in terms of RMSE error and the L^2 error. Perturbation analysis for each technique is done to check the performance variation as we increase the perturbation in the system by decreasing the value of ϵ for a given equation.

6.1. Performance

Table (3), shows the performance of each technique on the test and validation dataset in terms of average RMSE and L^2 error. RMSE is not demonstrated for L^2 EM(τ_{loc}) because in this case, the τ is not a scalar value, so calculating Euclidean distance is not possible. The supervised technique performs best in terms of RMSE due to the availability of ground truth. Both in terms of L^2 error and RMSE, L^2 EM(τ_g) technique is performing at par with the supervised technique for both validation and the test data. It shows that L^2 error can also be used for training loss as it achieves similar performance. It gives a reliable alternative in case of the unavailability of ground truth for training the network for such problems.

Table 3: Performance comparison of different techniques for validation and test dataset

	Validation data		**Test data**	
Technique	$\|\|\hat{u}(\hat{\tau}) - u\|\|_{L^2(\Omega_h)}$	$\|\|\hat{\tau} - \tau\|\|_{L^2(\Omega_h)}$	$\|\|\hat{u}(\hat{\tau}) - u\|\|_{L^2(\Omega_h)}$	$\|\|\hat{\tau} - \tau\|\|_{L^2(\Omega_h)}$
Supervised	5.13 e−6	2.79 e−7	7.88 e−6	3.72 e−7
L^2 EM(τ_g)	**5.00 e−6**	3.33 e−6	**7.76 e−6**	4.83 e−7
L^2 EM(τ_{loc})	6.42 e−5	NA	1.70 e−4	NA
PINN	8.11 e−3	NA	7.82 e−3	NA

6.2. Comparison with ANN based PDE solvers

PINNs are neural network based PDE solvers trained by minimizing the strong residual of the equation while encoding hidden laws as prior information. In contrast, the proposed network aims to minimize the L^2 error in the solution. Essentially, it is developed over the already established stabilized FEM technique. Hence it exploits the advantages of both conventional stabilized FEM technique and contemporary ANN techniques. One good check of the performance of SPDE-Net will be to compare its L^2 error with that of PINNs. We have compared the performance of the proposed network with the state-of-the-art PINN Raissi et al. (2017) networks, and the error comparison is shown in Table (4). SPDE-Net outperforms the residual-based PINN Raissi et al. (2017) network for solving singularly perturbed PDEs for all mesh sizes. Among the proposed neural network-based variants, L^2 error minimization (τ_g) technique gives the least error. It shows that L^2 error is a better choice than residual for loss. Also, reducing the RMSE error of τ is as good as reducing L^2 error of the solution for that of the analytical solution of the given problem.

Table 4: Comparison of SPDE-Net(Supervised and L^2 EM) with PINN in terms of L^2 error in the numerical solution produced by these techniques for test case $\epsilon = 1e{-}11, b = 1.0$ and different h)

h	**Supervised**	L^2 **EM**		**PINN**
		$\tau_{\mathbf{g}}$	$\tau_{\mathbf{loc}}$	
6.25 e−2	6.70 e−6	**2.72 e−6**	5.06 e−5	8.01 e−3
3.13 e−2	4.74 e−6	**1.92 e−6**	3.58 e−5	7.94 e−3
1.56 e−2	3.35 e−6	**1.36 e−6**	2.53 e−5	7.92 e−3
7.81 e−3	2.37 e−6	**9.60 e−7**	1.79 e−5	7.92 e−3
3.91 e−3	1.70 e−6	**6.90 e−7**	1.29 e−5	7.92 e−3
1.95 e−3	1.20 e−6	**4.88 e−7**	9.09 e−6	7.92 e−3

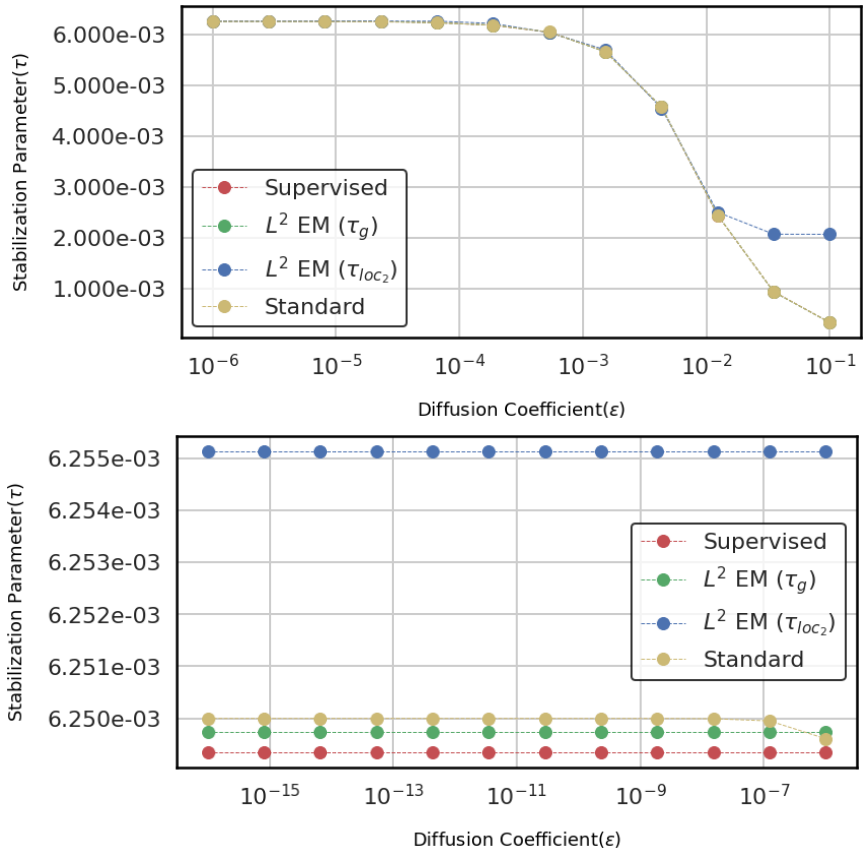

Figure 4: Perturbation analysis: Comparison of the τ from different techniques for (a) non-critical (b) critical test set

6.3. Perturbation analysis

One qualitative test of a prediction technique for τ is its susceptibility against small changes in the perturbation parameter (ϵ) with a constant b and h. Perturbation analysis for the proposed technique is done by testing it on 16 samples of ϵ from $[10^{-16}, 10^{-1}], b = 1.6, h = 0.02$.

For every test case, error and the stabilization parameter has been plotted for critical and non-critical data with a constant b and h as shown in Fig. (4), (5). For non-critical data-points, the value of τ decreases as we increase the value of ϵ, this trend is shown by all the techniques except L^2 EM(τ_{loc}) as shown in Fig. (4). Due to the deviation in τ from L^2 EM(τ_{loc}), it has maximum L^2 error in the solution as shown in Fig. (4). Supervised and L^2 EM(τ_g) has a comparable L^2 error and the value of τ.

For critical data points, the change in the value of τ is minimal, so all the graphs are flat and show similar trends as shown in Fig. (4(b)). The L^2 error minimization(τ_g) technique gives L^2 error almost similar to that of supervised technique, but the L^2 error minimization(τ_{loc}) technique is performing quite bad for both the data sets as can be seen in Fig. (5).

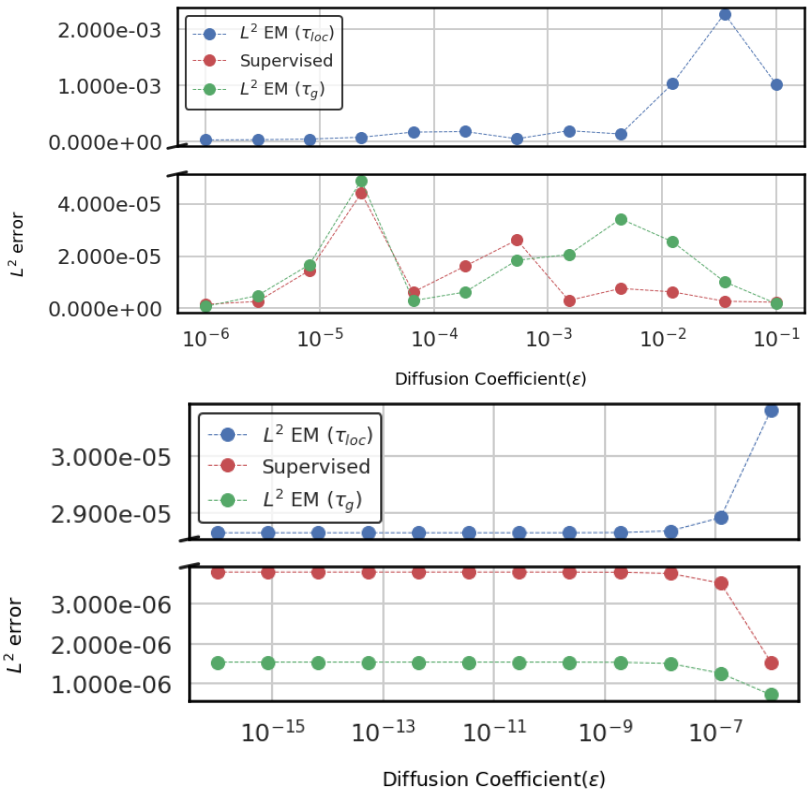

Figure 5: Perturbation analysis: Comparison of L^2 error from different techniques for (a) non-critical (b) critical test set

6.4. Qualitative analysis

For the model problem with $\epsilon = 1\,\mathrm{e}{-11}, b = 1.0, f = 1$, (which is a sample from critical dataset) the solution generated from Supervised, L^2 EM(τ_g) and L^2 EM(τ_{loc}), PINN network are compared with the analytical solution (2) in Fig. (6). All the proposed techniques show minimal oscillations and are quite close to the analytical solution. All the proposed techniques perform better than the PINN networks.

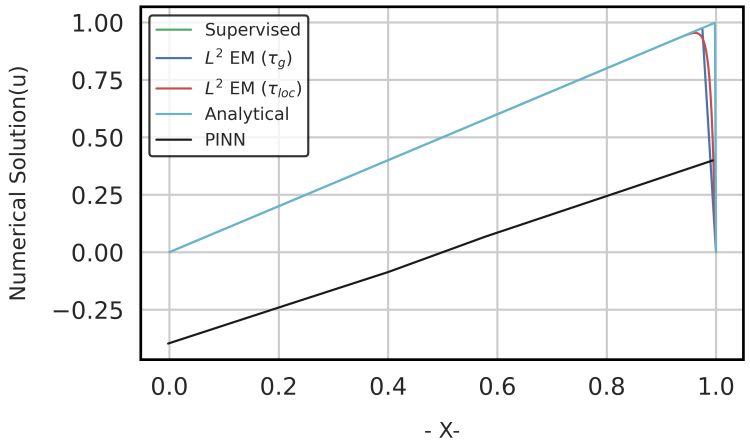

Figure 6: Qualitative analysis of solutions

7. Conclusion

In this paper, we presented a case for using deep learning to predict stabilization parameter in SUPG for solving singularly perturbed partial differential equations. It is successfully tested for various one-dimensional examples. The proposed network outperforms the state-of-the-art residual-based PINN networks for solving singularly perturbed differential equations. Based on the results obtained in this paper, we can observe that it is better to combine the neural networks with the capability of FEM instead of solely relying on ANNs for solving SPDEs. Moreover, L^2 EM performs at par and slightly better than the supervised technique and thus is helpful for training in the cases when the expression for stabilization parameter is not available even for the training.

Acknowledgments

First author would like to thank Ishaan Sood for his insights on Deep Learning pipeline.

References

M. Braack and E. Burman. Local projection stabilization for the oseen problem and its interpretation as a variational multiscale method. *SIAM Journal on Numerical Analysis*, 43(6): 2544–2566, 2006. doi: 10.1137/050631227. URL https://doi.org/10.1137/050631227.

Ignacio Brevis, Ignacio Muga, and Kristoffer G. van der Zee. Data-Driven Finite Elements Methods: Machine Learning Acceleration of Goal-Oriented Computations. *arXiv:2003.04485 [cs, math]*, March 2020. URL http://arxiv.org/abs/2003.04485. arXiv: 2003.04485.

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

Alexander N. Brooks and Thomas 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(1): 199 – 259, 1982b. ISSN 0045-7825. doi: https://doi.org/10.1016/0045-7825(82)90071-8. URL http://www.sciencedirect.com/science/article/pii/0045782582900718.

Erik Burman. A posteriori error estimation for interior penalty finite element approximations of the advection-reaction equation. *SIAM J. Numerical Analysis*, 47:3584–3607, 11 2009. doi: 10.1137/080733899.

German Capuano and Julian J. Rimoli. Smart finite elements: A novel machine learning application. *Computer Methods in Applied Mechanics and Engineering*, 345:363–381, March 2019. ISSN 0045-7825. doi: 10.1016/j.cma.2018.10.046. URL http://www.sciencedirect.com/science/article/pii/S0045782518305541.

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

Niccolò Discacciati, Jan S. Hesthaven, and Deep Ray. Controlling oscillations in high-order Discontinuous Galerkin schemes using artificial viscosity tuned by neural networks. *Journal of Computational Physics*, 409:109304, May 2020. ISSN 0021-9991. doi: 10.1016/j.jcp.2020.109304. URL http://www.sciencedirect.com/science/article/pii/S0021999120300784.

Jiequn Han, Arnulf Jentzen, and Weinan E. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115 (34):8505–8510, 2018. ISSN 0027-8424. doi: 10.1073/pnas.1718942115. URL https://www.pnas.org/content/115/34/8505.

J. S. Hesthaven and S. Ubbiali. Non-intrusive reduced order modeling of nonlinear problems using neural networks. *Journal of Computational Physics*, 363:55–78, June 2018. ISSN 0021-9991. doi: 10.1016/j.jcp.2018.02.037. URL http://www.sciencedirect.com/science/article/pii/S0021999118301190.

Volker John and Petr Knobloch. On spurious oscillations at layers diminishing (sold) methods for convection-diffusion equations: Part ia review. *Computer Methods in Applied Mechanics and Engineering*, 196:2197–2215, 03 2007. doi: 10.1016/j.cma.2006.11.013.

I. E. Lagaris, A. Likas, and D. I. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 9(5):987–1000, 1998. doi: 10.1109/72.712178.

Zichao Long, Yiping Lu, Xianzhong Ma, and Bin Dong. Pde-net: Learning pdes from data. In *International Conference on Machine Learning*, pages 3214–3222, 2018.

Niall Madden and Martin Stynes. Efficient generation of oriented meshes for solving convection-diffusion problems. *International Journal for Numerical Methods in Engineering*, 40:565–576, 01 1997. doi: 10.1002/(SICI)1097-0207(19970215)40:3⟨565::AID-NME80⟩3.0.CO;2-7.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 8026–8037. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf.

Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561*, 2017.

Maziar Raissi, Paris Perdikaris, and George 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*, 378:686–707, 2019.

H.G. Roos, M. Stynes, and L. Tobiska. *Robust Numerical Methods for Singularly Perturbed Differential Equations: Convection-Diffusion-Reaction and Flow Problems*. Springer Series in Computational Mathematics. Springer Berlin Heidelberg, 2008. ISBN 9783540344674. URL https://books.google.co.in/books?id=tdnw1p2JoEIC.

Justin Sirignano and Konstantinos Spiliopoulos. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339–1364, 2018. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2018.08.029. URL https://www.sciencedirect.com/science/article/pii/S0021999118305527.

Martin Stynes. Steady-state convection-diffusion problems. *Acta Numerica*, 14:445–508, 2005. doi: 10.1017/S0962492904000261.

Sangeeta Yadav and Sashikumaar Ganesan. How deep learning performs with singularly perturbed problems? In *2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, pages 293–297, 2019. doi: 10.1109/AIKE.2019.00058.

Sangeeta Yadav, Amandeep Kaur, and N. S. Bhauryal. Resolving the celestial classification using fine k-nn classifier. In *2016 Fourth International Conference on Parallel, Distributed and Grid Computing (PDGC)*, pages 714–719, 2016. doi: 10.1109/PDGC.2016.7913215.