http://www.aimspress.com/journal/mine

Mathematics in Engineering, 5(2): 1–26.
DOI:10.3934/mine.2023032
Received: 30 October 2021
Revised: 23 March 2022
Accepted: 21 April 2022
Published: 12 May 2022

*Research article*

# A machine learning approach to enhance the SUPG stabilization method for advection-dominated differential problems

**Tommaso Tassi**[1], **Alberto Zingaro**[2,*] **and Luca Dede'**[2]

[1] Oliver Wyman Srl, Via Broletto 16, 20121, Milano, Italy
[2] MOX, Laboratory of Scientific Computing, Dipartimento di Matematica, Politecnico di Milano, Piazza Leonardo da Vinci, 32, 20133, Milano, Italy

* **Correspondence:** Email: alberto.zingaro@polimi.it; Tel: +390223994604.

**Abstract:** We propose using machine learning and artificial neural networks (ANNs) to enhance residual-based stabilization methods for advection-dominated differential problems. Specifically, in the context of the finite element method, we consider the streamline upwind Petrov-Galerkin (SUPG) stabilization method and we employ ANNs to optimally choose the stabilization parameter on which the method relies. We generate our dataset by solving optimization problems to find the optimal stabilization parameters that minimize the distances among the numerical and the exact solutions for different data of differential problem and the numerical settings of the finite element method, e.g., mesh size and polynomial degree. The dataset generated is used to train the ANN, and we used the latter "online" to predict the optimal stabilization parameter to be used in the SUPG method for any given numerical setting and problem data. We show, by means of 1D and 2D numerical tests for the advection-dominated differential problem, that our ANN approach yields more accurate solution than using the conventional stabilization parameter for the SUPG method.

**Keywords:** partial differential equations; finite element method; stabilization methods; streamline upwind Petrov-Galerkin; machine learning; artificial neural networks

## 1. Introduction

The Galerkin-finite element (FE) method applied to partial differential equations (PDEs) with advection terms dominating over diffusion ones may suffer of numerical instability [22, 34]. These numerical instabilities cause the numerical solution to exhibit oscillations that increase in amplitude with a local increment of the transport dominance over diffusion, i.e., as soon as the local Péclet number becomes larger than one. In order to eliminate (or at least mitigate) numerical instabilities, a general employed strategy is to consider the generalized Galerkin method, i.e., the Galerkin method with additional stabilization terms [34]. Examples of stabilization methods to reduce numerical oscillations in the advection dominated regimes are for instance the upwind and streamline-diffusion method, both methods that are not strongly consistent. Instead, examples of strongly consistent methods are the Galerkin-least-squares, the streamline upwind Petrov-Galerkin (SUPG), and the Douglas-Wang methods [6, 34, 41]. In particular, for these strongly consistent stabilization methods, the formulation depends on the residual of the PDE (in strong formulation), other than on a parameter – called *stabilization parameter* – whose definition is crucial for the success of the stabilization strategies. Determining the values of such stabilization parameter is not straightforward, especially for 2D and 3D problems. In addition, a universal formulation of such parameter is lacking, especially as it is strongly dependent on data of the model and the numerical setting used for the FE approximation of the PDEs, e.g., low and high order FE methods, spectral element methods, isogeometric methods, etc. [7, 12, 42]. Some of the formulations for the stabilization parameter have been derived from analytical considerations on the differential problem in 1D, others are suitable only for a specific numerical approximation method, while the most comprehensive ones incorporate some (empirical) dependence on numerical setting, as e.g., the order of the FE [5, 6, 8–10, 16, 18, 38, 43–45].

In this work, we propose using machine learning (ML) [19, 30, 50] to make computers learn autonomously the values of the stabilization parameter for the FE approximation of advection dominated PDEs. Artificial neural networks (ANNs) [27, 46] are widely popular in ML and Deep Learning for a wide array of applications: they are indeed very versatile tools that are increasingly finding their way in scientific computing [29, 31], especially in the context of numerical approximation of PDEs. For instance, as substitute to standard numerical methods, ANNs can be employed as meshless methods in physics informed neural networks (PINNs) to directly approximate the solution of the PDE as it is trained by minimizing the (strong) residual of the PDE [35–37]. ANNs are also largely employed in a data-driven fashion in the context of model order reduction for parametric PDEs [17, 20, 21, 40, 51] and to enhance the stability properties of numerical methods for PDEs [13]. In fluid dynamics modelling, ANNs are massively adopted for flow features extractions, modelling, optimization and control. For flow features extractions, ANNs are used through clustering and classification to classify wake topologies [11]; for modeling fluid dynamics by reconstructing specific flows such as the near wall field in a turbulent flow [28] and for flow optimization [32], and control for aerodynamics applications [4]. ANNs are also used in a data-driven framework as a manner for providing alternative closure models for stress tensor [14] in Reynolds-Average Navier-Stokes (RANS) equations, for sub-grid scale models in large eddy simulation (LES) turbulence models [23, 24, 48, 49, 52], or for model learning input-output relationships in complex physical processes [39].

In this work, we use ANNs to learn the optimal stabilization parameter in advection dominated PDEs that are discretized by means of the FE method: the goal is to enhance the accuracy of the SUPG FE method by optimally selecting the stabilization parameter under different data of the PDE and numerical settings of the FE method. We found that the proposed ANN-enhanced stabilization method allows to improve accuracy and stabilization properties of the numerical solution compared to those results obtained by analytical expressions of the stabilization parameter. The numerical results obtained shed light also on the possibility to apply the presented strategy to learn closure laws for stabilization and turbulence models of fluid dynamics, as for instance to learn the stabilization parameters in the Variational Multiscale–LES model to model transitional and turbulent flows [3, 15, 54].

This work is organized as follows: in Section 2, we recall the SUPG stabilization method for advection-diffusion equations; in Section 3, we present our numerical strategy to compute an optimal SUPG stabilization parameter through a feed-forward fully connected ANN. In Section 4 we validate our method by comparing the ANN results with those obtained with the 1D advection-diffusion problem from which the expression of the theoretical stabilization parameter has been derived. In Section 5 we first show the ANN's training and we present our numerical results on the 2D advection-diffusion problem used for training and we finally generalize our findings using the ANN's prediction on a different advection-diffusion problem. Finally, in Section 7 we draw our conclusions highlighting possible future developments.

## 2. The SUPG method for advection-diffusion problems

We briefly recall the advection-diffusion differential problem and the SUPG stabilization method for the advection dominated regime.

Let $\Omega \in \mathbb{R}^d$, $d = 1, 2, 3$ be the physical domain with $\partial\Omega$ being its boundary. We consider the following problem in the unknown function $u$:

$$\begin{cases} -\nabla \cdot (\mu \nabla u) + \boldsymbol{\beta} \cdot \nabla u = f & \text{in } \Omega, \\ u = g & \text{on } \partial\Omega, \end{cases} \tag{2.1}$$

where $\mu, \boldsymbol{\beta}$, and $f$ are assigned functions or constants, with $\mu \in L^\infty(\Omega), \boldsymbol{\beta} \in [L^\infty(\Omega)]^d$, with $\nabla \cdot \boldsymbol{\beta} \in L^2(\Omega)$, and $f \in L^2(\Omega)$. The Dirichlet datum on the boundary is $g \in H^{1/2}(\partial\Omega)$. Let $V_g = \{v \in H^1(\Omega) : v|_{\partial\Omega} = g\}$ and $V_0 = H^1_0(\Omega)$; the weak formulation of Eq (2.1) reads

$$\text{find } u \in V_g \ : \ a(u, v) = F(v) \quad \text{for all } v \in V_0. \tag{2.2}$$

with the bilinear form $a(u, v)$ and the linear functional $F(v)$ respectively defined as:

$$\begin{aligned} a(u, v) &:= \int_\Omega \mu \, \nabla u \cdot \nabla v \, d\Omega + \int_\Omega v \boldsymbol{\beta} \cdot \nabla u \, d\Omega, \\ F(v) &:= \int_\Omega f \, v \, d\Omega. \end{aligned}$$

We consider a family of function spaces $V_h \subset V$ (either for $V_g$ and $V_0$) dependent on a parameter $h$ such that $\dim(V_h) = N_h < \infty$. Let $X^h_r = \{v^h \in C^0(\overline{\Omega}) : v^h|_K \in \mathbb{P}_r, \text{ for all } K \in \mathcal{T}_h\}$ be the function space of the FE discretization with piecewise Lagrange polynomials of degree $r \geq 1$, $\mathcal{T}_h$ a triangulation of $\Omega$ and $h$ the characteristic size of the mesh, comprised of elements $K \in \mathcal{T}_h$. By setting $V_h = X^r_h \bigcap V_0$, the Galerkin FE method applied to Eq (2.2) reads

$$\text{find } u_h \in V_{g,h} \ : \ a(u_h, v_h) = F(v_h) \quad \text{for all } v_h \in V_h, \tag{2.3}$$

where $V_{g,h} = X^r_h \bigcap V_g$. The standard Galerkin-FE method, in Eq (2.3), can generate numerical oscillations on $u_h$ if the problem is dominated by the advection term. In particular, these numerical instabilities can arise if the local Péclet number is $\mathbb{P}\mathrm{e}_h > 1$, where

$$\mathbb{P}\mathrm{e}_h = \frac{|\boldsymbol{\beta}| h}{2\mu}. \tag{2.4}$$

The generalized Galerkin method ( [33]) allows for eliminating or mitigating numerical oscillations by adding stabilization terms to the standard Galerkin formulation as

$$\text{find } u_h \in V_{g,h} \ : \ a(u_h, v_h) + b_h(u_h, v_h) = F(v_h) \quad \text{for all } v_h \in V_h. \tag{2.5}$$

In the SUPG method, the additional stabilization term reads:

$$b_h(u_h, v_h) = \sum_{K\in\mathcal{T}_h} \int_K R(u_h)\, \tau_K\, \frac{1}{2} \left(\nabla \cdot (\boldsymbol{\beta} v_h) + \boldsymbol{\beta} \cdot \nabla v_h\right) d\Omega, \tag{2.6}$$

where $R(u_h)$ is the residual in strong formulation of Eq (2.1), which is defined as:

$$R(u_h) := -\nabla \cdot (\mu \nabla u_h) + \boldsymbol{\beta} \cdot \nabla u_h - f.$$

The term $\tau_K$, appearing in Eq (2.6), is the *stabilization parameter*, which is the focus of this work. The stabilization parameter $\tau_K$ is generally defined locally, i.e., mesh element by element. In this paper, we consider a uniform stabilization parameter, thus $\tau_K = \tau$ for all $K \in \mathcal{T}_h$.

A universal and optimal definition of $\tau$ in terms of the problem data and numerical settings like the mesh size and FE degree is lacking. An extensive review of stabilization parameters for the SUPG method is reported in [25]. The most common choices for $\tau$ come from an analytic derivation made for the advection-diffusion problem in 1D with $f = 0$ and approximated by means of linear finite elements, which reads:

$$\widetilde{\tau}_1 = \frac{h}{2|\boldsymbol{\beta}|}\, \xi(\mathbb{P}\mathrm{e}_h), \tag{2.7}$$

where $\xi(\theta)$ is the upwind function:

$$\xi(\theta) = \coth(\theta) - \frac{1}{\theta}, \quad \theta > 0.$$

If a uniform mesh is used, as we consider in this paper, then the value of $h$ is uniform over $\mathcal{T}_h$; if in addition $\boldsymbol{\beta}$ and $\mu$ are constant, then this implies that $\tau$ is uniform over $\mathcal{T}_h$ . The choice of $\tau$ made in Eq (2.7) represents an optimal choice of the stabilization parameter as it yields a nodally exact numerical solution for the 1D advection-diffusion problem if $f = 0$ and the FE polynomial order is $r = 1$ [18, 34]. Thus, the stabilization parameter of Eq (2.7) may not be fully effective to provide the optimal stabilization for a general advection-diffusion problem, e.g., to guarantee a nodally exact solution in 2D/3D or when using FE degrees larger than $r = 1$. A commonly used generalization of the formula of Eq (2.7) to higher FE degrees ($r > 1$) is presented in [18] and reads:

$$\widetilde{\tau}_r = \frac{h}{2|\boldsymbol{\beta}| r}\, \xi\left(\frac{\mathbb{P}\mathrm{e}_h}{r}\right). \tag{2.8}$$

Differently from Eq (2.7), the stabilization parameter $\widetilde{\tau}_r$ in Eq (2.8) takes into account the contribution of higher polynomial degrees $r$. Still, this formula is not optimal as it does not guarantee a nodally exact numerical solution. Our goal is to find a general and optimal expression of the stabilization parameter holding for advection-diffusion problems and FE approximations of degree $r \geq 1$ to be dependent on: the dimension $d$, the FE degree $r$, the mesh size $h$, the forcing term $f$, the diffusion coefficient $\mu$ and the transport coefficient $\beta$.

## 3. ANN-based approach for determining the optimal stabilization parameter

We present our approach to determine the optimal stabilization parameter $\tau$ for the SUPG stabilization method by using an ANN. We consider as features (inputs) of our ANN: the FE degree $r$, the mesh size $h$ and the global Péclet number $\mathbb{P}e_g := |\boldsymbol{\beta}|L/(2\mu)$ of the advection-diffusion problem, where $L$ is the characteristic length of the problem, $\boldsymbol{\beta} \in \mathbb{R}^d$ and $\mu \in \mathbb{R}$ . As we consider $\boldsymbol{\beta}$ to be uniform and fixed, using $\mathbb{P}\mathrm{e}_g$ as input corresponds to varying the value of the diffusion coefficient $\mu$. The features (input) of the ANN read:

$$\mathbf{x}^{(i)} = \left[r,\, h,\, \mathbb{P}\mathrm{e}_g\right]; \tag{3.1}$$

the *target* (output) of interest is the optimal SUPG stabilization parameter that we denote with $\tau^*$:

$$\mathbf{y}^{(i)} = [\tau^*]\,. \tag{3.2}$$

The first step consist in generating the dataset, i.e., pairs of inputs and outputs to be used for training the ANN (*data generation* step). This consists in choosing repeatedly and randomly values of the features $\mathbf{x}^{(i)}$ in given ranges. These features are used to feed an *optimization problem* that, through an optimizer and a suitable error measure, provide an optimal stabilization parameter $\mathbf{y}^{(i)}$, i.e., the target of the given feature. Such optimization problem considers as error measure $E(\tau)$: the mismatch between the numerical solution $u_h$ and the exact one $u_{\text{ex}}$ over the nodes of the FE mesh. $E(\tau)$ reads as:

$$E(\tau) = \sum_{k=1}^{K_h} |\, e(\boldsymbol{x}_k, \tau)\,|, \qquad e(\boldsymbol{x}_k, \tau) = u_h(\boldsymbol{x}_k; \tau) - u(\boldsymbol{x}_k). \tag{3.3}$$

being $\boldsymbol{x}_k$ the $k$-th node of the FE mesh $\mathcal{T}_h$ and $K_h$ the number of nodes. $E(\tau)$ is an approximation of the $L_1(\Omega)$ norm. To find the minimum of $E(\tau)$ for different problem configurations we solve, for each instance of the input parameters $\mathbf{x}^{(i)}$, the following optimization problem:

$$\text{find } \tau^* \;:\quad \min_{\tau} E(\tau). \tag{3.4}$$

Thus, the dataset generated consists of $m$ pairs of inputs-outputs $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$, with $i = 1, \dots, m$. The latter is used for *training* the ANN. The latter will be then used to predict the optimal stabilization parameter $\mathbf{y}^{(j)} = \tau_{\text{ANN}}$ to be used for the FE approximation of the advection-diffusion problem in the settings provided by $\mathbf{x}^{(j)}$.

We report in Figure 1 a sketch of the procedure used to build the ANN for the prediction of the optimal stabilization parameter for any new feature $\mathbf{x}^{(j)}$.

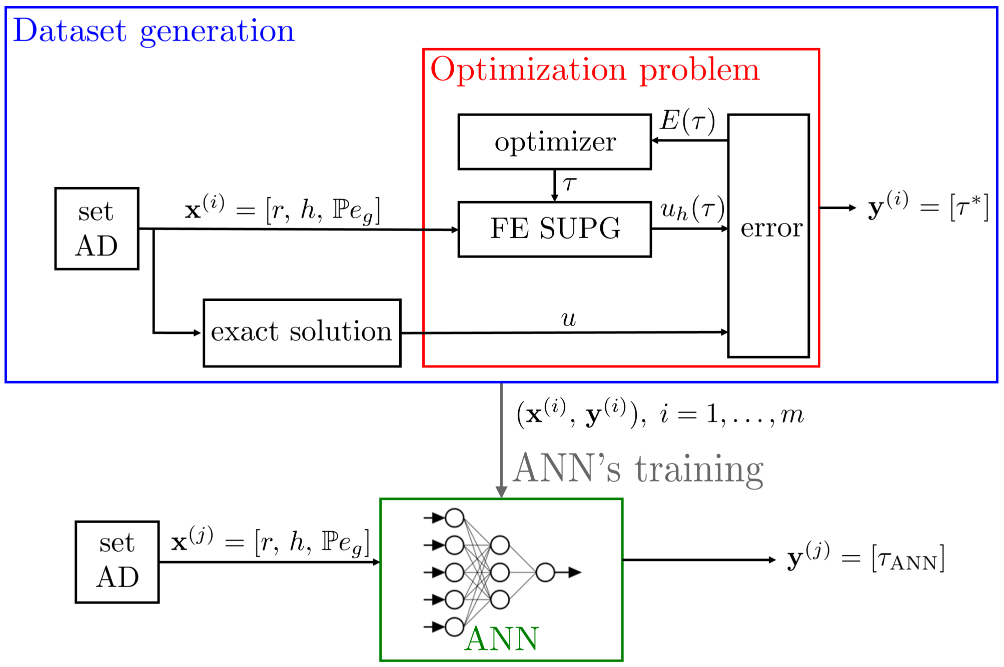

Dataset generation
Optimization problem
optimizer
$E(\tau)$
$\tau$
set AD
$\mathbf{x}^{(i)} = [r,\, h,\, \mathbb{P}e_g]$
FE SUPG
$u_h(\tau)$
error
$\mathbf{y}^{(i)} = [\tau^*]$
exact solution
$u$
$(\mathbf{x}^{(i)},\, \mathbf{y}^{(i)}),\ i = 1, \ldots, m$
ANN's training
set AD
$\mathbf{x}^{(j)} = [r,\, h,\, \mathbb{P}e_g]$
ANN
$\mathbf{y}^{(j)} = [\tau_{\mathrm{ANN}}]$

**Figure 1.** Representation of the strategy used for learning the the relation among $\tau$ and the advection-diffusion parameters and numerical settings.

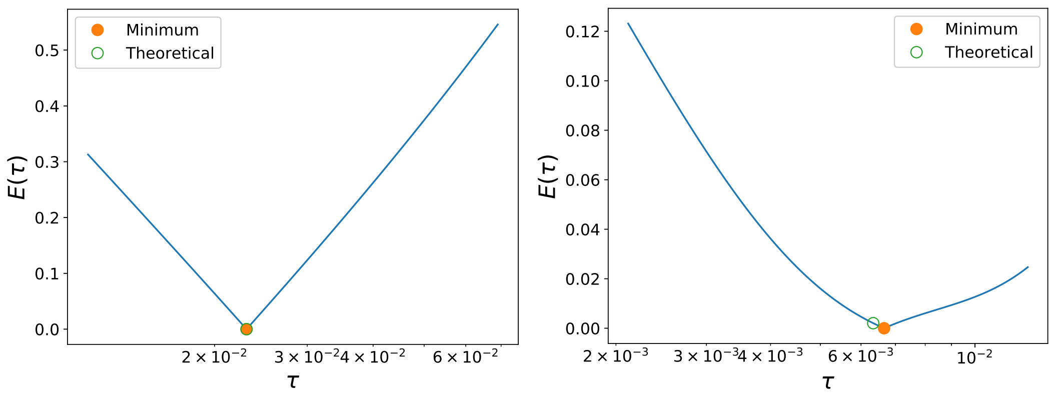
## 4. Numerical validation

In this section, we validate the proposed numerical strategy by means of a 1D advection-diffusion problem from which the expression of $\widetilde{\tau}_1$ in Eq (2.7) has been derived. In particular, we consider the advection-diffusion problem in Eq (2.1) with $\Omega = (0, 1)$, $f = 0$ and with Dirichlet BCs $u = 0$ on $x = 0$ and $u = 1$ on $x = 1$. It admits the following exact solution:

$$u(x) = \frac{\exp\left(\frac{\beta}{\mu}x\right) - 1}{\exp\left(\frac{\beta}{\mu}\right) - 1}.$$

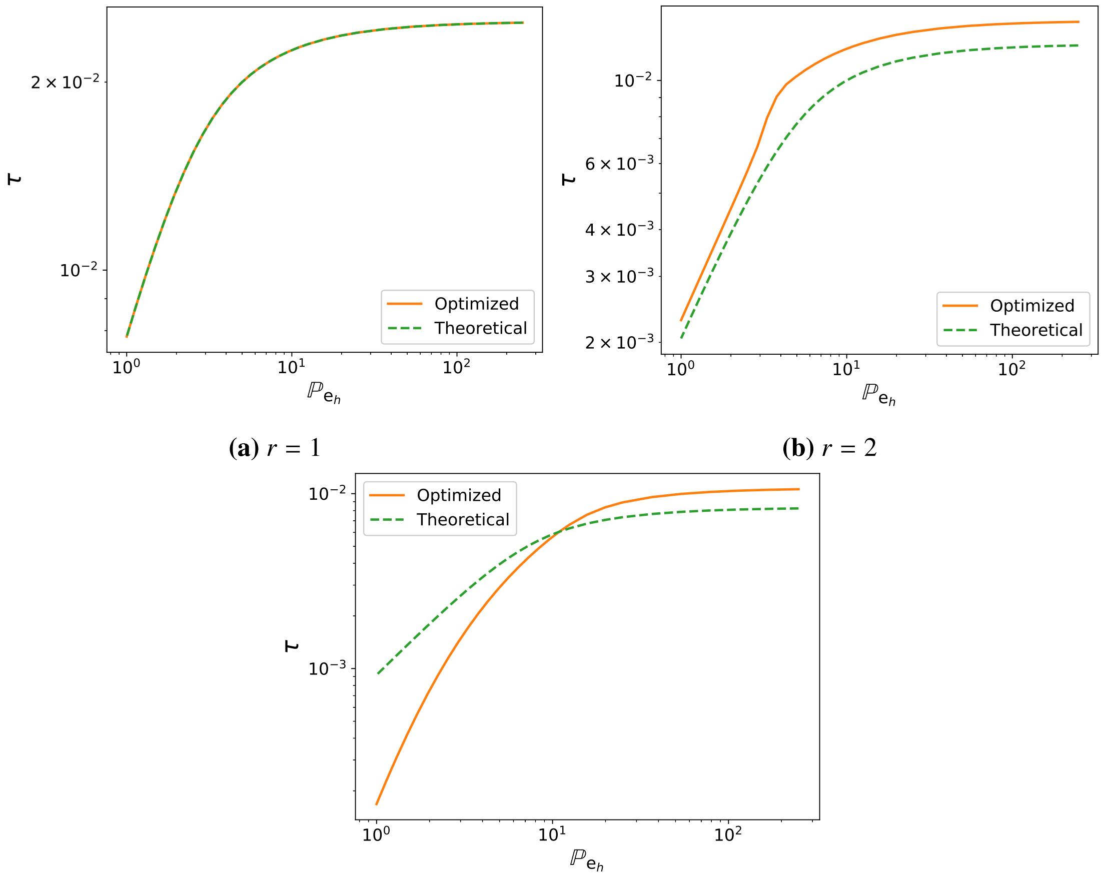
We recall that the stabilization parameter $\widetilde{\tau}_1$ of Eq (2.7) yields a nodally exact numerical solution if Eq (2.1) is solved by linear FE ($r = 1$).

We plot in Figure 2 the error measure $E(\tau)$ against the value of the stabilization parameter $\tau$, with $h = 1/20$, $\mathbb{P}e_h = 12.5$ and by using $r = 1$ (Figure 2a) and $r = 3$ (Figure 2b). We observe that $E(\tau)$ shows a minimum in $\tau^*$, thus suggesting the possibility to use an optimization algorithm to solve the problem. Specifically, we employ the L-BFGS-B optimization algorithm from `SciPy`, an open source library for `Python` [47]. Instead, the advection-diffusion problem has been solved using the FE open source library `FEniCS` [2], by applying the SUPG method on a uniform mesh. Particularly, the optimal value $\tau^*$ found for the case $r = 1$ corresponds to the one provided by the theory in Eq (2.7). For the case $r = 3$, we observe a optimal value of $\tau^*$ for which the error is minimized: in this case, it does not corresponds to the stabilization parameter $\widetilde{\tau}_r$ provided by the theory in Eq (2.8): as a matter of fact, the latter arises from empirical considerations to extend the parameter $\widetilde{\tau}_1$ in Eq (2.7) to polynomials degrees $r > 1$. However, this does not ensure a nodally exact numerical solution.



**(a)** $r = 1$ **(b)** $r = 3$

**Figure 2.** Error (3.3) of numerical solutions with SUPG method vs. the stabilization parameter $\tau$ for the 1D problem in Section 4 with $h = 1/20$ and $\mathbb{P}\mathrm{e}_h = 12.5$; comparison between the theoretical value $\widetilde{\tau}_r$ of Eq (2.8) and the optimal one $\tau^*$.



**(a)** $r = 1$ **(b)** $r = 2$

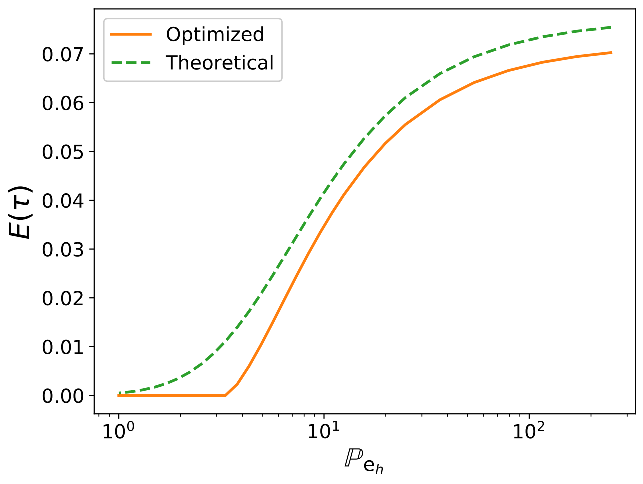
**(c)** $r = 3$

**Figure 3.** Comparison of the optimal stabilization parameter $\tau^*$ (full line) and the theoretical one $\widetilde{\tau}_r$ of Eq (2.8) (dashed line) against $\mathbb{P}\mathrm{e}_h$, with $h = 1/20$. Results are referred to the 1D advection-diffusion problem described Section 4.

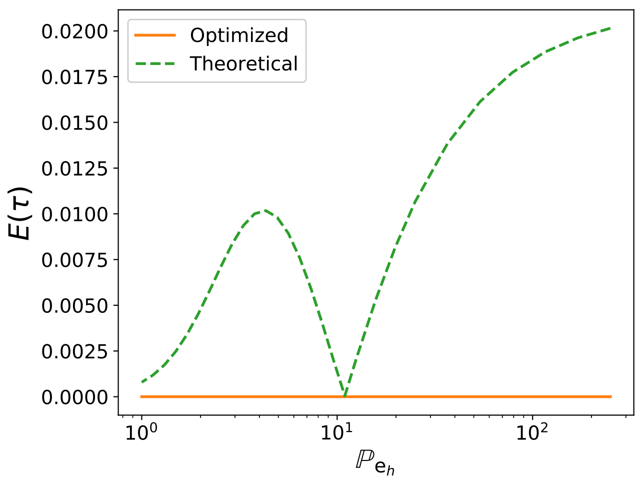
We solve the optimization problem for different values of the local Péclet number $1 \leq \mathbb{P}\text{e}_h \leq 250$ and we plot in Figure 3 the optimal stabilization parameter $\tau^*$ against the local Péclet number for $r = 1, 2$, and 3.

Figure 3a shows that, by employing a linear FE space $r = 1$, the optimal stabilization parameter $\tau^*$ obtained by minimizing $E(\tau)$ exactly matches the theoretical one $\widetilde{\tau}_1$ for every value of $\mathbb{P}\text{e}_h$, thus confirming the findings of Figure 2a. This also serves as validation of the optimization procedure that we proposed. By recalling that $\widetilde{\tau}_1$ of Eq (2.7) guarantees the numerical solution to be exact at nodes for this particular advection-diffusion problem and $r = 1$, we can infer that the optimization procedure is meaningful and it can be therefore exploited further, for example for $r > 1$. On the other hand, Figure 3b and 3c show instead a mismatch between the theoretical $\widetilde{\tau}_r$ of Eq (2.8) and the optimal one $\tau^*$, thus confirming the findings reported in Figure 2b.

In order to better appreciate the differences in the numerical solutions due to the choice of the stabilization parameters, we report, in Figure 4, a comparison of the error $E(\tau)$ obtained by means of the optimal $\tau^*$ and theoretical $\widetilde{\tau}_r$ stabilization parameters for $r = 2$ and 3. Moreover, we report in Figure 5 a comparison among the exact solution $u$, the SUPG-stabilized numerical solution $u_h^*$ obtained with the optimal stabilization parameter $\tau^*$, and the numerical solution $\widetilde{u}_h$ obtained with theoretical one $\widetilde{\tau}_r$. Specifically, we consider FE of degree $r = 3$ and two different values of $\mathbb{P}\text{e}_h$. In both these cases, the optimal parameter $\tau^*$ leads to a more accurate solution with respect to using the theoretical parameter $\widetilde{\tau}_r$. In particular, when $\mathbb{P}\text{e}_h$ is "small", $\widetilde{\tau}_r$ leads to overshooting in the numerical solution, while if $\mathbb{P}\text{e}_h$ is "large" the theoretical stabilization parameter leads to undershooting. Conversely, the optimal parameter $\tau^*$ accurately intercepts the exact solution at the nodes.

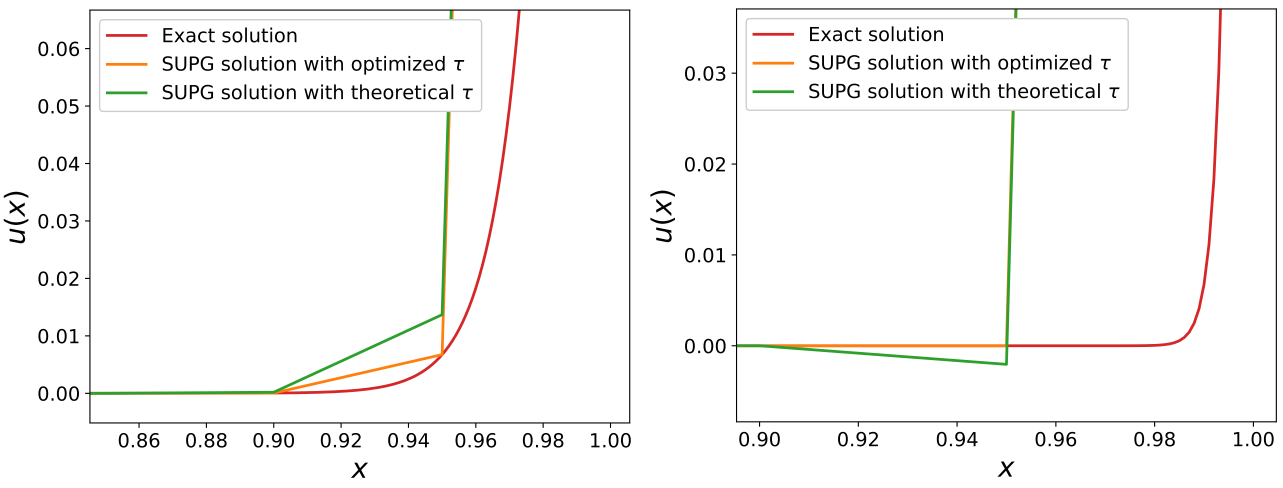


**(a)** $r = 2$



**(b)** $r = 3$

**Figure 4.** Comparison of the error of Eq (3.3) obtained with the optimal $\tau^*$ (full line) and theoretical $\widetilde{\tau}_r$ (dashed line) for varying $\mathbb{P}\text{e}_h$ with $h = 1/20$. Results referred to the 1D advection-diffusion problem of Section 4.



**(a)** $\mathbb{P}\mathrm{e}_h = 2.5$ **(b)** $\mathbb{P}\mathrm{e}_h = 12.5$

**Figure 5.** Boundary layers of the numerical solutions $u_h^*$ and $\widetilde{u}_h$ obtained with the SUPG method with optimal $\tau^*$ and theoretical one $\widetilde{\tau}_r$ for $h = 1/20$ and $r = 3$, respectively; comparison with the exact solution $u$ ($u_h^*$ is nodally exact at the node in $x = 0.95$). Results are referred to the 1D advection-diffusion problem of Section 4.

## 5. Numerical results

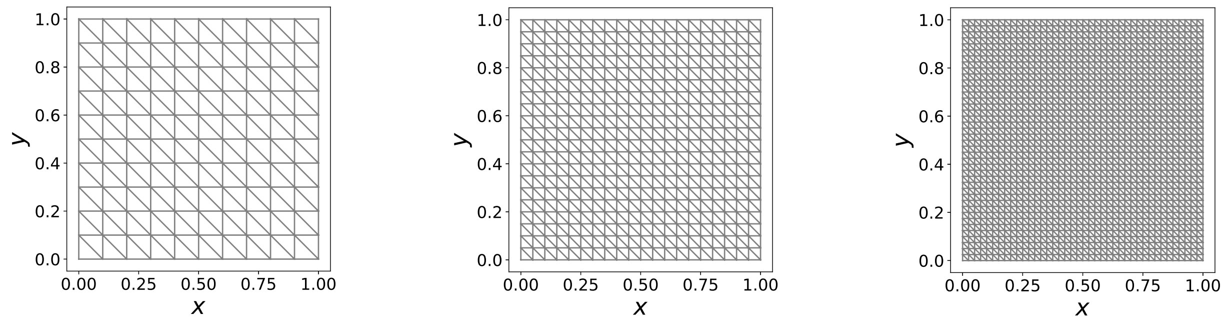
We first introduce the training set of our problem and we detail the setup of the ANN that we use in this work. Then, we show the prediction of the stabilization parameter by means of the ANN on different advection-diffusion problems.

### 5.1. *Training the ANN for a 2D advection-diffusion problem*

We apply the strategy presented so far to a 2D advection-diffusion problem to generate the dataset for the training of the ANN. Specifically, we consider in Eq (2.1): $\Omega = (0,1)^2$, $f = 0$, and $\boldsymbol{\beta} = (1,1)$. We prescribe the following exact solution on the whole boundary $\partial\Omega$:

$$u(x,y) = \frac{e^{(x/\mu)} - 1}{e^{(1/\mu)} - 1} + \frac{e^{(y/\mu)} - 1}{e^{(1/\mu)} - 1}. \tag{5.1}$$

We generate in $\Omega$ a structured mesh $\mathcal{T}_h$ of triangles with `FEniCS` [2], as shown in Figure 6. We generate the dataset by repeatedly solving the optimization problem of Eq (3.4) for varying set of features as described in Section 3; specifically, we choose the features as reported in Eq (3.1). The complete dataset contains $m = 900$ examples with $r = \{1,2,3\}$, $h = \left\{\frac{\sqrt{2}}{10}, \frac{\sqrt{2}}{20}, \frac{\sqrt{2}}{40}\right\}$ and values of $\mathbb{P}\mathrm{e}_g$ randomly chosend in the range $[7, 70'710]$ (uniform distribution), which yields $\mu \in [10^{-5}, 10^{-1}]$. We summarize the details for generating the dataset in Table 1. Figure 7 provides an overview of the dataset used for the training of the ANN, specifically the features $\mathbf{x}^{(i)} = [r,\ h,\ \mathbb{P}e_g]$ and the target $\mathbf{y}^{(i)} = [\tau^*]$. In this case, the characteristic length $L$ is fixed and set equal to $L = 1$.



**(a)** $h = \sqrt{2}/10$ **(b)** $h = \sqrt{2}/20$ **(c)** $h = \sqrt{2}/40$

**Figure 6.** Structured meshes used for the FE approximation of the 2D advection-diffusion problem.

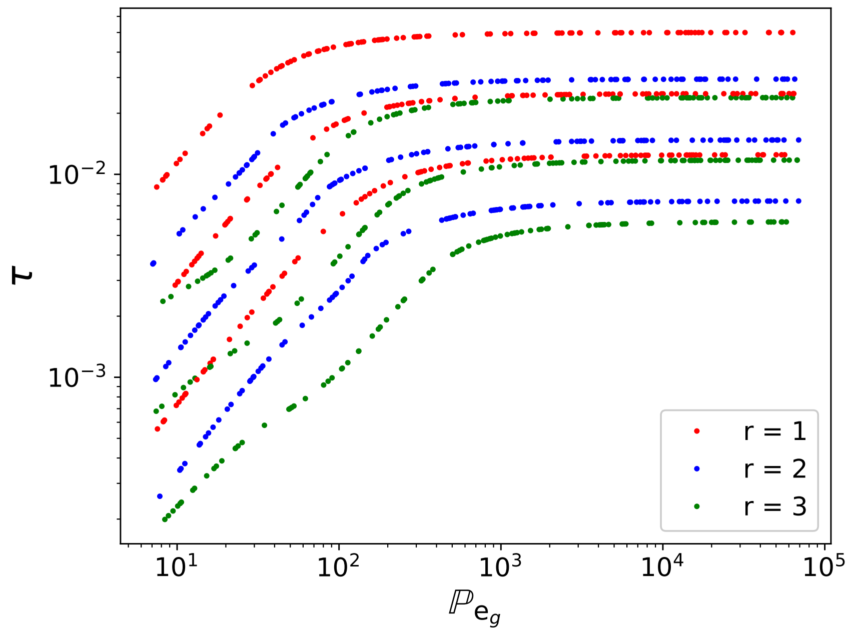


**Figure 7.** Visualization of the dataset used for the ANN training: target $\tau^*$ against feature $\mathbb{P}e_g$ colored by feature $r = 1$ (red), 2 (blue), and 3 (green) for different values of the feature $h$ (increasing values of $h$ from bottom to top) as listed in Table 1.

**Table 1.** Details of the dataset used for the ANN training.

| # Data set ($m$) | # Training set | # Validation set | $r$ | $h$ | $\mathbb{P}e_g$ |
|---|---|---|---|---|---|
| 900 | 720 (80%) | 180 (20%) | $\{1, 2, 3\}$ | $\left\{\frac{\sqrt{2}}{10}, \frac{\sqrt{2}}{20}, \frac{\sqrt{2}}{40}\right\}$ | randomly in $[7, 70'710]$. |

We train a fully-connected feed-forward ANN on the generated dataset by using the open source library `Keras` [26] built on top of `TensorFlow` [1]. We divide the dataset into two parts: a training dataset that takes 80% of the examples to be used for the ANN training and a validation dataset that takes the remaining 20%. We choose the loss function as the mean squared error that measures, for each training feature, the squared mismatch between the prediction of the ANN $\widetilde{y}^{(j)}$ and the actual target $y^{(j)}$. Specifically, the loss function is defined as

$$\mathcal{J} = \frac{1}{2m} \sum_{j=1}^{m} \left(\widetilde{y}^{(j)} - y^{(j)}\right)^2 . \tag{5.2}$$

We normalize the features by subtracting their sample mean and dividing by their sample standard deviation in order to help the weights to better adapt to the different scales of the features. Moreover, the targets and the feature $\mathbb{P}\mathrm{e}_g$ need special care as they are distributed over a wide range of values. Thus, we normalize them using by applying a base 10 logarithm. The normalized features and targets read:

$$\widetilde{\mathbf{x}} = \left[\ \frac{r-\bar{r}}{\sigma_r},\ \frac{h-\bar{h}}{\sigma_h},\ \frac{\log_{10}(\mathbb{P}\mathrm{e}_g)-\overline{\log_{10}(\mathbb{P}\mathrm{e}_g)}}{\sigma_{\log_{10}(\mathbb{P}\mathrm{e}_g)}}\ \right], \qquad \widetilde{\mathbf{y}} = \left[\ -\log_{10}(\tau^*)\ \right],$$

where $\bar{r}$, $\bar{h}$, $\overline{\log_{10}(\mathbb{P}\mathrm{e}_g)}$ are the sample mean of the training features $r$, $h$ and the logarithm of $\mathbb{P}\mathrm{e}_g$ respectively, while $\sigma_r$, $\sigma_h$ and $\sigma_{\log_{10}(\mathbb{P}\mathrm{e}_g)}$ are their sample standard deviations.

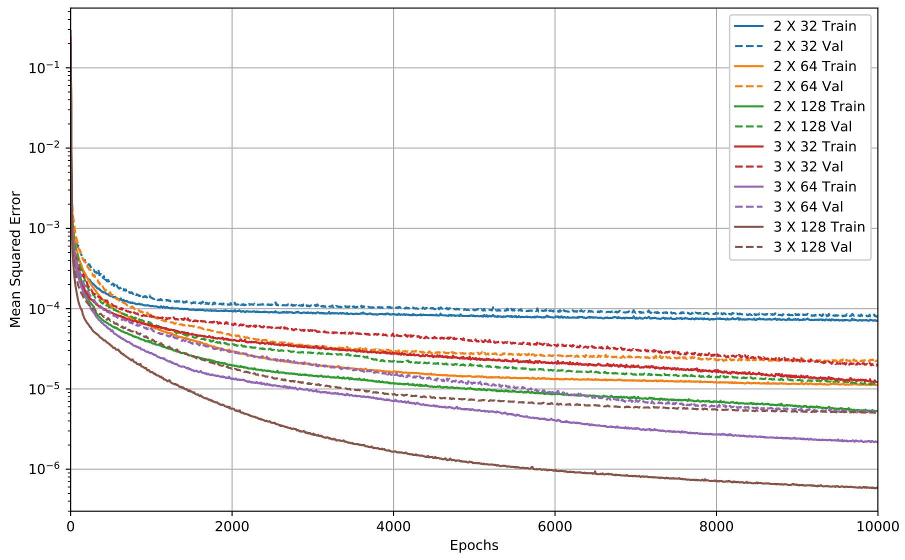


**Figure 8.** Comparison of training and validation errors over training epochs with different architectures: 2 and 3 hidden layers and 32, 64 and 128 nodes per layer.

In order to find the best performing ANN architecture, we carried out a study by testing different numbers of hidden layers, nodes per layer, optimization algorithm, its learning rate, and batch size. A comparison of the loss function with different architectures is given in Figure 8. Using 3 hidden layers is beneficial in terms of validation error drops, while using more than 64 nodes per layer does not bring to considerable advantages. Finally, we choose an ANN with 3 hidden layers, 64 nodes per layer and an output layer with a single node, all using a rectified linear unit (*ReLU*) activation function. We display the ANN architecture in Figure 9. We trained the ANN with the SGD optimization algorithm, a constant learning rate of 0.01, and mini-batch size of 32 samples. Moreover, we employed a momentum of 0.9 in the optimization algorithm to update the weights. The trained ANN is available in the `GitLab` repository [53].

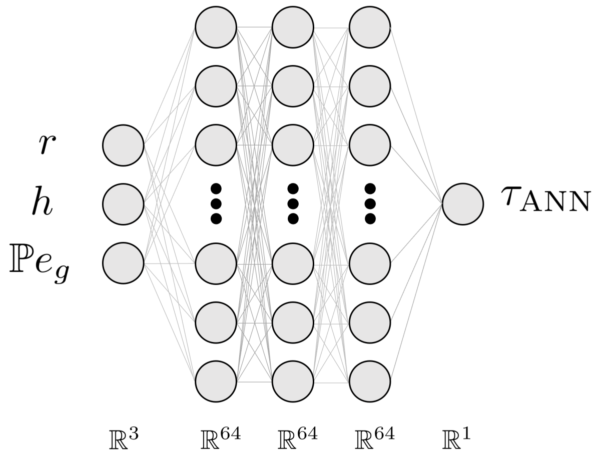


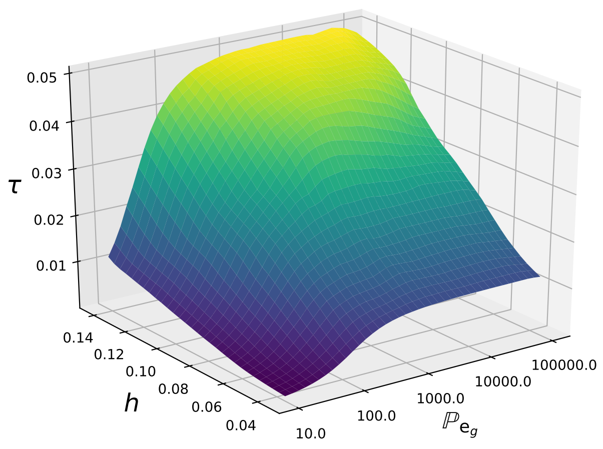
**Figure 9.** Architecture of the feed-forward fully-connected ANN.

Regarding the computational efficiency of the proposed strategy, we stress the clear distinction between the offline phase (dataset generation and ANN's training) and the online phase, where we use the ANN to predict a new stabilization parameter – alongside the use of the FE solver – for unseen input parameter values. The most demanding part of the strategy is the dataset generation, requiring approximately 2 h on a standard laptop to repeatedly solve the optimization problem 900 times. Moreover, the ANN's training phase required approximately 10'. The testing (online) phase, which is the one that is performed for application purposes, is considerably inexpensive, requiring only the real-time evaluation of a composition of linear functions, comparable with the evaluation of the empirical relation that brings to the theoretical values ( few milliseconds).

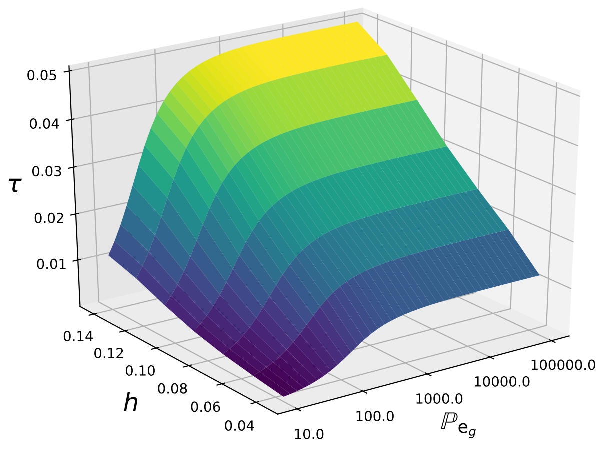
### *5.2. Predictions of the stabilization parameter by ANN*

Now, we compare the predictions of the ANN with the theoretical stabilization parameter $\widetilde{\tau}_r$ of Eq (2.8) [18,34]. We show in Figure 10 (left) the stabilization parameter $\tau_{\text{ANN}}$ predicted by the ANN by varying mesh size $h$ and global Péclet number $\mathbb{P}e_g$. For comparison, we report in Figure 10 (right) the corresponding theoretical stabilization parameter $\widetilde{\tau}_r$. We observe that the overall behavior of the trained ANN's predictions are qualitatively similar of the theoretical stabilization parameter $\widetilde{\tau}_r$. Nevertheless, it can be inferred that the values of the stabilization parameters are almost completely matched with linear FE ($r = 1$), while they quantitatively differ for $r = 2$ and $r = 3$. This was expected as $\widetilde{\tau}_r$ for $r > 1$ is an empirical extension of the formula for the case $r = 1$.

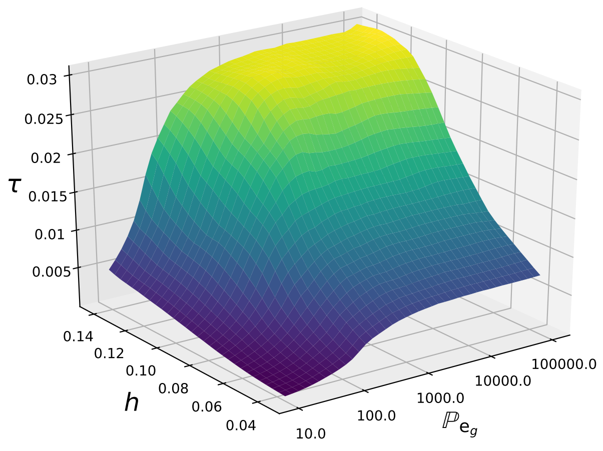
The ANN allows to make predictions with features outside the range of values for which it has been trained, that is for unseen values of such features. With this aim, we report in Figure 11 the comparison of the ANN's predictions $\tau_{\text{ANN}}$ with $\widetilde{\tau}_r$ for the FE degree $r = 4$. This comparison shows a clear difference between the theoretical and ANN's $\tau$ for $r = 4$ even though the general trend is maintained similar.



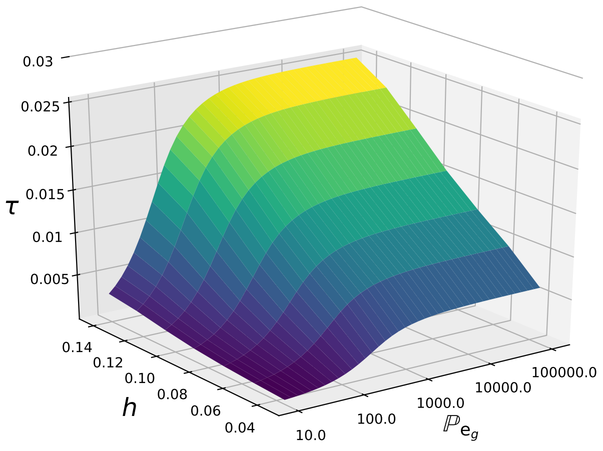
**(a)** $\tau_{\mathrm{ANN}}$ for $r = 1$



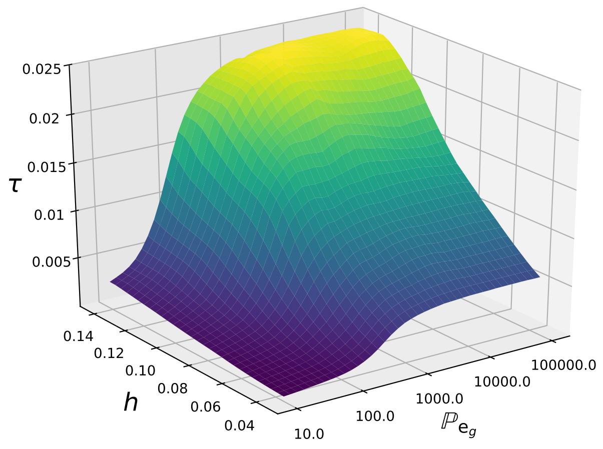
**(b)** $\widetilde{\tau}_r$ for $r = 1$



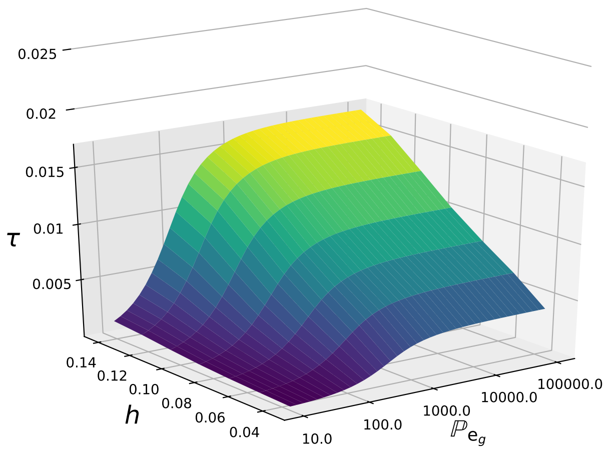
**(c)** $\tau_{\mathrm{ANN}}$ for $r = 2$



**(d)** $\widetilde{\tau}_r$ for $r = 2$

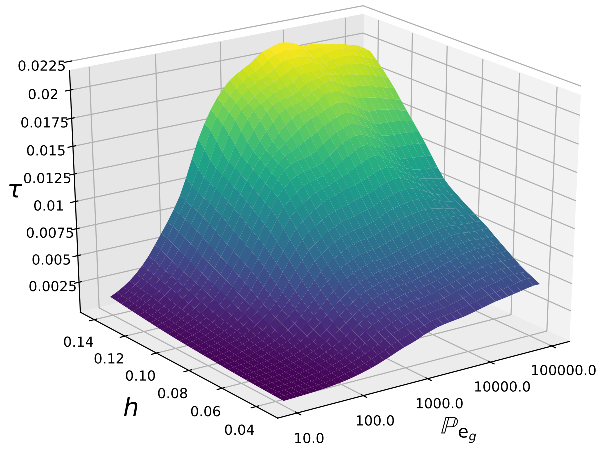


**(e)** $\tau_{\mathrm{ANN}}$ for $r = 3$

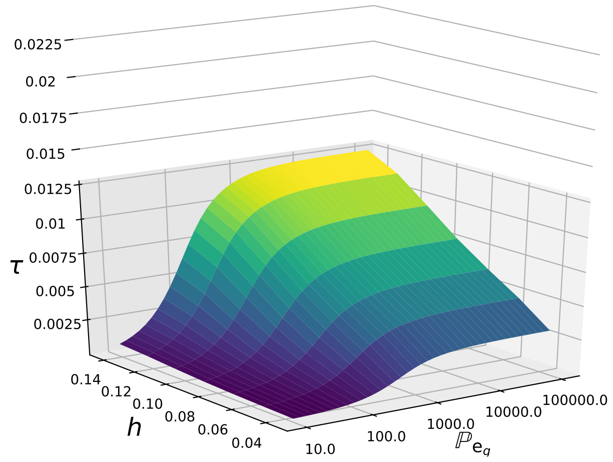


**(f)** $\widetilde{\tau}_r$ for $r = 3$

**Figure 10.** Stabilization parameter $\tau_{\mathrm{ANN}}$ predicted by the ANN and theoretical one $\widetilde{\tau}_r$ for varying $\mathbb{P}e_g$ and $h$ at different FE degrees $r = 1, 2,$ an 3 for the 2D advection-diffusion problem of Section 5.1.



**(a)** ANN's predicted $\tau$ at $r = 4$



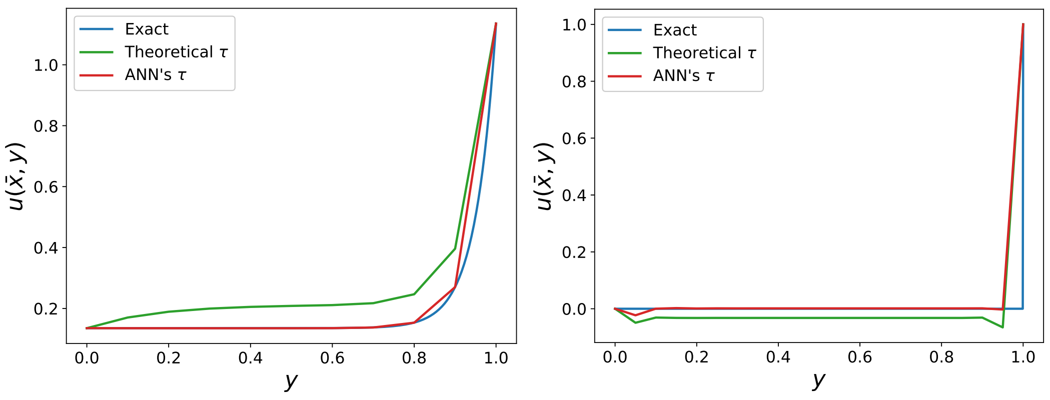
**(b)** Theoretical $\tau$ at $r = 4$

**Figure 11.** Stabilization parameter $\tau_{\text{ANN}}$ predicted by the ANN and theoretical one $\widetilde{\tau}_r$ for varying $\mathbb{P}\text{e}_g$ and $h$ with FE degree $r = 4$ for the 2D advection-diffusion problem of Section 5.1.

### 5.2.1. Test 1: predictions for the problem used in the ANN's training

We compare the numerical solutions $u_h^{\text{ANN}}$ and $\widetilde{u}_h$ obtained by means of the SUPG stabilization method with the parameter $\tau_{\text{ANN}}$ predicted by the ANN and the theoretical one $\widetilde{\tau}_r$, respectively; the comparison also involves the exact solution $u$ (5.1) of the 2D advection-diffusion problem used for the training of the ANN. In particular, we display the comparison of the former 2D solutions in Figure 12 along the line $(1-h, y)$ for any $y \in [0,1]$ with: (a) $\mathbb{P}\text{e}_h = 2$, $r = 1$, and $h = \sqrt{2}/10$ (Figure 12a); (b) $\mathbb{P}\text{e}_h = 500$, $r = 3$, and and $h = \sqrt{2}/20$ (Figure 12b). We notice that in both the cases, the numerical solution $u_h^{\text{ANN}}$ involving the stabilization parameter $\tau_{\text{ANN}}$ provides more accurate results than with the theoretical stabilization paramter $\widetilde{\tau}_r$. In particular, in the case (a), $\widetilde{u}_h$ involves a much smoother boundary layer and overshoots the exact solutions $u$, conversely to the nearly nodally exact numerical solution $u_h^{\text{ANN}}$. In the case (b), $u_h^{\text{ANN}}$ provides a much better representation of the bundary layer, without the undershooting of the solution $u$ exhibited by $\widetilde{u}_h$. Furthermore, we report in Table 2 absolute errors between the numerical solutions ($u_h^{\text{ANN}}$ and $\widetilde{u}_h$) and the exact one $u$. The errors, computed in $L^2(\Omega)$ and $H^1(\Omega)$ norms, show that the ANN-based SUPG stabilization method very often produces more accurate results than the ones obtained with theoretical stabilization parameter $\widetilde{\tau}_r$, especially for $r = 3$.

Moreover, to assess the ability of the ANN to predict the stabilization parameter out of the training range, we compare in Figure 13 the numerical solutions for high Péclet in the case $r = 4$. We observe that $u_h^{\text{ANN}}$ is more accurate than $\widetilde{u}_h$ even with FE degree out of the training range.



**(a)** $\mathbb{P}\mathrm{e}_h = 2$, $h = \sqrt{2}/10$, $r = 1$ **(b)** $\mathbb{P}\mathrm{e}_h = 500$, $h = \sqrt{2}/20$, $r = 3$

**Figure 12.** Test 1: comparison of the solutions $u$ (5.1) (blue), $\widetilde{u}_h$ (green), and $u_h^{\mathrm{ANN}}$ (red) of the 2D advection-diffusion problem along the line $(1-h, y)$ for any $y \in [0, 1]$.

**Table 2.** Test 1: comparison of the errors in norms $L^2$ and $H^1$ between the numerical solutions ($u_h^{\mathrm{ANN}}$ and $\widetilde{u}_h$) and the exact one $u$ (5.1) of the 2D advection-diffusion problem used for the training (Section 5.1) for different values of the features.

| | | | $\widetilde{\tau}_r$ $(e = \widetilde{u}_h - u)$ | | $\tau_{\mathrm{ANN}}$ $(e = u_h^{\mathrm{ANN}} - u)$ | |
|---|---|---|---|---|---|---|
| $r$ | $h$ | $\mathbb{P}\mathrm{e}_h$ | $\lVert e\rVert_{L^2(\Omega)}$ | $\lVert e\rVert_{H^1(\Omega)}$ | $\lVert e\rVert_{L^2(\Omega)}$ | $\lVert e\rVert_{H^1(\Omega)}$ |
| 1 | $\sqrt{2}/10$ | 2 | $9.56 \cdot 10^{-2}$ | $7.93 \cdot 10^{-1}$ | $6.49 \cdot 10^{-2}$ | $5.31 \cdot 10^{-1}$ |
| 2 | $\sqrt{2}/10$ | 2 | $6.48 \cdot 10^{-2}$ | $5.31 \cdot 10^{-1}$ | $6.50 \cdot 10^{-2}$ | $5.33 \cdot 10^{-1}$ |
| 3 | $\sqrt{2}/10$ | 2 | $6.50 \cdot 10^{-2}$ | $5.33 \cdot 10^{-1}$ | $6.49 \cdot 10^{-2}$ | $5.32 \cdot 10^{-1}$ |
| 1 | $\sqrt{2}/20$ | 500 | $2.99 \cdot 10^{-4}$ | $5.30 \cdot 10^{-3}$ | $3.11 \cdot 10^{-5}$ | $5.50 \cdot 10^{-4}$ |
| 2 | $\sqrt{2}/20$ | 500 | $2.71 \cdot 10^{-2}$ | $4.86 \cdot 10^{-1}$ | $2.50 \cdot 10^{-2}$ | $4.48 \cdot 10^{-1}$ |
| 3 | $\sqrt{2}/20$ | 500 | $1.05 \cdot 10^{-2}$ | $1.90 \cdot 10^{-1}$ | $1.68 \cdot 10^{-3}$ | $3.46 \cdot 10^{-2}$ |

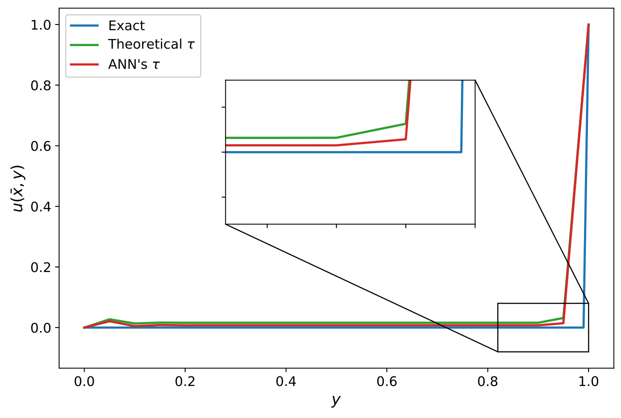


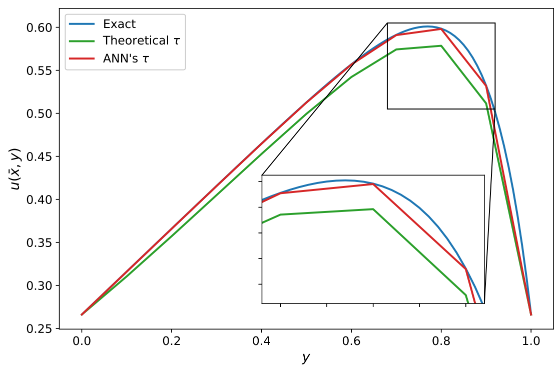
**Figure 13.** Test 1: comparison of the solutions $u$ (5.1) (blue), $\widetilde{u}_h$ (green), and $u_h^{\mathrm{ANN}}$ (red) of the 2D advection-diffusion problem with $\mathbb{P}\mathrm{e}_g = 7'071$ and $h = \sqrt{2}/20$, along the line $(1-h, y)$ for any $y \in [0, 1]$ for $r = 4$ (outside of the training range).

### 5.2.2. Test 2: predictions for an unseen problem with constant forcing term

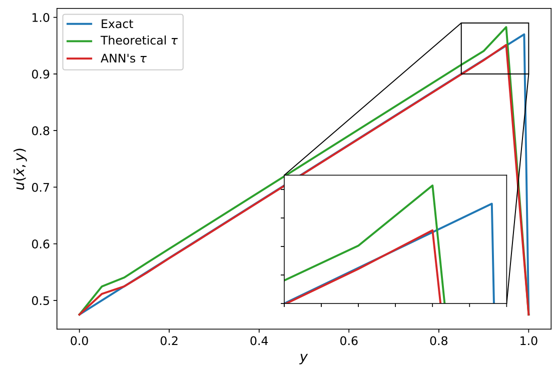
We check the model generalization of the ANN trained for the 2D advection-diffusion problem with exact solution $u$ of Eq (5.1) by predicting $\tau_{\text{ANN}}$ for an unseen 2D advection-diffusion problem. In particular, we consider the advection-diffusion problem of Eq (2.1) in $\Omega = (0,1)^2$, with $f = 1$ and $\boldsymbol{\beta} = (1,1)$. We prescribe the following exact solution on the whole boundary $\partial\Omega$:

$$u(x,y) = \frac{1}{2}(x+y) + \frac{1 - \frac{1}{2}(e^{(x/\mu)} + e^{(y/\mu)})}{e^{(1/\mu)} - 1}. \tag{5.3}$$

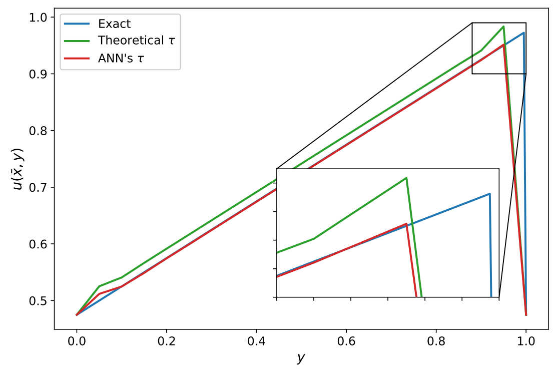
We compare in Figure 14 the numerical solutions $u_h^{\text{ANN}}$ and $\widetilde{u}_h$ with the novel exact solution $u$ for different Péclet and mesh sizes. As for the previous numerical tests, the stabilization parameter $\tau_{\text{ANN}}$ predicted by the network provides more accurate numerical solutions $u_h^{\text{ANN}}$ than with the theoretical stabilization parameter $\widetilde{\tau}_r$. In particular, the boundary layers and overall solution behaviours are better represented.



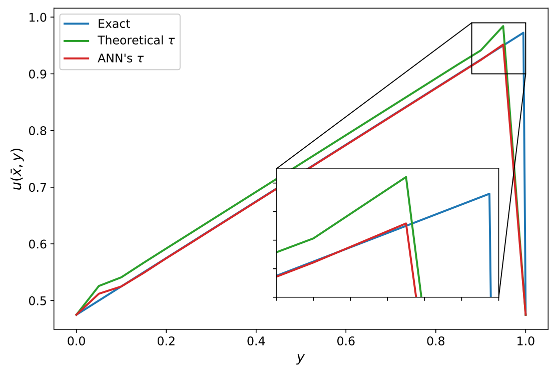
**(a)** $\mathbb{P}\text{e}_g = 7$, $h = \sqrt{2}/10$, $r = 1$



**(b)** $\mathbb{P}\text{e}_g = 7'071$, $h = \sqrt{2}/20$, $r = 3$



**(c)** $\mathbb{P}\text{e}_g = 14'142$, $h = \sqrt{2}/10$, $r = 3$

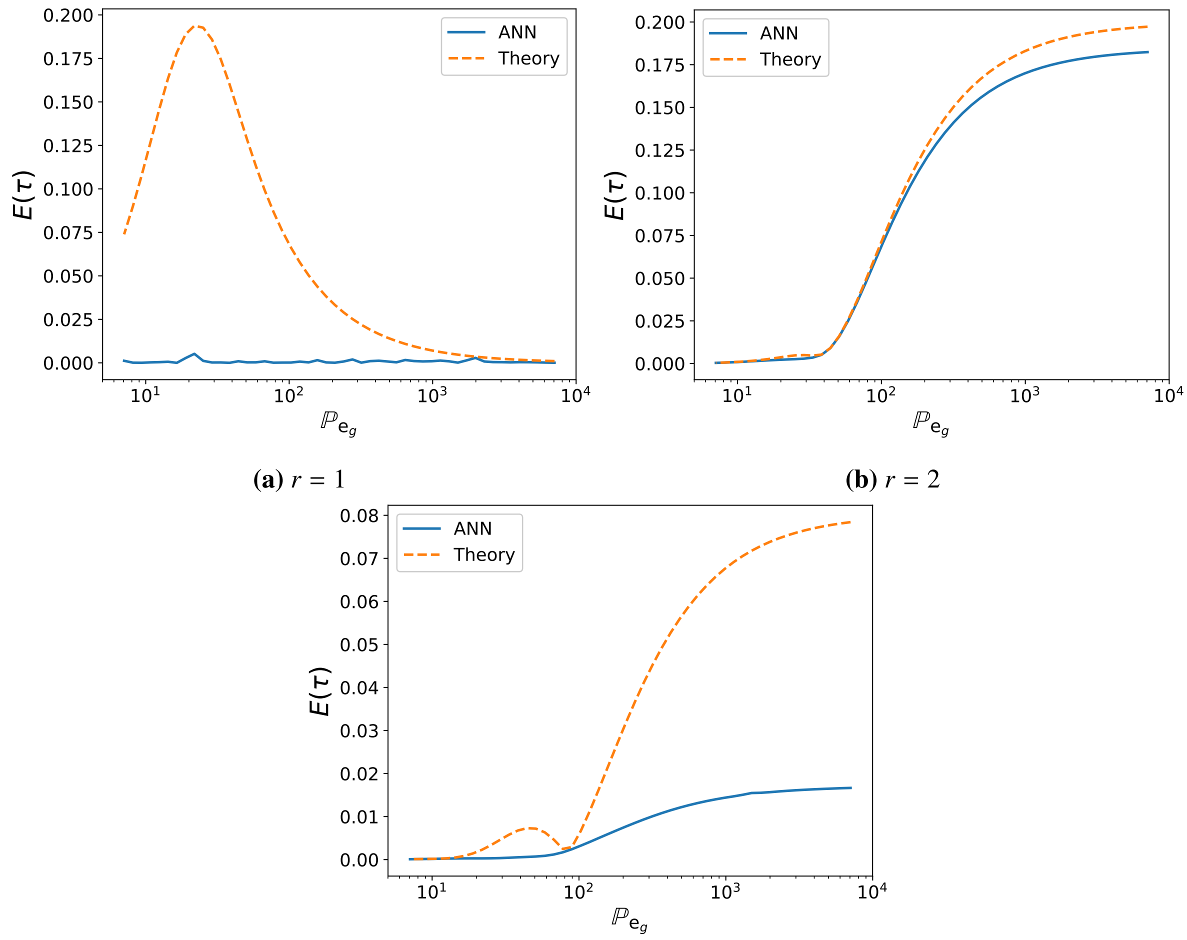


**(d)** $\mathbb{P}\text{e}_g = 70'710$, $h = \sqrt{2}/20$, $r = 3$

**Figure 14.** Test 2: Comparison of the solutions $u$ (5.3) (blue), $\widetilde{u}_h$ (green), and $u_h^{\text{ANN}}$ (red) of the *unseen* 2D advection-diffusion problem along the line $(1-h, y)$ for any $y \in [0,1]$.

We better assess the former qualitative consideration, by computing the error $E(\tau)$ associated with the numerical solutions with SUPG stabilization for the parameters $\tau_{\text{ANN}}$ and $\widetilde{\tau}_r$; different values of

$\mathbb{P}\mathrm{e}_g$ and $r$ are considered. In particular, Figure 15 compares $E(\tau)$ against $\mathbb{P}\mathrm{e}_g$ (in logarithmic scale) for different values of $r$: the error obtained by using $\tau_{\text{ANN}}$ is always lower than the one achieved with $\widetilde{\tau}_r$. These results indicate that $\tau_{\text{ANN}}$, although trained on a specific advection-diffusion problem, can be used to make predictions of the optimal $\tau$ for an unseen advection-diffusion problem in place of the theoretical stabilization parameter.

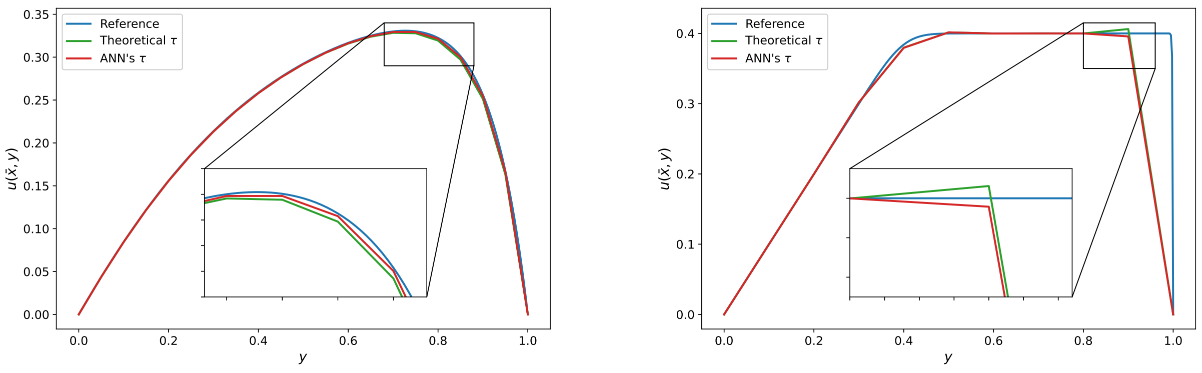


**(a)** $r = 1$ **(b)** $r = 2$ **(c)** $r = 3$

**Figure 15.** Test 2: Comparison of errors $E(\tau)$ in Eq (3.3) obtained with $\tau_{\text{ANN}}$ (blue, full line) and $\widetilde{\tau}_r$ (orange, ashed line) applied to the unseen 2D advection-diffusion problem with the exact solution $u$ (5.3); $E(\tau)$ against $\mathbb{P}\mathrm{e}_g$ with $h = \sqrt{2}/10$ and FE degrees $r = 1, 2$, and 3.

### 5.2.3. Test 3: predictions for and unseen problem with constant forcing term without exact solution

In this section, we consider an advection-diffusion problem with constant forcing term $f = 1$, advection coefficient $\boldsymbol{\beta} = (1, 1)$, and boundary conditions $u = 0$ on $\partial\Omega$. Although there is not an exact solution to the given problem, we consider as "ground truth" (reference) solution a numerical solution obtained on a much finer grid ($h = \sqrt{2}/400$) without stabilization. In Figure 16, we compare the SUPG numerical solutions $\widetilde{u}_h$, $u_h^{\text{ANN}}$ against our "ground truth" solution for two different Péclet numbers. The numerical solution with the stabilization parameter $\tau_{\text{ANN}}$ provides more accurate results with respect to the the solution obtained with the theoretical parameter. In particular, we highlight the largest discrepancies observed in the corners of the extracted solution.



**(a)** $\mathbb{P}\mathrm{e}_g = 7$, $h = \sqrt{2}/20$, $r = 1$ **(b)** $\mathbb{P}\mathrm{e}_g = 707$, $h = \sqrt{2}/10$, $r = 3$

**Figure 16.** Test 3: comparison of the solutions $u$ (blue), $\widetilde{u}_h$ (green), and $u_h^{\text{ANN}}$ (red) of an *unseen* 2D advection-diffusion problem along the line $(0.5, y)$ for any $y \in [0, 1]$.

### 5.2.4. Test 4: prediction for an unseen problem with a non-constant forcing term

In this section, we cope the case of a non costant forcing term by considering advection-diffusion problem of Eq (2.1) in $\Omega = (0, 1)^2$ with $\boldsymbol{\beta} = (1, 1)$, and we prescribe the following exact solution on the whole boundary $\partial\Omega$:

$$u(x, y) = -\frac{\operatorname{atan}\Big((x - 1/2)^2 + (y - 1/2)^2 - 1/16\Big)}{\sqrt{\mu}}. \tag{5.4}$$

We display the exact solution of the considered problem in Figure 17.

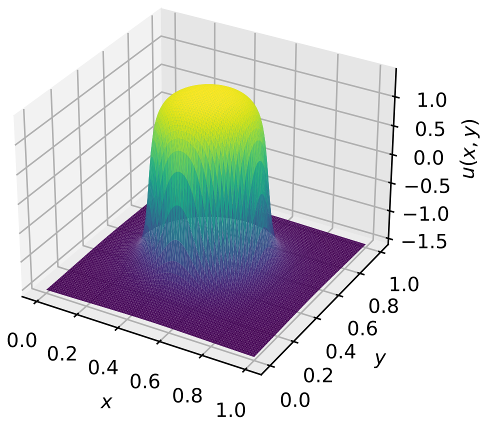
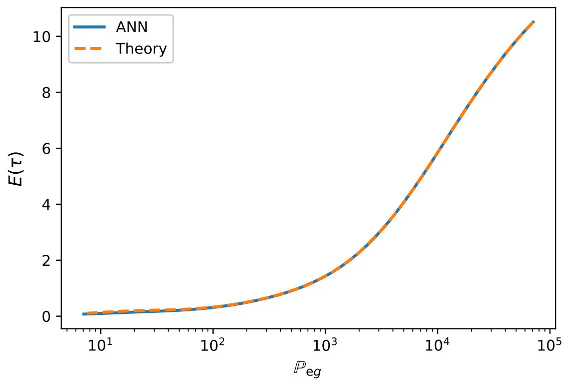
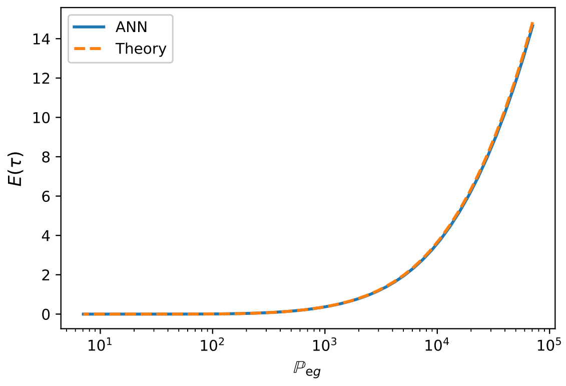


**Figure 17.** Test 4: exact solution in Eq (5.4) of the relative advection-diffusion problem.

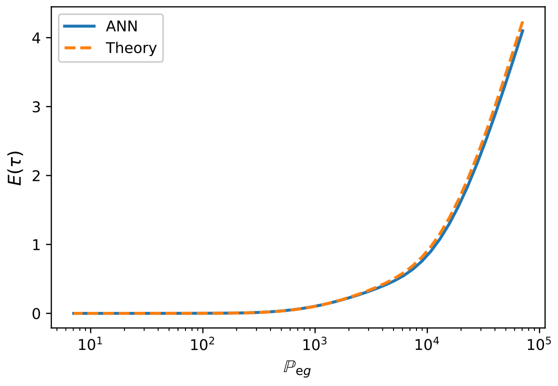
In particular, Figures 18 and 19 show $E(\tau)$ against $\mathbb{P}\mathrm{e}_g$ for different values of $r$ with two different meshes with $h = \sqrt{2}/10$ and $h = \sqrt{2}/20$, respectively. The error obtained by using $\tau_{\mathrm{ANN}}$ is always comparable to the one achieved with $\widetilde{\tau}_r$, for both mesh levels and for all the FE degrees considered. These results suggest that $\tau_{\mathrm{ANN}}$, although trained on a specific advection-diffusion problem with $f = 0$, is robust with respect to unseen parameters and data in the advection-diffusion problem, including non constant $f$. Nevertheless, further studies can be conducted to better assess the role of a non-constant forcing term into the ANN's training phase, possibly encompassing the accuracy of the theoretical stabilization parameter in this scenario too.



**(a)** $r = 1$

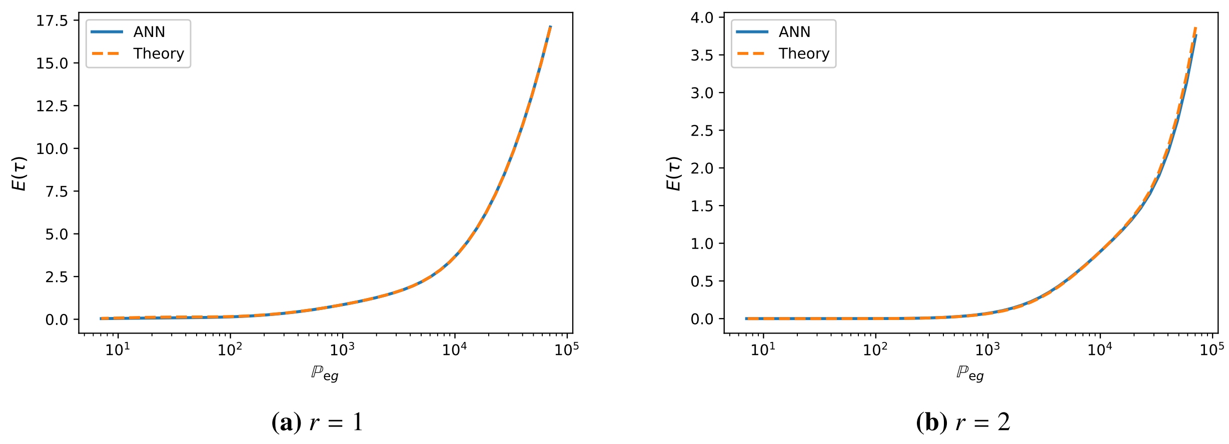


**(b)** $r = 2$



**(c)** $r = 3$

**Figure 18.** Test 4: comparison of errors $E(\tau)$ in Eq (3.3) obtained with $\tau_{\mathrm{ANN}}$ (blue, full line) and $\widetilde{\tau}_r$ (orange, ashed line) applied to the unseen 2D advection-diffusion problem with the exact solution $u$ (5.4); $E(\tau)$ against $\mathbb{P}\mathrm{e}_g$ with $h = \sqrt{2}/10$ and FE degrees $r = 1, 2$, and 3.



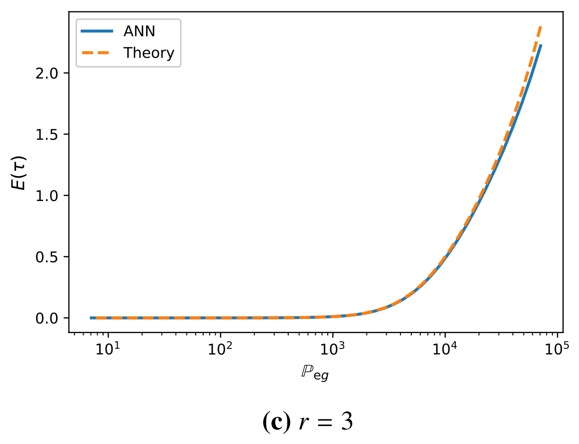


**Figure 19.** Test 4: comparison of errors $E(\tau)$ in Eq (3.3) obtained with $\tau_{\text{ANN}}$ (blue, full line) and $\widetilde{\tau}_r$ (orange, ashed line) applied to the unseen 2D advection-diffusion problem with the exact solution $u$ (5.4); $E(\tau)$ against $\mathbb{P}\text{e}_g$ with $h = \sqrt{2}/20$ and FE degrees $r = 1, 2$, and 3.

## 6. Direction of the advection field and stabilization parameters

We investigate the effect of the advection direction on the value of the stabilization parameter. We define $\theta$ as the angle between the advection velocity and the $x$-axis: $\boldsymbol{\beta} = (|\boldsymbol{\beta}| \cos\theta, \ |\boldsymbol{\beta}| \sin\theta)^T$. We carry out the optimization strategy introduced in Section 3 including also $\theta$ in the input parameter $\mathbf{x}^{(i)}$. We apply the optimization to the advection-diffusion problem with exact solution in Eq (5.3) varying $\theta$ in $[\pi/12, \pi/2]$ and we report a comparison between the optimal, theoretical and ANN's stabilization parameter in Figure 20 (left). We set $|\boldsymbol{\beta}| = \sqrt{2}$, $\mathbb{P}\text{e}_g = 7'071$, $h = \sqrt{2}/20$, $r = 3$. We recall that, differently from the optimization strategy, in this plot we are still employing the ANN trained without accounting for the advection direction. We can observe that the optimal $\tau$ is affected by $\theta$, a feature that is not accounted by the theoretical stabilization parameter. Furthermore, in Figure 20 (right), we compare the mean errors obtained with the optimal, theoretical and ANN's stabilization parameter. It can be observed that the ANN still provides an advantage with respect to the theoretical stabilization. However, we believe that by including the advection direction as an additional feature in the training phase of the ANN, a higher accuracy can be obtained that the one achieved in this paper, where a single direction ($\theta = \frac{\pi}{4}$) has been considered.

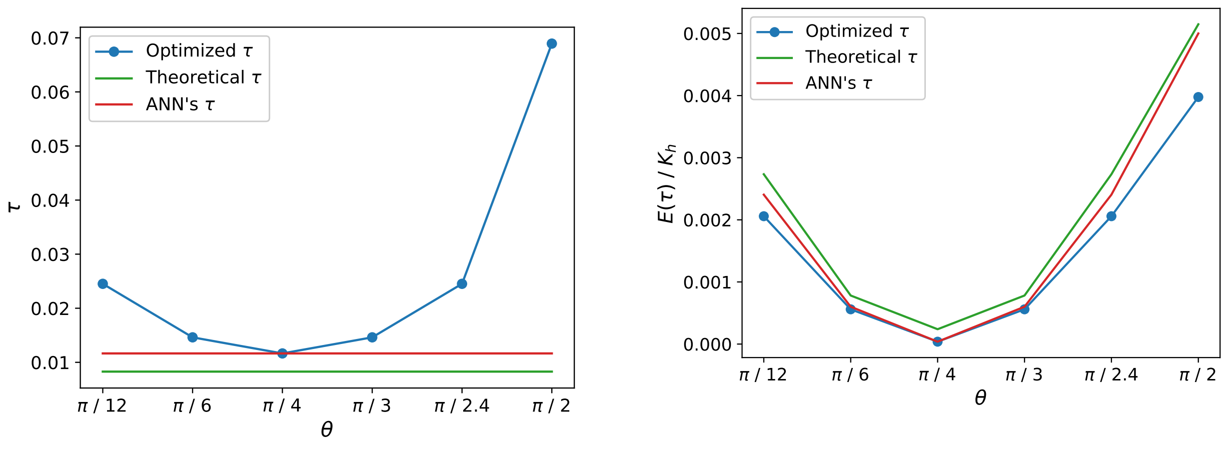


**Figure 20.** Comparison of optimal, theoretical and ANN's predicted stabilization parameters and their relative errors obtained for varying values of the advection angle $\theta$ for advection-diffusion problem with exact solution in Eq (5.3) with $\mathbb{P}e_g = 7'071$, $h = \sqrt{2}/20$, $r = 3$.

## 7. Conclusions

In this work, we presented an approach based on machine learning and ANN to compute the optimal stabilization parameter to be used in the SUPG FE approximation of advection-diffusion problems. Indeed, albeit the expression of the stabilization parameter is available for 1D problems and FE of degree $r = 1$, its extension to more general advection-diffusion problems (2D and 3D) and FE degrees $r > 1$ is still lacking.

We validated our approach against the 1D case, for which the ANN-stabilization parameter matches the already optimal, theoretical one for $r = 1$, leading to nodally exact numerical solutions Instead, for higher polynomials degrees $r > 1$, remarkable differences are observed among the theoretical and optimal stabilization parameter: we observed better accuracy and stabilization properties of the numerical solution with the optimal stabilization parameter with respect to the theoretical one.

Then, we generated the dataset on a 2D advection-diffusion problem, and we used it to train a fully-connected feed-forward ANN. We applied the predictions of the network on the original advection-diffusion problem for unseen input values and we also apply it to an unseen advection-diffusion problem to check for model generalization. Our numerical results showed that the proposed ANN-based approach provides more accurate numerical solutions than using the theoretical stabilization parameter for the SUPG method.

This work represents a step towards the enhancements of stabilization methods for the FE approximation of advection-dominated differential problems. In particular, this work is limited to few features for the ANN-based stabilization parameter and provides as output a single optimal stabilization parameter meant for the whole mesh FE $\mathcal{T}_h$. Natural extensions of this work will therefore involve local stabilization parameters, i.e., element by element over the mesh $\mathcal{T}_h$, to better account for non-uniform meshes, differential problems with varying coefficients, and capturing the local behaviour of the solution. In addition, to enhance the robustness of the ANN-based approach, possible additional inputs of the network are the angle of the advection velocity and the nodal value of the forcing term. A future development consists in the extension of the proposed approach to the 3D case for which, as for the 2D case, a universal definition of the stabilization parameter does not exist. The extension to the 3D case surely would present a larger cost in the training phase; however, once trained, the ANN could be used online real-time.

Finally, we envision extending the proposed machine learning approach to SUPG and variational multiscale stabilization methods for the FE approximation of the Navier-Stokes equations.

## Acknowledgments

The authors acknowledge Prof. C. Canuto, Politecnico di Torino, for the fruitful discussions and useful suggestions. L. D. and A. Z. are members of the INdAM Research group GNCS. L. D. has been partially funded by the research project PRIN 2020, n.20204LN5N5, by MIUR. The authors ackwnowledge the anonymous reviewers for their insightful comments and suggestions.

## Conflict of interest

The authors declare no conflict of interest.

## References

1. M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro. et al., TensorFlow: Large-scale machine learning on heterogeneous systems, arXiv:1603.04467.
2. M. S. Alnæs, J. Blechta, J. Hake, A. Johansson, B. Kehlet, A. Logg, et al., The fenics project version 1.5, *Archive of Numerical Software*, **3** (2015), 9–23. https://doi.org/10.11588/ans.2015.100.20553
3. Y. Bazilevs, V. M. Calo, J. A. Cottrell, T. J. R. Hughes, A. Reali, G. Scovazzi, Variational multiscale residual-based turbulence modeling for large eddy simulation of incompressible flows, *Comput. Method. Appl. Mech. Eng.*, **197** (2007), 173–201. https://doi.org/10.1016/j.cma.2007.07.016
4. N. Bénard, J. Pons-Prats, J. Périaux, G. Bugeda, P. Braud, J. P. Bonnet, et al., Turbulent separated shear flow control by surface plasma actuator: experimental optimization by genetic algorithm approach, *Exp. Fluids*, **57** (2016), 22. https://doi.org/10.1007/s00348-015-2107-3
5. P. B. Bochev, C. R. Dohrmann, M. D. Gunzburger, Stabilization of low-order mixed finite elements for the Stokes equations, *SIAM J. Numer. Anal.*, **44** (2016), 82–101. https://doi.org/10.1137/S0036142905444482
6. 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, *Comput. Method. Appl. Mech. Eng.*, **32** (1982), 199–259. https://doi.org/10.1016/0045-7825(82)90071-8
7. C. Canuto, M. Y. Hussaini, A. Quarteroni, T. A. Zang, *Spectral methods. Fundamentals in single domains*, Berlin, Heidelberg: Springer, 2006.

8. R. Codina, On stabilized finite element methods for linear systems of convection-diffusion-reaction equations, *Comput. Method. Appl. Mech. Eng.*, **188** (2000), 61–82. https://doi.org/10.1016/S0045-7825(00)00177-8

9. R. Codina, Analysis of a stabilized finite element approximation of the Oseen equations using orthogonal subscales, *Appl. Numer. Math.*, **58** (2008), 264–283. https://doi.org/10.1016/j.apnum.2006.11.011

10. R. Codina, J. Principe, O. Guasch, S. Badia, Time dependent subscales in the stabilized finite element approximation of incompressible flow problems, *Comput. Method. Appl. Mech. Eng.*, **196** (2007), 2413–2430. https://doi.org/10.1016/j.cma.2007.01.002

11. B. Colvert, M. Alsalman, E. Kanso, Classifying vortex wakes using neural networks, *Bioinspir. Biomim.*, **13** (2018), 025003. https://doi.org/10.1088/1748-3190/aaa787

12. J. A. Cottrell, T. J. R. Hughes, Y. Bazilevs, *Isogeometric analysis: Toward integration of CAD and FEA*, John Wiley & Sons, 2009. https://doi.org/10.1002/9780470749081

13. N. Discacciati, J. S. Hesthaven, D. Ray, Controlling oscillations in high-order discontinuous Galerkin schemes using artificial viscosity tuned by neural networks, *J. Comput. Phys.*, **409** (2020), 109304. https://doi.org/10.1016/j.jcp.2020.109304

14. K. Duraisamy, G. Iaccarino, H. Xiao, Turbulence modeling in the age of data, *Annu. Rev. Fluid Mech.*, **51** (2019), 357–377. https://doi.org/10.1146/annurev-fluid-010518-040547

15. D. Forti, L. Dede', Semi-implicit BDF time discretization of the Navier–Stokes equations with VMS-LES modeling in a high performance computing framework, *Comput. Fluids*, **117** (2015), 168–182. https://doi.org/10.1016/j.compfluid.2015.05.011

16. L. P. Franca, S. L. Frey, T. J. R. Hughes, Stabilized finite element methods: I. application to the advective-diffusive model, *Comput. Method. Appl. Mech. Eng.*, **95** (1992), 253–276. https://doi.org/10.1016/0045-7825(92)90143-8

17. S. Fresca, L. Dede', A. Manzoni, A comprehensive deep learning-based approach to reduced order modeling of nonlinear time-dependent parametrized PDEs, *J. Sci. Comput.*, **87** (2021), 61. https://doi.org/10.1007/s10915-021-01462-7

18. A. C. Galeao, R. C. Almeida, S. M. C. Malta, A. F. D. Loula, Finite element analysis of convection dominated reaction–diffusion problems, *Appl. Numer. Math.*, **48** (2004), 205–222. https://doi.org/10.1016/j.apnum.2003.10.002

19. I. Goodfellow, Y. Bengio, A. Courville, *Deep learning*, MIT Press, 2016.

20. M. Guo, J. S. Hesthaven, Data-driven reduced order modeling for time-dependent problems, *Comput. Method. Appl. Mech. Eng.*, **345** (2019), 75–99. https://doi.org/10.1016/j.cma.2018.10.029

21. J. S. Hesthaven, S. Ubbiali, Non-intrusive reduced order modeling of nonlinear problems using neural networks, *J. Comput. Phys.*, **363** (2018), 55–78. https://doi.org/10.1016/j.jcp.2018.02.037

22. T. J. R. Hughes, *The finite element method: linear static and dynamic finite element analysis*, Courier Corporation, 2012.

23. M. Janssens, S. J. Hulshoff, Advancing artificial neural network parameterisation for atmospheric turbulence using a variational multiscale model, *J. Adv. Model. Earth Syst.*, **14** (2022), e2021MS002490. https://doi.org/10.1029/2021MS002490

24. M. Janssens, Machine learning of atmospheric turbulence in a variational multiscale model, 2019. Available from: `http://resolver.tudelft.nl/uuid:bd090309-305e-4c04-93b7-64f1b79df8d4`.

25. V. John, P. Knobloch, On spurious oscillations at layers diminishing (sold) methods for convection–diffusion equations: Part I–A review, *Comput. Method. Appl. Mech. Eng.*, **196** (2007), 2197–2215. https://doi.org/10.1016/j.cma.2006.11.013

26. Keras. Available from: `https://keras.io`.

27. G. Kutyniok, P. Petersen, M. Raslan, R. Schneider, A theoretical analysis of deep neural networks and parametric PDEs, *Constr. Approx.*, **55** (2021), 73–125. https://doi.org/10.1007/s00365-021-09551-4

28. M. Milano, P. Koumoutsakos, Neural network modeling for near wall turbulent flow, *J. Comput. Phys.*, **182** (2002), 1–26. https://doi.org/10.1006/jcph.2002.7146

29. S. Mishra, A machine learning framework for data driven acceleration of computations of differential equations, *Mathematics in Engineering*, **1** (2019), 118–146. https://doi.org/10.3934/Mine.2018.1.118

30. T. M. Mitchell, *Machine learning*, New York: McGraw-hill, 1997.

31. P. Neittaanmaki, S. Repin, Artificial intelligence and computational science, In: *Computational sciences and artificial intelligence in industry*, **76** (2022), 27–35. https://doi.org/10.1007/978-3-030-70787-3_3

32. G. Novati, L. Mahadevan, P. Koumoutsakos, Controlled gliding and perching through deep-reinforcement-learning, *Phys. Rev. Fluids*, **4** (2019), 093902. https://doi.org/10.1103/PhysRevFluids.4.093902

33. A. Quarteroni, A. Valli, *Numerical approximation of partial differential equations*, Berlin, Heidelberg: Springer, 1994. https://doi.org/10.1007/978-3-540-85268-1

34. A. Quarteroni, *Numerical models for differential problems*, 3 Eds., Cham: Springer, 2017. https://doi.org/10.1007/978-3-319-49316-9

35. M. Raissi, G. E. Karniadakis, Hidden physics models: Machine learning of nonlinear partial differential equations, *J. Comput. Phys.*, **357** (2018), 125–141. https://doi.org/10.1016/j.jcp.2017.11.039

36. M. Raissi, P. Perdikaris, G. E. Karniadakis, Machine learning of linear differential equations using Gaussian processes, *J. Comput. Phys.*, **348** (2017), 683–693. https://doi.org/10.1016/j.jcp.2017.07.050

37. 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, *J. Comput. Phys.*, **378** (2019), 686–707. https://doi.org/10.1016/j.jcp.2018.10.045

38. T. C. Rebollo, B. M. Dia, A variational multi-scale method with spectral approximation of the sub-scales: Application to the 1D advection–diffusion equations, *Comput. Method. Appl. Mech. Eng.*, **285** (2015), 406–426. https://doi.org/10.1016/j.cma.2014.11.025

39. F. Regazzoni, L. Dede', A. Quarteroni, Machine learning of multiscale active force generation models for the efficient simulation of cardiac electromechanics, *Comput. Method. Appl. Mech. Eng.*, **370** (2020), 113268. https://doi.org/10.1016/j.cma.2020.113268

40. F. Regazzoni, L. Dede', A. Quarteroni, Machine learning for fast and reliable solution of time-dependent differential equations, *J. Comput. Phys.*, **397** (2019), 108852. https://doi.org/10.1016/j.jcp.2019.07.050

41. H. G. Roos, M. Stynes, L. Tobiska, *Numerical methods for singularly perturbed differential equations*, Berlin, Heidelberg: Springer, 1996. https://doi.org/10.1007/978-3-662-03206-0

42. C. Schwab, *p- and hp-Finite element methods: Theory and application to solid and fluid mechanics*, Oxford University Press, 1998.

43. G. Scovazzi, M. A. Christon, T. J. R. Hughes, J. N. Shadid, Stabilized shock hydrodynamics: I. A Lagrangian method, *Comput. Method. Appl. Mech. Eng.*, **196** (2007), 923–966. https://doi.org/10.1016/j.cma.2006.08.008

44. G. Scovazzi, B. Carnes, X. Zeng, S. Rossi, A simple, stable, and accurate linear tetrahedral finite element for transient, nearly, and fully incompressible solid dynamics: a dynamic variational multiscale approach, *Int. J. Numer. Meth. Eng.*, **106** (2016), 799–839. https://doi.org/10.1002/nme.5138

45. T. E. Tezduyar, Y. Osawa, Finite element stabilization parameters computed from element matrices and vectors, *Comput. Method. Appl. Mech. Eng.*, **190** (2000), 411–430. https://doi.org/10.1016/S0045-7825(00)00211-5

46. University of Illinois at Urbana-Champaign. Center for Supercomputing Research, Development, and G Cybenko, Continuous valued neural networks with two hidden layers are sufficient, 1988. Available from: `https://searchworks.stanford.edu/view/4620277`.

47. P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, et al., SciPy 1.0: fundamental algorithms for scientific computing in Python, *Nat. Methods*, **17** (2020), 261–272. https://doi.org/10.1038/s41592-019-0686-2

48. C. Xie, J. Wang, K. Li, C. Ma, Artificial neural network approach to large-eddy simulation of compressible isotropic turbulence, *Phys. Rev. E*, **99** (2019), 053113. https://doi.org/10.1103/PhysRevE.99.053113

49. C. Xie, J. Wang, W. E, Modeling subgrid-scale forces by spatial artificial neural networks in large eddy simulation of turbulence, *Phys. Rev. Fluids*, **5** (2020), 054606. https://doi.org/10.1103/PhysRevFluids.5.054606

50. B. Yegnanarayana, *Artificial neural networks*, PHI Learning Pvt. Ltd., 2009.

51. M. Zancanaro, M. Mrosek, G. Stabile, C. Othmer, G. Rozza, Hybrid neural network reduced order modelling for turbulent flows with geometric parameters, *Fluids*, **6** (2021), 296. https://doi.org/10.3390/fluids6080296

52. Z. Zhou, G. He, S. Wang, G. Jin, Subgrid-scale model for large-eddy simulation of isotropic turbulent flows using an artificial neural network, *Comput. Fluids*, **195** (2019), 104319. https://doi.org/10.1016/j.compfluid.2019.104319

53. A. Zingaro, ANN-SUPG, Project ID: 30854063, GitLab repository. Available from: `https://gitlab.com/albertozingaro/ann-supg`.

54. A. Zingaro, F. Menghini, L. Dede', A. Quarteroni, Hemodynamics of the heart's left atrium based on a Variational Multiscale-LES numerical method, *Eur. J. Mech. B/Fluids*, **89** (2021), 380–400. https://doi.org/10.1016/j.euromechflu.2021.06.014