Predicting the stabilization quantity with neural networks for Singularly Perturbed Partial Differential Equations

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

Abstract

We propose *SPDE-Net*, an artificial neural network (ANN) to predict the stabilization parameter for 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 ANN. Three training strategies for the ANN have been proposed, i.e. supervised, L^2 error minimization (global), and L^2 error minimization (local). The proposed method has been observed to yield accurate results and even outperform some of the existing state-of-the-art ANN-based partial differential equation (PDE) solvers, such as Physics Informed Neural Network (PINN).

1. Introduction

The solution of a convection-diffusion equation is required by many applications to model physical phenomena which involve the transfer of quantities like particles, energy, fluid, etc. When the convection velocity is large in magnitude as compared to the diffusion, the problem becomes a Singularly Perturbed Differential Equation, and the numerical solution shows spurious oscillations. These equations have a small perturbation parameter such that their solution approaches a discontinuous limit when the diffusion parameter approaches the value Zero (Roos et al., 2008). Conventional techniques such as the Galerkin FEM are inadequate to solve SPDEs due to the presence of interior/boundary layers.

Stabilization techniques are often employed to inject extra artificial diffusion in the equation so as to reduce the spurious oscillations in the numerical solution. The numerical accuracy of any stabilization technique depends a lot on the stabilization parameter. It is challenging to find an optimal value of the stabilization parameter. Many research efforts have been made in this direction [(John et al., 1998)].

Many neural network-based PDE solvers also exist in literature such as residual minimizing PINN networks. Numerical solutions for SPDEs using PINN networks have large numerical errors. So, both the stabilized FEM and ANN-based PDE solvers have their limitations. One way to improve both of these techniques would be to use stabilized FEM with the stabilization parameter predicted from ANN. This is the motivation for this current research work. In the end, we will show that ANN aided stabilized FEM schemes perform better than pure ANN-based PDE solvers.

In this research, we propose "*SPDE-Net*": an artificial neural network (ANN) for predicting stabilization parameter for solving SPDEs using streamline upwind/Petrov-Galerkin (SUPG) technique. *SPDE-Net* combines the universal approximating abilities of ANN with the solving power of FEM for solving SPDEs. The purpose of this work is to aid conventional stabilization techniques with contemporary methods of optimization and prediction. The paper is organized as follows: Section 2 reviews the existing stabilization and machine learning techniques for FEM. Here we highlight the contributions and the novelty of the proposed technique. Section 3 explains mathematical preliminaries required for the understanding of current work, such as the convection-diffusion equation, its weak formulation and SUPG stabilization. Section 4 explains the proposed technique, network architecture and learning methods. Section 5 gives details of the learning experiments conducted and the metrics used for evaluation. Section 6 shows results and analysis while comparing the methods. Section 7 concludes the current research and hints its further scope for extension.

2. Related Work

2.1. Streamline Upwind Petrov Galerkin(SUPG)

Many stabilization techniques have been developed such as SUPG [(Brooks & Hughes, 1982)], Local Projection Stabilization (LPS), and Subgrid Viscosity (SGV). SUPG is a residual-based very popular stabilization technique and is the focus of this article. In this technique, a residual-based

[*]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.

term is generally added to the weak form of the equation, only in the direction of streamline, so there remains no cross-wind diffusion. Usually, the residual term is multiplied with a user-chosen coefficient called the stabilization parameter which impacts the accuracy of the numerical solution very much. Knowing an optimal value of the stabilization parameter is very important as adding more diffusion can result in extra smearing of layers at the same point, adding less will show up oscillations in the numerical solution. The lower numerical accuracy of the SUPG scheme for SPDEs boils down to the unavailability of an optimal stabilization parameter.

2.2. Machine Learning in FEM

Now machine learning is quite useful in scientific computing. Many pieces of research have stated ANN to be a strong candidate for solving unsolved problems of scientific computing leveraged by its universal approximation abilities and free control over the hyperparameters. It can be successfully used for solving PDEs [(Lagaris et al., 1998)], [(Sirignano & Spiliopoulos, 2018)] and learning PDEs from data by injecting data observations in PDE models [(Raissi et al., 2017)]. In another work, [(Hesthaven & Ubbiali, 2018)] has proposed POD-NN which uses ANN in the interpolation step in order to develop an ANN-aided non-intrusive Radial Basis(RB) method using proper orthogonal decomposition. In [(Capuano & Rimoli, 2019)], a smart finite element method is proposed that 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 for generating the dataset or to speed up one or more numerical steps of FEM. To the best of our knowledge, there has been no study that trains an end-to-end deep learning model involving FEM as part of the computational graph itself. In this work, the software framework has two parts: 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. Integrating these two solvers/libraries is the most challenging part since most of the FEM packages do not generate a computational graph and even if they perform there is hardly any existing software that has variables that can store the gradients of the functions operated on any variable. Gradients remain a significant requirement for performing any deep-learning task. In this study, we have used an interface to stitch the computational graphs generated by a FEM package (*DOLFIN* [(Logg & Wells, 2010)]) and a deep learning framework (*PyTorch*). In similar research for Discontinuous Galerkin(DG) [(Discacciati et al., 2020)], the numerical solution is taken as the input to the neural network. This limits its applicability to the cases where the solution is identified in advance. To overcome this, we have considered equation-based input features. In another work [(Yadav & Ganesan, 2019)], the stabilization parameter was predicted by reducing the residual of the equation, 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 Artificial Neural Network(ANN) based supervised and L^2 error minimizing (L^2 EM) techniques for predicting stabilization parameter for Streamline Upwind Petrov Galerkin(SUPG) Technique.
- 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 NN-aided FEM solvers solve SPDE with lesser numerical error than that with pure neural network solvers such as PINNs

3. Preliminaries

In this section, a convection-diffusion problem with a boundary layer is presented. Further, the SUPG finite element formulation is derived. We will use standard notations $L^p(\Omega)$, $W^{k,p}(\Omega)$, $H^k(\Omega) = W^{k,2}(\Omega)$, where $1 \leq p < \infty$, $k \geq 0$, for the usual function spaces and Sobolev space respectively. $(\cdot, \cdot)$ will denote the inner product in the $L^2(\Omega)$ space. $|\boldsymbol{a}|$ stands for the Euclidean norm of a vector $\boldsymbol{a} \in \mathbb{R}^d$, where $d = 1, 2$.

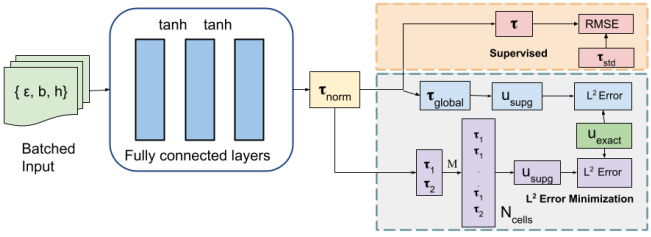
3.1. Convection Diffusion Problem

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

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

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.

Figure 1. Schematic diagram of *SPDE-Net*: An end-to-end deep learning+FEM framework for solving SPDE



3.2. Weak Formulation

In this section, the weak formulation of eq. (1) is derived. Let $V = v \in H_0^1(\Omega)$. Multiply eq. (1), with a test function $v \in V$ and do integration by parts on the higher order term. As an output, the weak formulation of eq. (1) will be to find $u \in H^1(\Omega)$ such that for all v

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

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

$$a(u, v) = \int_\Omega \epsilon u' v' dx + \int_\Omega b u' v dx \tag{3}$$

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

3.3. SUPG Stabilization

As the Galerkin solution is unstable for convection-dominated problems, we are adding the SUPG stabilization term proposed by (Brooks & Hughes, 1982). The residual $R(u)$ of eq. (1) is

$$R(u) = -\epsilon u'' + bu' - f \tag{5}$$

It has been added to the discrete form. Now, the modified weak form is to find $u_h \in V_h$ such that:

$$\begin{aligned} a_h(u_h, v_h) =& f(v_h) \quad \forall \quad v_h \in V_h \\ a_h(u_h, v_h) =& \epsilon(u_h', v_h') + (bu_h', v_h) + \sum_{i \in \Omega_h} \tau_i (R(u), bv_h')_{\Omega_h} \end{aligned} \tag{6}$$

τ_i is a user-chosen non-negative stabilization parameter. Its value plays an important role in the quality of the approximated solution. A very large value of τ can lead to unexpected smearing, whereas a low value will not suppress the spurious oscillations. So an optimal value of the stabilization parameter is required to control oscillations and smearing both properly. As mentioned in [(Madden & Stynes, 1997)], literature does not offer a standard formula for τ_i for higher-dimensional problems. So we have attempted to predict the value of τ using deep learning for one-dimensional problems to develop the theory and experimentation as a base for solving higher-dimensional problems later. Following is an existing formula for τ.

$$\begin{aligned} \text{For local Peclet number, } Pe = \frac{bh}{2\epsilon}; \\ \tau = \frac{h}{2b}(\coth(Pe) - \frac{1}{Pe}); \end{aligned} \tag{7}$$

where h is the mesh size. This formula is only applicable to equations with constant convection coefficients. It is not directly extendable to the higher dimensions, but it is quite informative for developing a more generalized technique using ANNs. From this formula, we could deduce that the dependent variables which will be used as input features to the proposed neural network, essentially the stabilization parameter τ depend on the (ϵ, b, h).

4. SPDE-Net: Predicting SUPG stabilization parameter

The problem of approximating the stabilization parameter has been moulded into 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) which is 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,

$$\hat{\tau}_i(\theta) = G_\theta(\epsilon_i, b_i, h_i), \tag{8}$$

$$\hat{u}_i(\theta) = u_{\text{SUPG}}(\epsilon_i, b_i, h_i, \hat{\tau}_i(\theta)), \tag{9}$$

$$\theta^*_{supervised} = argmin \sum_{i=1}^{N} loss\left(\hat{\tau}_i(\theta), \tau_i\right), \tag{10}$$

$$\theta^*_{L^2 EM} = argmin \sum_{i=1}^{N} loss\left(\hat{u}_i(\theta), u_i\right), \tag{11}$$

where, G_θ is θ parametrized SPDE-Net, $\hat{u}_i(\theta)$ is the SUPG solution of eq. (6), τ_i is stabilization parameter (eq. (7)), 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 Figure (3).

4.1. Network architecture

The network architecture is shown in Figure (1). The neural network model consists of three fully connected layers with $tanh$ non-linearity. As part of pre-processing, Pe is calculated for each input $\{\epsilon$, b and h$\}$ in the input normalization step given below. The network outputs normalized $\hat{\tau}_{norm}$ which is re-scaled to get the actual $\hat{\tau}$.

$$\text{Input to the neural network: } Pe = \frac{bh}{2\epsilon} \tag{12}$$

$$\text{Output of the neural network: } \hat{\tau}_{norm} \tag{13}$$

$$\text{Re-scaled output: } \hat{\tau} = \hat{\tau}_{norm} \frac{h}{b} \tag{14}$$

The obtained $\hat{\tau}_{norm}$ is, thereafter, used to run testing experiments as explained in the further sections.

4.2. Supervised learning

The supervised learning model uses RMSE between predicted τ and the standard τ (eq. (7)) as a loss function.

This is a regression task which does not use output sigmoid activation. This model cannot be extended to the higher dimensions as the value of τ is not generally known, especially in higher dimensions. It serves as an important baseline to assess the performance of the proposed L^2 error minimization technique.

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

L^2 error minimization is used in situations where the expression for τ is unknown but the analytical solution for the equation is known for training the network. Here the L^2 error between the $\hat{u}(\hat{\tau})$ and the solution u 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 $\hat{\tau}$ is considered for the whole domain Ω_h.

- Local Stabilization Parameter (τ_{loc}): Each cell of the domain will have a local τ. 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.

5. Experiments

This section represents the constituents of the experimental setup. It details about the training, testing and validation dataset. Hyper-parameters obtained from ablation study are given followed by the evaluation metrics considered 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 (4) and $f = 1$ for supervised training and L^2 EM. For each example, the domain Ω is discretized using P_1 finite element with mesh-size h. 20% samples from the training set are separated for validation.

5.2. Hyper-parameters

The base model is trained from scratch using PyTorch. 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.

6. Results

In this section, the performance of each technique is computed for 1D problems on the test and validation dataset. Supervised technique, L^2 EM (τ_g) and L^2 EM (τ_{loc}) technique have been 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. RMSE is not shown for L^2 error minimization(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 due to the availability of ground truth. Both in terms of L^2 error and RMSE, the L^2 error minimization(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 the case of the unavailability of ground truth for training the network for such problems.

6.2. Comparison with ANN-based PDE solvers

PINNs are neural networks-based PDE solvers trained by minimizing the strong residual of the equation while encoding hidden laws as prior information whereas the proposed network aims to minimize the L^2 error in the solution but in a different manner. Essentially, it is developed over the already established stabilized FEM technique. Hence it exploits the advantages of both the conventional stabilized FEM technique and the contemporary ANN technique. 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 (2). SPDE-Net outperforms the residual-based PINN 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 with respect to that of the analytical solution of the given problem.

7. Conclusion

In this paper, we presented a case for using deep learning to predict stabilization parameters for SUPG for solving singularly perturbed partial differential equations. It is successfully tested for various 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 say it is better to combine the neural networks with the capability of FEM instead of solely relying on neural networks for solving SPDEs. L^2 EM performs at par and slightly better than the supervised technique and thus is useful for training when the formula for stabilization parameter is not even known for the training.

Table 2. 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**
		$\tau_{\mathbf{g}}$	$\tau_{\mathbf{loc}}$	
$6.25\,\mathrm{e}{-2}$	$6.70\,\mathrm{e}{-6}$	$\mathbf{2.72\,e{-6}}$	$5.06\,\mathrm{e}{-5}$	$8.01\,\mathrm{e}{-3}$
$3.13\,\mathrm{e}{-2}$	$4.74\,\mathrm{e}{-6}$	$\mathbf{1.92\,e{-6}}$	$3.58\,\mathrm{e}{-5}$	$7.94\,\mathrm{e}{-3}$
$1.56\,\mathrm{e}{-2}$	$3.35\,\mathrm{e}{-6}$	$\mathbf{1.36\,e{-6}}$	$2.53\,\mathrm{e}{-5}$	$7.92\,\mathrm{e}{-3}$
$7.81\,\mathrm{e}{-3}$	$2.37\,\mathrm{e}{-6}$	$\mathbf{9.60\,e{-7}}$	$1.79\,\mathrm{e}{-5}$	$7.92\,\mathrm{e}{-3}$
$3.91\,\mathrm{e}{-3}$	$1.70\,\mathrm{e}{-6}$	$\mathbf{6.90\,e{-7}}$	$1.29\,\mathrm{e}{-5}$	$7.92\,\mathrm{e}{-3}$
$1.95\,\mathrm{e}{-3}$	$1.20\,\mathrm{e}{-6}$	$\mathbf{4.88\,e{-7}}$	$9.09\,\mathrm{e}{-6}$	$7.92\,\mathrm{e}{-3}$

References

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

Capuano, G. and Rimoli, J. J. 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.

Discacciati, N., Hesthaven, J. S., and Ray, D. 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.

Hesthaven, J. S. and Ubbiali, S. 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.

John, V., Matthies, G., Schieweck, F., and Tobiska, L. A streamline-diffusion method for nonconforming finite element approximations applied to convection-diffusion problems. *Computer Methods in Applied Mechanics and Engineering*, 166:85–97, November 1998. doi: 10.1016/S0045-7825(98)80014-5.

Lagaris, I. E., Likas, A., and Fotiadis, D. I. 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.

Logg, A. and Wells, G. N. Dolfin: Automated finite element computing. *ACM Transactions on Mathematical Software*, 37(2), 2010. doi: 10.1145/1731022.1731030.

Madden, N. and Stynes, M. 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.

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.

Roos, H., Stynes, M., and Tobiska, L. *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.

Sirignano, J. and Spiliopoulos, K. 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.

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

A. Appendix

A.1. Performance

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	$5.13\,\mathrm{e}{-6}$	$2.79\,\mathrm{e}{-7}$	$7.88\,\mathrm{e}{-6}$	$3.72\,\mathrm{e}{-7}$
L^2 EM(τ_g)	$\mathbf{5.00\,e{-6}}$	$3.33\,\mathrm{e}{-6}$	$\mathbf{7.76\,e{-6}}$	$4.83\,\mathrm{e}{-7}$
L^2 EM(τ_{loc})	$6.42\,\mathrm{e}{-5}$	NA	$1.70\,\mathrm{e}{-4}$	NA
PINN	$8.11\,\mathrm{e}{-3}$	NA	$7.82\,\mathrm{e}{-3}$	NA

A.2. Dataset

Table 4. Dataset properties

Supervised/L^2 Error-Minimization		
Parameters	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

A.3. Evaluation Metrics

Root Mean Square Error(RMSE) with standard τ (eq. 7) and L^2 error between $\hat{u}(\hat{\tau})$ and u (eq. 17) are the two metrics used to quantitatively evaluate the performance of the proposed technique. N is the no. of examples in the dataset.

$$RMSE = \frac{\sqrt{\sum_{i=1}^{N}(\hat{\tau} - \tau)^2}}{N} \tag{15}$$

$$L^2 \text{ error: } ||e_h||_0 = ||u - u_h||_{L^2(\Omega)} = \left(\int_{\Omega} (\hat{u}(\hat{\tau}) - u)^2 dx\right)^{\frac{1}{2}} \tag{16}$$

A.4. Qualitative analysis

For testsample $\epsilon = 1\,\mathrm{e}{-11}, b = 1.0, f = 1$, (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 (17) in Figure (2). All the proposed techniques show minimal oscillations and are quite close to the analytical solution as compared to the solution produced by PINN networks.

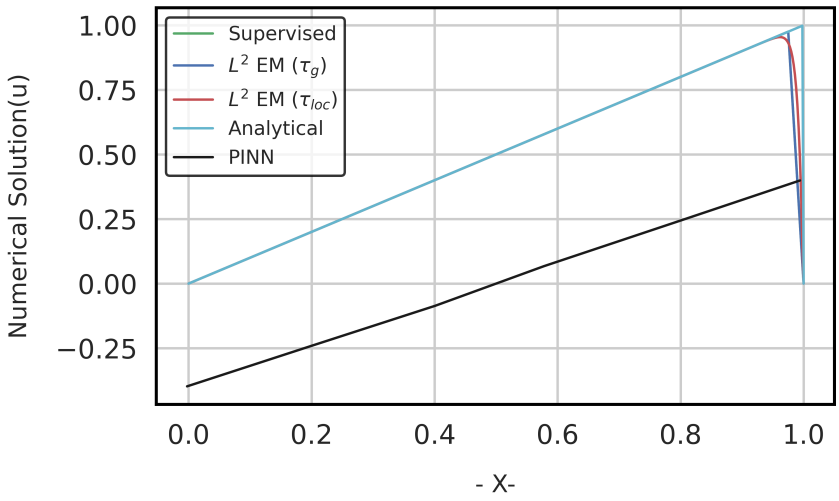


Figure 2. Qualitative analysis of solutions

A.5. Analytical solution of the equation

$$u(x) = \alpha x + \frac{(R - L - \alpha)[\exp^{(-\beta(1-x)}) - \exp^{-\beta}]}{1 - \exp^{-\beta}} + L$$
$$\text{where } \alpha = \frac{f}{b} \quad \beta = \frac{b}{\epsilon} \tag{17}$$

A.6. Training Pipeline

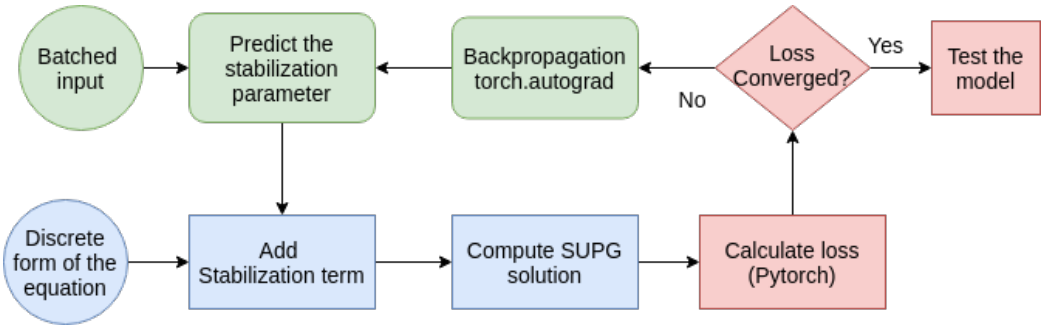


Figure 3. Overview of training algorithm