Fast solver for advection dominated diffusion using residual minimization and neural networks

Tomasz Służalec[1][0000−0001−6217−4274] and Maciej Paszyński[2][0000−0001−7766−6052]

[1] Jagiellonian University, Kraków, Poland
[2]AGH University of Science and Technology, Kraków, Poland

Abstract. Advection-dominated diffusion is a challenging computational problem that requires special stabilization efforts. Unfortunately, the numerical solution obtained with the commonly used Galerkin method delivers unexpected oscillation resulting in an inaccurate numerical solution. The theoretical background resulting from the famous inf-sup condition tells us that the finite-dimensional test space employed by the Galerkin method does not allow us to reach the supremum necessary for problem stability. We enlarge the test space to overcome this problem. We do it for a fixed trial space. The method that allows us to do so is the residual minimization method. This method, however, requires the solution to a much larger system of linear equations than the standard Galerkin method. We represent the larger test space by its set of optimal test functions, forming a basis of the same dimension as the trial space in the Galerkin method. The resulting Petrov-Galerkin method stabilizes our challenging advection-dominated problem. We train the optimal test functions offline with the neural network to speed up the computations. We also observe that the optimal test functions, usually global, can be approximated with local support functions, resulting in a low computational cost for the solver and a stable numerical solution.

1 Introduction

The neural networks can be introduced as a non-linear function

$$y_{out} = ANN(x_{in}) = \theta_n \sigma(...\sigma(\theta_2 \sigma(\theta_1 x_{in} + \phi_1) + \phi_2) + ...) + \phi_n, \qquad (1)$$

where $\{\theta_j\}_{j=1...n}$ are matrices, possibly with different numbers of rows and columns, and $\{\phi_j\}_{j=1...n}$ are bias vectors, possibly with a different number of rows. The selected architecture of the neural network results in a different number and dimensions of matrices and bias vectors. The classical choice for the activation function is the sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$, besides several other possibilities (rectified linear unit (ReLU) or leaky ReLU [2,20]).

Recently, in computational science, the classical finite element methods are often augmented with the neural networks [3,8,15,21]. An overview of the application of the Deep Neural Networks into finite element method is presented in [8].

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36021-3_52

Paper [15] considers the problem of representing some classes of real-valued univariant approximators with Deep Neural Networks based on the rectified linear unit (ReLU) activation functions. The space generated by the neural networks with ReLU activation functions contains the space of the linear finite element method. The authors claim that the convergence rate of the DNN to the solution is of a similar order to the convergence rate of the classical finite element method. By taking $(ReLU)^k$ this result can be generated to higher-order finite element method as well [21]. The Deep Neural Networks can also help guide refinements in goal-oriented adaptivity [3].

The finite element method utilizes high-order basis functions, e.g., Lagrange polynomials which are C^0 between finite elements [5,6], or B-spline basis functions, which can be of higher order and continuity [1,4,9].

The simulations of difficult, unstable time-dependent problems, like advection-dominated diffusion [12,13], high-Reynolds number Navier-Stokes equations [11], or high-contrast material Maxwell equations, have several important applications in science and engineering. These challenging engineering problems require special stabilization methods, such as Streamline-Petrov Upwind Galerkin method (SUPG) [10], discontinuous Galerkin method (DG) [14,17], as well as residual minimization (RM) method [11–13].

2 Methodology

2.1 Galerkin method

Let us introduce an advection-diffusion problem in the strong form: Find $u \in C^2(0,1)$ such that

$$-\epsilon \frac{d^2u(x)}{dx^2} + 1\frac{du(x)}{dx} = 0, \quad x \in (0,1). \tag{2}$$

The advection part "wind" coefficient is given as 1, and the ϵ represents a diffusion coefficient. We recall that the problem is numerically unstable for small values of ϵ; thus solution cannot be correctly approximated by the classical Galerkin method without a very large computational mesh. The weak formation, suitable for the Galerkin method, is the following:

$$\int_0^1 -\epsilon \frac{d^2u(x)}{dx^2} v(x)dx + \int_0^1 1\frac{du(x)}{dx} v(x)dx = 0, \quad \forall v \in V. \tag{3}$$

We apply a Cauchy boundary condition

$$-\epsilon \frac{du}{dx}(0) + u(0) = 1.0, \; u(1) = 0, \tag{4}$$

and we rewrite the formula in a simpler form

$$-\epsilon(u'', v)_0 + (u', v)_0 = 0, \quad \forall v \in V. \tag{5}$$

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36021-3_52

We integrate by parts to get

$$\epsilon(u', v')_0 + (u', v)_0 + u(0)v(0) = v(0) \ \forall v \in V. \tag{6}$$

Now, we select a finite set of test functions, and we formulate a discrete form as: Find $u_h \in U_h, u_h = \sum_{i=1,\dots,21} u_i B_i(x)$:

$$b(u_h, v_h) = l(v_h) \quad \forall v_h \in V_h \tag{7}$$
$$b(u_h, v_h) = \epsilon\,(u_h', v_h')_0 + (u_h', v_h)_0 + u_h(0)v_h(0) \tag{8}$$
$$l(v_h) = v_h(0). \tag{9}$$

In the discrete formulation, our equation is "averaged" by the test functions $v(x)$. This is why the accuracy of the numerical solution u_h depends on the quality of the test space V_h. In the classical Galerkin method, we seek a solution as a linear combination of basis functions. In the Galerkin method, the trial space, where we seek the solution, is equal to the test space. We choose $U_h = V_h$; thus, v_h are the same 21 basis functions that are used to approximate the solution u_h. For example, we choose quadratic B-splines with C^0 separators as they are equivalent to the Lagrange basis functions.

2.2 Petrov-Galerkin method

More proper results numerical results give Petrov-Galerkin method, where we choose a different basis for a solution and a different basis for testing. Still, our solution is a linear combination of basis functions, for example, linear B-splines. We test with another basis, for example, quadratic B-splines with C^0 separators. The Petrov-Galerkin formulation is similar to the Galerkin method, but the trail space is different than the test space: Find $u_h \in U_h, u_h = \sum_{i=1,\dots,11} u_i B_i(x)$ such that:

$$b(u_h, v_h) = l(v_h) \quad \forall v_h \in \hat{V}_h \tag{10}$$
$$b(u_h, v_h) = \epsilon\,(u_h', v_h')_0 + (u_h', v_h)_0 + u_h(0)v_h(0) \tag{11}$$
$$l(v_h) = v_h(0). \tag{12}$$

where v_h contains carefully selected 11 elements of V_h. These test functions are linearly independent, and they are linear combination of the original basis functions from V_h.

$$\hat{V}_h \ni v_i = \{\alpha_1^i w_1 + \dots + \alpha_{21}^i w_{21} | w_j \in V_h, \alpha_j^i \in \mathbb{R}\},\ i = 1, \dots, 11 \tag{13}$$

Here V_h is the space of all 21 test functions, and $\hat{V}_h$ are sub-space defined by selected 11 basis functions:

$$11 = \dim U_h = \dim \hat{V}_h \leq \dim V_h = 21. \tag{14}$$

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36021-3_52

2.3 Advection-diffusion problem with arbitrary coefficients

When we simulate real-life applications with the advection-diffusion equations, the advection and diffusion coefficients can change from one time step to another. We generalize our problem into arbitrary ϵ coefficient as then it would be more general and could be used in real-life simulation.

For a given $\epsilon \in \mathbb{R}$ find $u_h \in U_h$ such that:

$$\begin{aligned} \hat{b}(\epsilon, u_h, v_h) &= l(v_h) \quad \forall v_h \in \hat{V}_h \\ \hat{b}(\epsilon, u_h, v_h) &= \epsilon\,(u_h', v_h')_0 + (u_h', v_h)_0 + u_h(0)v_h(0). \end{aligned} \tag{15}$$

The problem is difficult to solve since it requires proper space of the optimal test functions $\hat{V}_h$ for stabilization with the Petrov-Galerkin method. The optimal test functions v_h can be different for every ϵ. The problem remains numerically unstable for small ϵ, and we want the optimal test functions to be quickly adapted for a new problem for different ϵ values.

2.4 Theoretical U_h V_h space implications and limitations

To find the optimal test functions we recall that $b(\cdot,\cdot)$ satisfies the following inequality

$$\alpha\|u\|^2 \leq b(u,u) \leq M\|u\|^2, \tag{16}$$

where M is the continuity constant $b(u,v) \leq M\|u\|\|v\|$ and α is coercivity constant $b(u,u) \geq \alpha\|u\|^2$. Since $\alpha \leq M$, the constant is always $\frac{M}{\alpha} \geq 1$.

In the ideal case, we would like to have the approximation error resulting from the Galerkin solution u_h equal to the distance of the trial space from the exact solution.

Cea Lemma *[Céa, Jean (1964). Approximation variationnelle des problèmes aux limites (Ph.D. thesis)]*

$$||u - u_h|| \leq \frac{M}{\alpha}\text{dist}\{U_h, u\}. \tag{17}$$

Unfortunately, this is only the case if $\frac{M}{\alpha} = 1$. In reality, $\frac{M}{\alpha} \geq 1$, and the best solution in the approximation space can be worse than the distance of the space to the exact solution.

The coercivity constant can be estimated from the following:

Babuška theorem (inf-sup condition) *[Babuška, Ivo (1970). "Error-bounds for finite element method". Numerische Mathematik]*

$$\inf_{u\in U}\sup_{v\in V}\frac{b(u,v)}{\|v\|\|u\|} = \alpha > 0. \tag{18}$$

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36021-3_52

The problem is that during the simulation, we select finite dimensional spaces $U_h \subset U$ and $V_h \subset V$, and we seek $u_h \in U_h, v_h \in V_h$. Then we have:

$$\begin{aligned} \inf_{u\in U}\sup_{v\in V}\frac{b(u,v)}{\|v\|\|u\|} = \alpha > \alpha_h = \inf_{u_h\in U_h}\sup_{v_h\in V_h}\frac{b(u_h,v_h)}{\|v_h\|\|u_h\|}, \\ \frac{M}{\alpha_h} > \frac{M}{\alpha} \geq 1. \end{aligned} \tag{19}$$

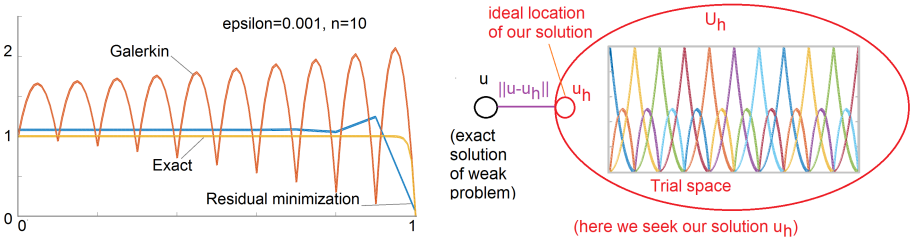

Fig. 1: Comparison of the Galerkin method with trial=test=quadratic B-splines with C^0 separators, and the Petrov-Galerkin method with linear B-splines for trial and quadratic B-splines with C^0 separators for test, and the exact solution.Ideally, the approximation of a solution should be as good as the distance of the space where it lives (trial space) to the exact solution.

The supremum may not be realized in the finite-dimensional subset of the infinite-dimensional test space. This gives the conclusion that we need to test with larger test space V_h, so it realizes the supremum for α, and $\frac{M}{\alpha}$ is closer to 1.

In order to get a better solution, we need to solve in a fixed trial space and test in the larger test space. This approach is used in Petrov-Galerkin $U_h \neq V_h$, but in the case of the "classical" Galerkin method, trial space is equal to test space $U_h = V_h$.

For example, we employ linear B-splines for the trial space $U_h = \{B_{i,1}(x)\}_{1,\ldots,n_x}$, and quadratic B-splines for the test space $V_h = \{B_{i,2}(x)\}_{1,\ldots,N_x}$. We seek the solution in the trial space U_h with 11 basis functions. Our larger test space V_h has 21 basis functions. We need to compute 11 optimal test functions. They are linear combinations of the 21 base functions of V_h.

2.5 Residual minimization - optimal test functions for given ϵ

In the residual minimization method we need to prescribe the norm and scalar product, e.g, $g(u,v) = \int_0^1 (uv + u'v')dx$ or $G_{ij} = g(B_{i,2}, B_{j,2}) = \int_0^1 (B_{i,2}B_{j,2} + \frac{dB_{i,2}}{dx}\frac{dB_{j,2}}{dx})dx$ to minimize the residual of the solution (or the numerical error).

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36021-3_52

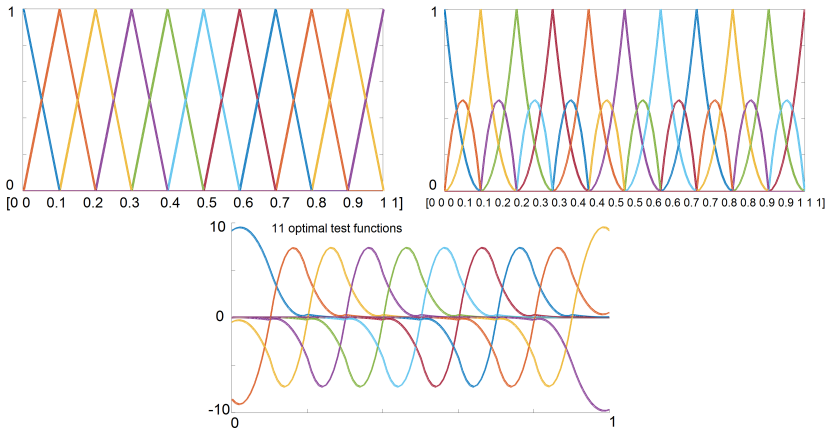

Fig. 2: First plot show 11 linear B-splines, second plot show 21 quadratic B-splines with C^0 separators. In the classical Galerkin method the same basis is used for solution and for testing: $U_h = V_h$. For the Petrov-Galerkin algorithm we carefully choose linearly independent 11 basic functions from the test space of 21 quadratic B-splines.

In the residual minimization problem we minimize the norm under the constrained defined as the solution of our problem

$$\begin{bmatrix} G & B \\ B^T & 0 \end{bmatrix} \begin{bmatrix} r \\ u \end{bmatrix} = \begin{bmatrix} F \\ 0 \end{bmatrix}, \tag{20}$$

where G is the Gram matrix (scalar product matrix) B and F are the problem matrix and right-hand side.

$$G = \begin{bmatrix} g(B_{1,2}, B_{1,2}) & \cdots & g(B_{1,2}, B_{N_x,2}) \\ \vdots & \ddots & \vdots \\ g(B_{N_x,2}, B_{1,2}) & \cdots & g(B_{N_x,2}, B_{N_x,2}) \end{bmatrix}, B = \begin{bmatrix} b(B_{1,1}, B_{1,2}) & \cdots & b(B_{n_x,1}, B_{1,2}) \\ \vdots & \ddots & \vdots \\ b(B_{1,1}, B_{N_x,2}) & \cdots & b(B_{n_x,1}, B_{N_x,2}) \end{bmatrix},$$

$$u = \begin{bmatrix} u_1 \\ \vdots \\ u_{n_x} \end{bmatrix}, \quad r = \begin{bmatrix} r_1 \\ \vdots \\ r_{N_x} \end{bmatrix}, \quad F = \begin{bmatrix} l(B_{1,2}) \\ \vdots \\ l(B_{N_x,2}) \end{bmatrix}.$$

where by $B_{i,2}$ we denote basis functions from the test space, and by $B_{i,1}$ we denote basis functions from the trial space. Our solution is $(u_1, ..., u_{n_x})$. Here $(r_1, ..., r_{N_x})$, represents the residual - the local error map.

The residual minimization method is equivalent to the Petrov-Galerkin formulation with the optimal test functions. The coefficients $\{w_i^k\}_{i=1,...,n_x}$ of $k = 1, ..., n_x$ optimal test functions

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36021-3_52

$\{w^1, \cdots, w^{n_x}\}$ are obtained by solving $Gw = B$.

$$\begin{bmatrix} g(B_{1,2}, B_{1,2}) & \cdots & g(B_{1,2}, B_{N_x,2}) \\ \vdots & \ddots & \vdots \\ g(B_{N_x,2}, B_{1,2}) & \cdots & g(B_{N_x,2}, B_{N_x,2}) \end{bmatrix} \begin{bmatrix} w_1^1 & w_1^2 & \cdots w_1^{n_x} \\ \vdots & \vdots & \vdots \\ w_{N_x}^1 & w_{N_x}^2 & \cdots w_{N_x}^{n_x} \end{bmatrix} = \begin{bmatrix} b(B_{1,2}, B_{1,1}) & \cdots & b(B_{1,2}, B_{n_x,1}) \\ \vdots & \ddots & \vdots \\ b(B_{n_x,2}, B_{1,1}) & \cdots & b(B_{n_x,2}, B_{n_x,1}) \end{bmatrix}$$

This gives system of linear equations with multiple right-hand sides. Solving the above system gives the optimal test functions $V_h^{opt} = span\{w^1, \cdots, w^{n_x}\}$ that form a subspace $V_h^{opt} \subset V_h$. The Petrov-Galerkin formulation with the optimal test functions gives the best possible, up to the trial space used, solution. We get this best possible solution by solving:

$$\begin{bmatrix} b(B_{1,1}, w^1) & \cdots & b(B_{n_x,1}, w^1) \\ \vdots & \ddots & \vdots \\ b(B_{1,1}, w^{n_x}) & \cdots & b(B_{n_x,1}, w^{n_x}) \end{bmatrix} \begin{bmatrix} u_1^{opt} \\ \vdots \\ u_{n_x}^{opt} \end{bmatrix} = \begin{bmatrix} l(w^1) \\ \vdots \\ l(w^{n_x}) \end{bmatrix}, \tag{21}$$

where $(u_1^{opt}, \cdots, u_{n_x}^{opt})$ is the optimal solution in the base of linear B-splines.

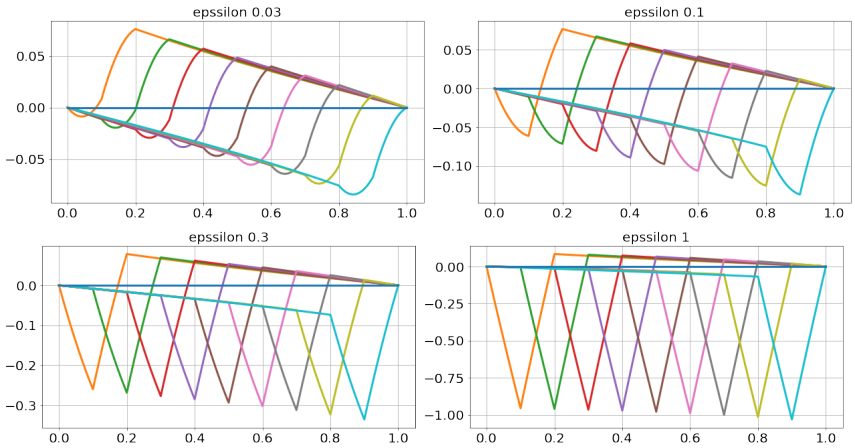

Fig. 3: Test functions for different ϵ calculated using RM method.

3 Numerical results

3.1 Efficient numerical solution using artificial neural net and Petrov-Galerkin method

We recall that our problem is formulated with the Petrov-Galerkin method. We seek the solution u, but it requires the knowledge of the optimal V_h for each $\epsilon \in \mathbb{R}$. In order to automatically obtain the optimal test functions for a given ϵ, we approximate V_h with the neural networks.

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36021-3_52

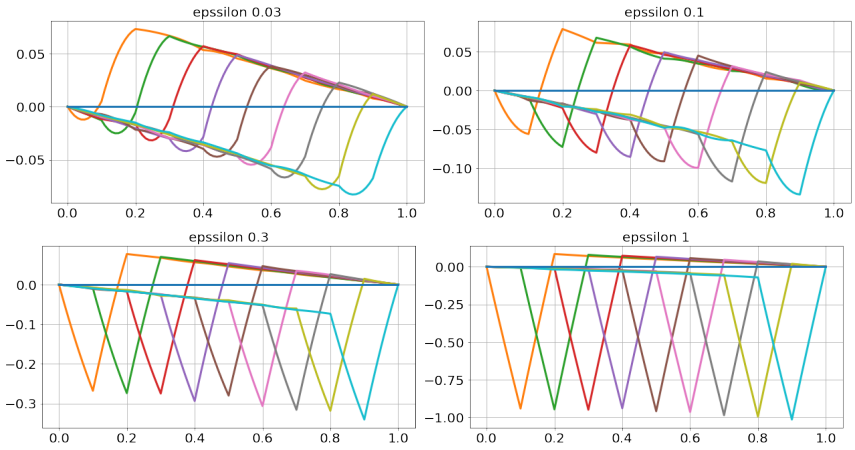

Fig. 4: Test functions for different ϵ calculated using proposed method using simple neural net with 3 hidden layers.

We have to design proper data-set for algorithm. For the given predefined trial $B_{i,1}$ and test $B_{i,2}$ basis functions we need to create set of pairs $(\epsilon, (w_i^k))$ and $(\epsilon, (w^k))$ (mapping from ϵ coefficient into the coefficients w_i^k of the optimal test functions).

- We randomly select $\epsilon \in (0, 1)$ - representative sample should be denser towards 0 - we can use exponential distribution to do that.
- We use residual minimization method to find the coefficients of the optimal test functions w^k

$$\begin{bmatrix} g(B_{1,2}, B_{1,2}) \cdots g(B_{1,2}, B_{N_x,2}) \\ \cdots\cdots\cdots \\ g(B_{N_x,2}, B_{1,2}) \cdots g(B_{N_x,2}, B_{N_x,2}) \end{bmatrix} \begin{bmatrix} w_1^1 \cdots w_1^{n_x} \\ \cdots\cdots \\ w_{n_x}^1 \cdots w_{n_x}^{n_x} \end{bmatrix} = \begin{bmatrix} b(B_{1,1}, B_{1,2}) \cdots b(B_{n_x,1}, B_{1,2}) \\ \cdots\cdots\cdots \\ b(B_{1,1}, B_{N_x,2}) \cdots b(B_{n_x,1}, B_{N_x,2}) \end{bmatrix}$$

In order to enhance the online computation we would use artificial neural network to select the proper optimal test functions. Neural network is a function:

$$\mathsf{NN}(x) = A_n \sigma \left(A_{n-1} \sigma (...\sigma(A_1 x + B_1)... + B_{n-1}) + B_n = y, \right. \tag{22}$$

where A_i are the matrices of coefficients A_i^{mn}, B_i are bias vectors with coordinates B_i^m, and σ is the non-linear activation function.

Our aim in first numerical experiment [16] was to check if for given size of trial and test spaces we can construct a function using artificial neural network that gives single w_i^k coefficient for all $w^k \in V_h^{opt}$.

$$\forall \epsilon \in \mathbb{R}_+ \quad \mathsf{NN}_{i,k}(\epsilon) \longrightarrow \omega_i^k \approx w_i^k, \quad w^k \approx [\mathsf{NN}_{1k}(\epsilon), \cdots, \mathsf{NN}_{n_x k}(\epsilon)] \tag{23}$$

It would be impossible to test and train NN for every ϵ so we have to choose a range of ϵ. We make a mapping that takes logarithm of ϵ onto some predefined

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36021-3_52

closed interval $[-1, 1]$. Such scaling makes the input of NN more sensitise to smaller values of epsilon.

$$scaling(\epsilon) : \forall \epsilon \in E \subset \mathbb{R}_+ \quad \log_{10} \epsilon \longrightarrow [-1, 1] \tag{24}$$

Let $\epsilon_{min} = \inf E$ and $\epsilon_{max} = \sup E$ then:

$$\text{for } \epsilon < \epsilon_{min},\ scaling(\epsilon) > 1 \text{ and for } \epsilon > \epsilon_{max},\ scaling(\epsilon) < -1. \tag{25}$$

Conclusion from 25 is that we can choose scalled epsilons outside the training set. If our set E have representative values then it would approximate well all the epsilons in $\mathbb{R}_+$. Well designed set E implies proper working outside the training set. Small values of ϵ gives more numerically unstable solutions then bigger values of ϵ. First, we generate the optimal test functions for each ϵ, and we check if NN can approximate well the coefficients of test function. We experiment with the optimal number of layer and neurons. We have found that the optimal approximation is obtained when we construct one neural network for one optimal test function.

$$\forall \epsilon \in E \quad \mathsf{NN}_k(scaling(\epsilon)) \longrightarrow \upsilon^k = [\omega_1^k, \cdots, \omega_{n_x}^k], \quad \upsilon^k \approx w^k \in V_h^{opt} \tag{26}$$

Our equation for computing the optimal solution with the optimal test functions as provided by the artificial neural networks takes the following form:

$$\begin{bmatrix} b(B_{1,1}, \mathsf{NN}_1(\epsilon)) & \cdots & b(B_{n_x,1}, \mathsf{NN}_1(\epsilon)) \\ \vdots & \ddots & \vdots \\ b(B_{1,1}, \mathsf{NN}_{n_x}(\epsilon)) & \cdots & b(B_{n_x,1}, \mathsf{NN}_{n_x}(\epsilon)) \end{bmatrix} \boldsymbol{u} = \begin{bmatrix} l(\mathsf{NN}_1(\epsilon)) \\ \vdots \\ l(\mathsf{NN}_{n_x}(\epsilon)) \end{bmatrix}. \tag{27}$$

To keep computations using NN as simple as possible, we propose 3 hidden layer neural network. At input and output, we use the linear activation function and ReLU ($\sigma(x) = \max\{0, x\}$) in hidden layers. We use the Adam optimizer with default settings, and we use the loss function defined as the mean square error. The mean absolute percentage error was also needed in this case as it monitors overall sensitivity for all values. We use Python language with the TensorFlow package compiled to use GPU support. The training procedure takes only 4 seconds to perform 1000 epochs on Nvidia GTX 1650Ti. In the [16], we focused on justification and investigated the aim of using the neural network in coefficient approximation. One or two layers could approximate one single coefficient but performed badly when it came to the whole test function. Too many layers also affect the quality of the coefficients, as neural nets tend to seek patterns. The 4 layer neural network works well in the case of matrix approximation, where there is plenty of repetition of similar values. In the [19], we discussed such cases and the procedure to find the optimal neural net setup.

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36021-3_52

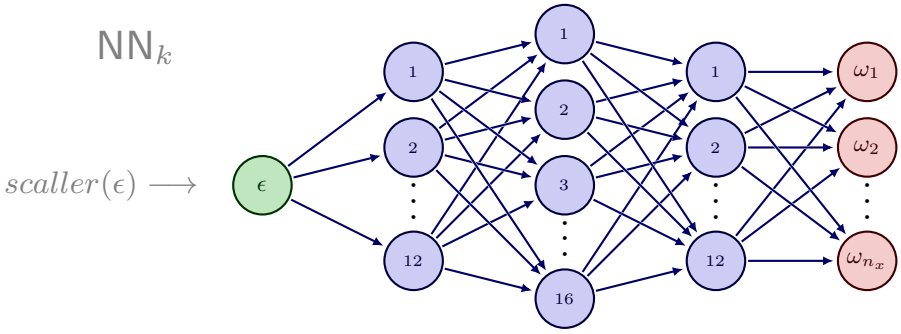

Fig. 5: NN used in computations have 3 hidden dense layers with 12,16 and 12 neurons respectively.

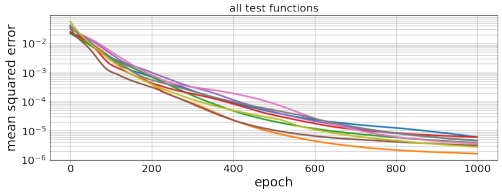

Fig. 6: Learning procedure for all 9 of 11 optimal test functions (for $n = 11$). The first and last test functions are skipped due to the 0 boundary condition.

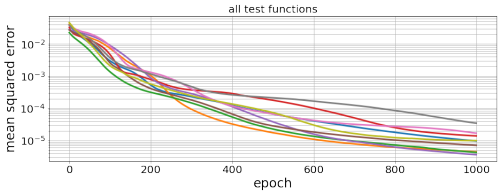

Fig. 7: Interpolation of optimal test functions for a set of epsilons inside the training range that were not used during training. The test set was designed in a way that between every two neighboring epsilons from the train set, one was selected for the test set.

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36021-3_52

The algorithm for Petrov-Galerkin method enhanced with neural networks providing the optimal test functions:

1. Predefine trail U_h and test V_h spaces for solution that satisfies $\dim U_h < \dim V_h$.
2. Construct set E from representative sample of diffusion parameter ϵ
   - for example selection can be done by using exponential distribution from interval $(0, 1)$.
3. Using residual minimization find V_h^{opt}
   - for every $\epsilon \in E$ we calculate optimal test functions by solving $Gw = B$
4. Fit $scaling(\cdot)$ for elements from set E.
5. For every $\epsilon \in E$ construct the data-set consisting set of pairs $(scaller(\epsilon), w^k)$, $w^k \in V_h^{opt}$
6. fit NN_i for every $\epsilon \in E$ with data set $(scaling(\epsilon), w^i)_{i=1,\dots,n_x}$ to generate coefficients approximating space V_h^{opt}.
7. Simulate equation:
   (a) for a given new $\hat{\epsilon}$ generate approximate w_{opt} by using $\upsilon_k = \mathsf{NN}_k(scaller(\hat{\epsilon}))$, $k = 1, \dots, n_x$
   (b) calculate $u \approx u^{opt}$ by solving the equation:

$$\begin{bmatrix} b(B_{1,1}, \upsilon^1) & \cdots & b(B_{n_x,1}, \upsilon^1) \\ \vdots & \ddots & \vdots \\ b(B_{1,1}, \upsilon^{n_x}) & \cdots & b(B_{n_x,1}, \upsilon^{n_x}) \end{bmatrix} \boldsymbol{u} = \begin{bmatrix} l(\upsilon^1) \\ \vdots \\ l(\upsilon^{n_x}) \end{bmatrix}. \tag{28}$$

MAPE	Number of epochs				
Test function	100	300	1000	3000	10000
2	104.732	26.100	6.229	2.371	0.374
3	110.895	23.452	0.826	0.988	0.224
4	102.360	23.616	4.977	1.811	2.848
5	130.999	15.936	7.905	5.779	0.546
6	11971.054	149.536	14.874	6.198	3.352
7	90.811	37.762	6.205	1.578	2.661
8	98.659	14.200	11.952	1.851	0.034
9	104.788	18.215	10.215	0.561	0.002
10	114.502	25.796	1.870	0.806	0.282

Table 1: The table shows relation between number of epochs and the mean squared percentage error (MAPE) for $n = 11$.

We can check what is the minimum number of samples (epsilons) to train the neural network to solve the equation with satisfactory low error.

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36021-3_52

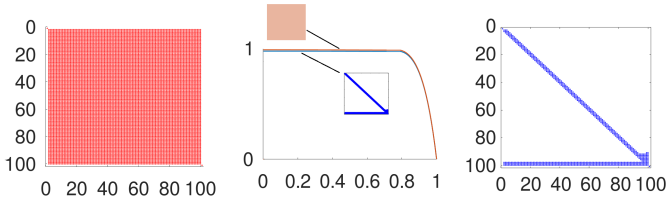

Fig. 8: Trial space linear B-splines, test space quadratic B-splines with C^0 separators, 100 elements, $\epsilon = 0.1$. (Left panel:) Dense matrix from optimal test functions. (Right panel:) Sparse matrix with low rank terms removed. (Middle panel:) Solutions from dense matrix and modified sparse matrix.

3.2 Dealing with global optimal test functions.

The optimal test functions are global, and thus they cannot be used directly for efficient computations, since the Petrov-Galerkin system build with the optimal test functions is global (see left panel in Figure 8).

Given a trial function $u_h \in U_h$, the corresponding optimal test function v realizes the supremum defining V'-norm of Bu_h, so it satisfies

$$v = argmax_{w \in V} \frac{b(u_h, w)}{\|w\|_V}. \tag{29}$$

Since $b(u_h, v)$ is defined by an integral over the domain Ω of some expression involving products of u_h, v and their gradients, behavior of v outside the support K of u_h is irrelevant to the value of $b(u_h, v)$, but does influence $\|v\|_V$. Let $g = v|_{\partial K}$ be the trace of v on ∂K and consider a function $w \in H^1(\Omega \setminus K)$ such that $w|_{\partial\Omega} = 0$, $w|_{\partial K} = g$. Such w can be extended to $\tilde{w} \in H^1_0(\Omega) = V$ by

$$\tilde{w}(x) = \begin{cases} w(x), & x \notin K \\ v(x), & x \in K \end{cases}. \tag{30}$$

Then $\tilde{w}|_K = v|_K$, and so $b(u_h, \tilde{w}) = b(u_h, v)$. By the maximization property of v, $\|\tilde{w}\|_V \geq \|v\|_V$. As $\tilde{w}$ and v are equal on K, it follows that

$$\int_{\Omega \setminus K} \|\nabla v\|^2 \, dx \leq \int_{\Omega \setminus K} \|\nabla w\|^2 \, dx. \tag{31}$$

and so v restricted to $\Omega \setminus K$ has the minimal L^2-norm of the gradient among all the functions satisfying boundary conditions $\cdot|_{\partial\Omega} = 0$ and $\cdot|_{\partial K} = g$. By Dirichlet's principle, v is the solution of the Laplace equation $\Delta u = 0$ on $\Omega \setminus K$ with these boundary conditions. For example, in 1D case where Ω and K are intervals, solution of such boundary value problem is a linear function, hence optimal test functions outside the support of the corresponding trial function decrease linearly until reaching 0 at the boundary.

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36021-3_52

Let us illustrate it on the advection-dominated diffusion equations with 100 elements in one dimension. We show on Figure 9 two exemplary basis functions v_{20} and v_{100} and how they change with ϵ.

The basis of all optimal test functions for 100 elements with trial space of linear B-splines and test space of quadratic B-splines with C^0 separators are presented on left panel in Figure 10. We can replace this basis by a linear combination, where all except the last two optimal test functions have local supports (they are equal to a very small number on the other parts of the domain, and this number results from the differences of slopes of the original functions and it decreases to zero when we increase the number of elements). Using these basis functions for Petrov-Galerkin formulation results in a sparse matrix that can be factorized in a linear cost (see right panel in Figure 8). Most of the off-diagonal terms are very small, and their contribution is low rank and can be neglected (neglecting them does not alter the solution, see the middle panel in Figure 8). This localization of the optimal test functions can be generalized to two and three dimensions.

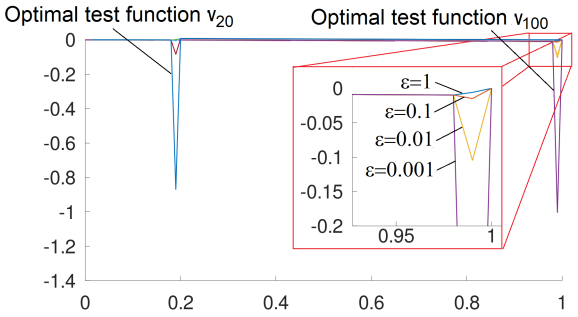

Fig. 9: Trial space with linear B-splines, 100 elements, test space with quadratic B-splines with C^0 separators, 100 elements. Two selected optimal test functions v_{20} and v_{100} for different ϵ.

4 Conclusions

We showed that the neural network could learn coefficients of the optimal test functions for different parameters of the PDE. They allow for the automatic stabilization of advection-dominated diffusion problems. Moreover, the optimal test functions can be approximated with the function having local support, thus making the matrix of the coefficients of the global test functions sparse. We have verified our methodology using a one-dimensional advection-dominated diffusion problem. Future work will involve the generalization of the method into higher dimensions, as well as replacing the solver with the hierarchical matrices [7, 18]. Some preliminary results on the hierarchical matrices solver for two-dimensional problems are discussed in [19].

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36021-3_52

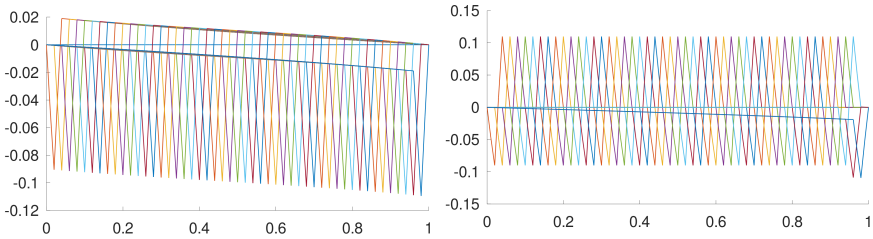

Fig. 10: Trial space linear B-splines, test space quadratic B-splines with C^0 separators, 100 elements. (Left panel:) All optimal test functions. (Right panel:) Optimal test functions replaced by a linear combination (each optimal test function subtracted from the previous one).

5 Acknowledgement

The Authors are thankful for support from the funds assigned to AGH University of Science and Technology by the Polish Ministry of Science and Higher Education. Research project partly supported by program "Excellence initiative – research university" for the AGH University of Science and Technology.
The publication has been supported by a grant from the Faculty of Management and Social Communication under the Strategic Programme Excellence Initiative at Jagiellonian University.

References

1. Y. Bazilevs, V.M. Calo, J.A. Cottrell, T.J.R. Hughes, A. Reali, and G. Scovazzi. Variational multiscale residual-based turbulence modeling for large eddy simulation of incompressible flows. *Computer Methods in Applied Mechanics and Engineering*, 197(1):173–201, 2007.
2. J. Berg and K. Nystrom. Data-driven discovery of PDEs in complex datasets. *Journal of Computational Physics*, 384:239–252, 2019.
3. I. Brevis, I. Muga, and K. van der Zee. Data-driven finite elements methods: Machine learning acceleration of goal-oriented computations. *Arxiv*, arXiv:2003.04485:1–24, 2020.
4. J Austin Cottrell, TJR Hughes, and Yuri Bazilevs. *Isogeometric analysis: toward integration of CAD and FEA*. John Wiley & Sons, 2009.
5. L. Demkowicz. 2D *hp*-adaptive finite element package (2Dhp90) Version 2.0. *Ticam Report*, 2:06, 2002.
6. L. Demkowicz, W. Rachowicz, and Ph. Devloo. A fully automatic *hp*-adaptivity. *Journal of Scientific Computing*, 17(1-4):117–142, 2002.
7. Wolfgang Hackbusch, Lars Grasedyck, and Steffen Börm. An introduction to hierarchical matrices. *Mathematica Bohemica, v.127, 229-241 (2002)*, 127, 01 2002.
8. C.F. Higham. Deep learning: An introduction for applied mathematicians. *SIAM Review*, 61(4):860–891, 2019.
9. T.J.R. Hughes, J.A. Cottrell, and Y. Bazilevs. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Computer methods in applied mechanics and engineering*, 194(39):4135–4195, 2005.

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36021-3_52

10. T.J.R. Hughes, L. Franca, and M. Mallet. A new finite element formulation for computational fluid dynamics: Vi. convergence analysis of the generalized supg formulation for linear time-dependent multidimensional advective-diffusive systems. *Computer Methods in Applied Mechanics and Engineering*, 63(1):97–112, 1987.
11. M. Łoś, Q. Deng, I. Muga, M. Paszyński, and V. Calo. Isogeometric residual minimization method (iGRM) with direction splitting preconditoner for stationary advection-diffusion problems. *arXiv:1906.06727, Computer Methods in Applied Mechanics and Engineering (in press.)*, 2020.
12. M. Łoś, J. Munoz-Matute, I. Muga, and M. Paszyński. Isogeometric residual minimization method (iGRM) with direction splitting for non-stationary advection–diffusion problems. *Computers & Mathematics with Applications*, 79(2):213–229, 2019.
13. M. Łoś, J. Munoz-Matute, I. Muga, and M. Paszyński. Isogeometric residual minimization for time-dependent stokes and navier-stokes problems. *arXiv:2001.00178, Computers & Mathematics with Applications (in press.)*, 2020.
14. M. Łoś, S. Rojas, M. Paszyński, I. Muga, and V. Calo. Discontinuous galerkin based isogeometric residual minimization for the stokes problem (DGIRM). *invited to the special issue of Journal of Computational Science on 20th anniversary of ICCS conference*, 2020.
15. J. A. A. Opschoor, P. C. Petersen, and C. Schwab. Deep ReLU networks and high-order finite element methods. *Analysis and Applications*, 18(5):715–770, 2020.
16. M. Paszyński and T. Służalec. Petrov-galerkin formulation equivallent to the residual minimization method for finding an optimal test functions. *ECCOMAS Congress 2022*, 5–9 June 2022, Oslo, Norway.
17. D. Pietro and A. Ern. *Mathematical Aspects of Discontinuous Galerkin Methods*. Springer, 2011.
18. Phillip Schmitz and Lexing Ying. A fast nested dissection solver for cartesian 3D elliptic problems using hierarchical matrices. *Journal of Computational Physics*, 258:227–245, 02 2014.
19. T. Służalec, M. Dobija, A. Paszyńska, I. Muga, and M. Paszyński. Automatic stabilization of finite-element simulations using neural networks and hierarchical matrices. *arxiv.org/abs/2212.12695*, 2022.
20. G. Tsihrintzis, D. N. Sotiropoulos, and L. C. Jain. *Machine learning paradigms: Advances in data analytics*. Springer, 2019.
21. Jinchao Xu. Finite neuron method and convergence analysis. *Communications in Computational Physics*, 28:1707–1745, 2020.

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36021-3_52