Augmenting Petrov-Galerkin method with optimal test functions by DNN learning the inverse of the Gram matrix

Tomasz Służalec[0000−0001−6217−4274]

AGH University of Krakow, Poland
sluzalec@agh.edu.pl

Abstract. The Petrov-Galerkin (PG) method is a robust alternative to the Galerkin method for finite element simulations of challenging partial differential equations (PDEs). The solution of the Galerkin method is obtained from the linear system $\mathbf{B}x = F$ resulting from discretization of trial and test spaces. Although the Galerkin method enforces the equality of trial and test spaces ($U_h = V_h$) and relies on the inf-sup stability condition to ensure solution accuracy, it often fails for difficult problems where the discrete inf-sup constant α_h significantly deviates from the abstract inf-sup constant α. This discrepancy leads to numerical instability and incorrect solutions. The PG method addresses this by allowing distinct trial and test spaces ($U_h \neq V_h$), enabling the selection of test functions that improve the discrete inf-sup constant α_h. Of particular interest is the Petrov-Galerkin method with optimal test functions (PGO), where the test functions are computed to maximize α_h, ensuring stable solutions even for ill-conditioned problems. The PGO method modifies the discrete test space to approximate the abstract stability properties as closely as possible. The computation of optimal test functions involves solving $\mathbf{GW} = \mathbf{B}$, where $\mathbf{B}$ is the Galerkin matrix, and $\mathbf{G}$ is the Gram matrix of the test space's inner product. Solving $\mathbf{B}^T\mathbf{W}x = \mathbf{W}^T F$ then yields a stable solution. However, the added computational cost of inverting $\mathbf{G}^{-1}$ poses a significant overhead. In this work, we propose a novel approach that leverages deep neural networks (DNNs) to approximate the inverse of the Gram matrix for a class of advection-diffusion problems with variable diffusion coefficients. By training the DNN to predict $\mathbf{G}^{-1}$, we eliminate the computational overhead of matrix inversion, enabling efficient and stable solutions of PDEs. Our results demonstrate the effectiveness of the DNN-enhanced PGO method in maintaining stability and accuracy, even for difficult computational problems where the standard Galerkin method fails. This approach represents a significant advancement in the practical applicability of the Petrov-Galerkin framework for solving complex PDEs.

Keywords: Advection-diffusion problem, Galerkin method, Petrov-Galerkin method with optimal test functions, Deep Neural Networks, Gram matrix

1 Introduction

Petrov-Galerkin (PG) method [1,4,13,11] is an alternative to the Galerkin method for performing finite element simulations of difficult Partial Differential Equations (PDE). In the Galerkin method, we consider the weak formulation of the PDE, reading: Find $u \in U$ such that $b(u,v) = l(v)$ for all $v \in V$. For the problems that are bounded $b(u,v) \leq \alpha \|u\|_U \|v\|_V$ and inf-sup stable $inf_{u\in U} sup_{v\in V} \frac{b(u,v)}{\|v\|\|u\|} = \alpha > 0$ there exists a unique "stable" solution. On the discrete level, e.g. using the B-spline basis functions discretization from isogeometric analysis (IGA) [7,3], in the Galerkin method, the trial and the test spaces are considered to be equal $U_h = V_h$, and they generate the system of linear equations $\mathbf{B}x = F$. The inf-sup condition on the discrete level, where we have replaced the infinite dimensional trial and test spaces U, V by their discrete counterparts U_h, V_h may not be fulfilled, namely $inf_{u\in U} sup_{v\in V} \frac{b(u,v)}{\|v\|\|u\|} = \alpha > \alpha_h = inf_{u_h\in U_h} sup_{v_h\in V_h} \frac{b(u_h,v_h)}{\|v_h\|\|u_h\|}$. Let μ be the continuity constant $b(u,v) \leq \mu \|u\|_U \|v\|_V$. In this case $\frac{\mu}{\alpha_h} > \frac{\mu}{\alpha} \geq 1$ and we can get incorrect "unstable" solutions. However, in the Petrov-Galerkin method, the trial and test spaces can be different, and the discrete inf-sup constant can be "better". Of our particular interest is the Petrov-Galerkin method with the optimal test functions (PGO). In the PGO method, we compute the optimal test functions that allow us to obtain correct solutions even for difficult computational problems. We modify the discrete test space, so we get the discrete inf-sup constant α_h as close to the abstract inf-sup constant α as possible, using these spaces. The matrix of coefficients of the optimal test functions $\mathbf{W}$ can be computed by solving $\mathbf{GW} = \mathbf{B}$, or $\mathbf{W} = \mathbf{G}^{-1}\mathbf{B}$, where $\mathbf{B}$ is the discrete Galerkin problem matrix, and $\mathbf{G}$ is the Gram matrix of the inner product for which we can prove that our weak formulation is inf-sup stable. Having the matrix of the coefficients of the optimal test functions, we solve $\mathbf{B}^T\mathbf{W}x = \mathbf{W}^T F$ which provides a "stable" solution. In other words, the PGO method allows one to obtain a correct solution of difficult "unstable" problems, for which the Galerkin method results in incorrect "unstable" solutions. The overhead of using the Petrov-Galerkin method is the cost of inverting the Gram matrix $\mathbf{G}$. In this paper, for a class of advection-diffusion problems with variable diffusion coefficients, we train the Deep Neural Network (DNN) the inverse of the Gram matrix. Using this DNN, we can obtain stable solution of our PDE without the additional overhead, just by asking the DNN for the inverse of the Gram matrix.

In our previous work [15,16] we have trained the neural network the matrix of the coefficients of the optimal test functions $\mathbf{W}$. However, having the inverse of the Gram matrix is more useful, since it can be applied to directly compute the coefficients of the optimal test functions for finite element method computations, as well as it can be applied to obtain the robust loss function in the Robust Variational Physics Informed Neural Networks (RVPINN) [14]. DNNs have been used to augment the mesh-based simulations, supporting automatic selection of mesh refinements [12,17] in the finite element method setup [5], or training the Variational PINN solution using test functions span over the computational mesh [8,14].

The structure of the paper is the following. In Section 2 we introduce a family of exemplary advection-diffusion problems, including the B-spline based discretization. Section 3 describes the design and training of the neural network. Section 4 presents exemplary numerical results. The paper is concluded in Section 5.

2 Problem formulation

We focus on the advection-diffusion problem: Given $\Omega = (0,1)^2 \subset \mathbb{R}^2$ we solve

$$\beta \cdot \nabla u(x_1, x_2) - \nabla \cdot (\epsilon\,(x_2)\,\nabla u(x_1, x_2)) = f(x_1, x_2) \tag{1}$$

with the $sin(\Pi x_2)$ Dirichlet b.c. on the left boundary, and the arbitrary right-hand side functions $f(x_1, x_2)$. Here $\beta = (\beta_x, \beta_y) = (1, 0)$ is the assumed advection vector, and ϵ has a non-constant distribution. Namely, We assume the following five layers in the computational domain, where we vary the diffusion coefficient arbitrarily from $(1, 10)$. The exemplary setup of the computational domain is illustrated in Figure 1.

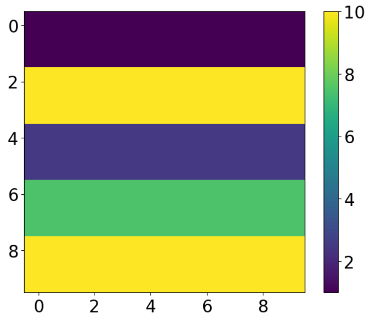

Fig. 1: The sample body in consideration.

The weak formulation of the problem: Find $u_h \in U_h$ such that $b(u_h, v_h) = l(v_h)$, $\forall v_h \in V_h$ reads

$$\begin{aligned} b(u,v) = \beta_x \left(\frac{\partial u}{\partial x}, v\right)_\Omega + \beta_y \left(\frac{\partial u}{\partial y}, v\right)_\Omega + \left(\epsilon \frac{\partial u}{\partial x}, \frac{\partial v}{\partial x}\right)_\Omega + \left(\epsilon \frac{\partial u}{\partial y}, \frac{\partial v}{\partial y}\right)_\Omega \\ - \left(\epsilon \frac{\partial u}{\partial x} n_x, v\right)_\Gamma - \left(\epsilon \frac{\partial u}{\partial y} n_y, v\right)_\Gamma \end{aligned} \tag{2}$$

where $n = (n_x, n_y)$ is the versor normal to $\Gamma = \partial\Omega$, and

$$l(v) = (f, v)\,. \tag{3}$$

We discretize with B-spline basis functions. For example, let us consider the discretization with B-spline basis functions with knot vectors and knot points.

```
knot_x = knot_y = [0 0 0 1 2 3 4 5 6 7 8 8 8];
points_x = points_y = [0 0.125 0.25 0.375 0.5 0.625 0.75 0.875 1];
```

The basis functions along x axis are obtained by introducing knot points $\xi_i = points\_x[knot_x[i]+1]$ into the recursive formula (8),

$$
\begin{aligned}
&B_{i,0}(\xi) = 1 \text{ if } \xi_i \le \xi \le \xi_{i+1}, \text{ otherwise } 0,\\
&B_{i,p}(\xi) = \frac{\xi - \xi_i}{\xi_{i+p} - \xi_i} B_{i,p-1}(\xi) + \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_i} B_{i+1,p-1}(\xi),
\end{aligned}
\tag{4}
$$

for the order p defined as the number of repetitions of the first $knot\_x[1]$ minus one, assuming that the subsequent knots inserted into the denominator must be different, and if they are not different, then the given term is changed to zero. Similarly, the basis functions along y axis are obtained by introducing knot points $\xi_i = points\_y[knot_y[i]+1]$ into the recursive formula (5). The two-dimensional basis functions are obtained by the tensor products of the one-dimensional basis.

$$
\{B^{xy}_{ij,p}(x,y) = B^x_{i,p}(x) B^y_{j,p}(y)\}_{i=1,\ldots,N_x; j=1,\ldots,N_y} \tag{5}
$$

In the Galerkin method these basis functions define the trial and test spaces. The problem matrices are defined as follows

$$
\begin{aligned}
\mathbf{B}_{n,m} = \left(\frac{\partial B^x_{i,p}}{\partial x} B^y_{j,p}, B^{xy}_{kl,p} \right)_\Omega + \epsilon \left(\frac{\partial B^x_{i,p}}{\partial x} B^y_{j,p}, \frac{\partial B^x_{k,p}}{\partial x} B^y_{l,p} \right)_\Omega \\
+ \epsilon \left(B^x_{i,p} \frac{\partial B^y_{j,p}}{\partial y}, B^x_{k,p} \frac{\partial B^y_{l,p}}{\partial y} \right)_\Omega ,
\end{aligned}
\tag{6}
$$

where $n = i + (j-1)N_y$, $m = k + (l-1)N_y$, and the right-hand side vector is

$$
f_m = -\left((f(x,y), B^{xy}_{kl,p} \right)_\Omega . \tag{7}
$$

To deliver the Petrov-Galerkin formulation with the optimal test functions, we need to compute the matrix of coefficients of the optimal test functions $\mathbf{W}$ by solving the system of equations $\mathbf{GW} = \mathbf{B}$ (see [6,18,2]). Here $\mathbf{G}$ is the Gram matrix of selected inner product, namely

$$
\begin{aligned}
\mathbf{G}_{n,m} = \left(B^{xy}_{ij,p}, B^{xy}_{kl,p} \right)_\Omega + \left(\epsilon \frac{\partial B^x_{i,p}}{\partial x} B^y_{j,p}, \frac{\partial B^x_{k,p}}{\partial x} B^y_{l,p} \right)_\Omega \\
+ \left(\epsilon B^x_{i,p} \frac{\partial B^y_{j,p}}{\partial y}, B^x_{k,p} \frac{\partial B^y_{l,p}}{\partial y} \right)_\Omega ,
\end{aligned}
\tag{8}
$$

For efficient solution of the system $\mathbf{GW} = \mathbf{B}$ we need to compute the inverse of $\mathbf{G}^{-1}$. Having the matrix of the coefficients of the optimal test functions $\mathbf{W} = \mathbf{G}^{-1}\mathbf{B}$, we can solve the stabilized problem $\mathbf{B}^T \mathbf{W} x = \mathbf{W}^T F$.

3 Application of Deep Neural Networks

We apply Deep Neural Networks as the expert providing the inverse of the Gram matrix for the given configuration of the diffusion parameters. The exemplary structure of the Gram matrix **G** is presented in Figure 2. The Gram matrix is sparse. The selected structure of the neural network is presented in Figure 3. Although the choice of an optimal DNN architecture is typically experimental at the beginning, the selection process resembles approaches used in genetic algorithms. We systematically explore various architectures, evaluating them through validation metrics after training for a limited number of epochs. The architectures tested differ in depth, width, and activation function types. Subsequently, the most promising architectures undergo fine-tuning and further validated. For this problem we applied different combinations of the feed forward neural networks, with different numbers of neurons $4, 6, 8, 12, 15, 32, 64$, different loss functions, and activation functions. The most promising was the neural network with 4 hidden layers with 32, 12 , 64, and 32 neurons receptively. The input layer consists of the five ϵ_i, $i = 1, 2, 3, 4, 5$ values describing the diffusion coefficients on the layers of the domain, from the top layer to the bottom layer. The last output layer has the dimension of 100×100 and give the whole inverse matrix $\mathbf{G}^{-1}$ assuming the quadratic B-splines span over eight intervals for the discretization. We employ the ReLU activation functions in hidden layers and we use mean squared error as the loss function. The dataset consists of 3125 $(= 5^5)$

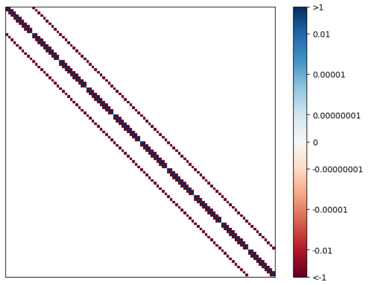

Fig. 2: The sample of calculated **G** matrix (size 100×100), for the computational mesh with 10×10 quadratic B-spline basis functions in each direction.

diffusion records on the input (they are all the combinations of $1, 2.5, 5, 7.5$ and 10 diffusion coefficients spanning over the five uniform layers) and the computed inverse matrices $\mathbf{G}^{-1}$, obtained by generating the Gram matrix with given diffusion coefficients and inverting the matrix. The input for the neural network is (continuous) not categorical; thus, we can calculate the value for any epsilon from the interval $[1, 10]$. The proposed architecture was trained and validated for the same dataset. The explanation for this approach is that we want the neural network to be the approximation of the data, so we want it to be overfitted to

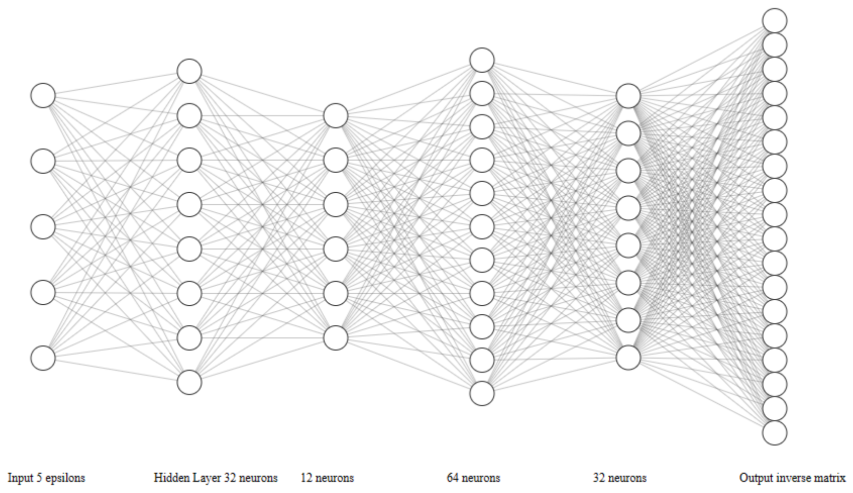

Fig. 3: The neural network architecture.

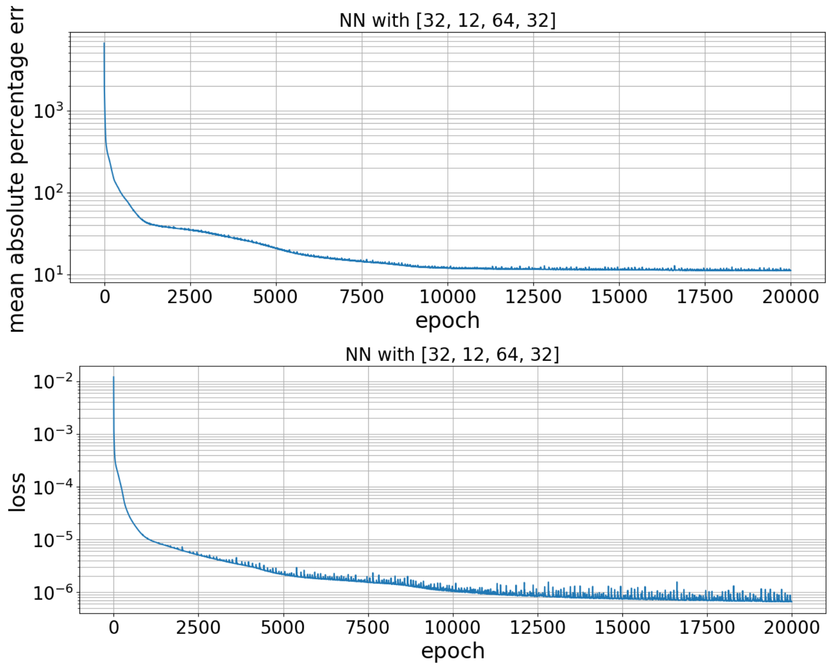

Fig. 4: Learning procedure for the feed forward neural network with 32, 12 ,64, 32 neurons. The learning process consists of 20000 epochs with mean squared error as the loss function.

perform the task well. In contrast to the classification or segmentation problems performed by the deep learning approach, our approach did not require the need to perform some generalization task that is needed to classify the new data. Here in our problem, we have to consider that we have mid steps in output between every known target output - continuous elements of the input have direct and smooth impact for the continuity/nonlinearity change of the target matrix. We consider the neural network expert to be as the black box function that can generate the inverse matrix. The neural network training is presented in Figure 4. We performed 20,000 iterations using the ADAM algorithm [9]. On the first panel in Figure 4 we present the relative error $\sum_{i,j} \frac{|\mathbf{G}^{-1}_{i,j} - \mathcal{G}^{-1}_{i,j}|}{|\mathcal{G}^{-1}_{i,j}|}$ where $\mathcal{G}^{-1}$ is the Gram matrix computed exactly, and the $\mathbf{G}^{-1}$ is the Gram matrix estimated by the neural network. On the second panel we present the MSE loss function defined as $\sum_{i,j} |\mathbf{G}^{-1}_{i,j} - \mathcal{G}^{-1}_{i,j}|$.

The primary advantage of employing DNNs to approximate the inverse Gram matrix arises during the online phase (runtime) of solving PDEs. While the offline phase (training the DNN) involves additional computational overhead, it is performed once and remains valid across multiple simulations. During the runtime phase, the trained DNN rapidly approximates the inverse Gram matrix, significantly reducing the computation time compared to the direct inversion method. Specifically, direct inversion of the Gram matrix has a computational complexity of approximately $O(n^3)$ for matrices of size n, whereas a forward pass through the trained neural network is significantly cheaper (typically $O(n^2)$ or better depending on architecture and parallelization), thus dramatically improving computational efficiency.

4 Numerical results

In this section, we verify the trained neural network by considering five different configurations of the diffusion coefficients in layers of the computational domain. They are presented on the first panel in Figure 5 (five uniform layers $\{1, 10, 2.5, 7.5, 10\}$), on the first panel in Figure 7 (two layers $\{10, 1\}$, span over 2/5 and 3/5 of the domain), on the first panel in Figure 9 (three layers $\{10, 2.5, 1\}$, span of 1/5, 1/5, and 3/5 of the domain), on the first panel in Figure 11 (four layers $\{1, 2.5, 7, 1\}$ span over 1/5, 1/5, 1/5 and 2/5 of the domain), on the first panel in Figure 13 (three layers $\{1, 2.5, 1\}$, span over 3/5, 1/5 and 1/5 of the domain), and on the first panel in Figure 15 (one uniform layer with $\epsilon = 1$). In the right panels of Figures 5, 7, 9, 11, and 13 there are the inverse Gram matrices $\mathbf{G}^{-1}$ as obtained from the neural network. In general, we can see the symmetric layered structure of the inverse of the Gram matrix. For each considered case, we generate the Gram matrix $\mathbf{G}$ exactly, and we compute $\mathbf{G}^{-1}\mathbf{G}$ that is supposed to be equal to the identity matrix $\mathbf{I}$ if the neural networks provided correct result. On the first panel of Figures 6, 8, 10, 12, 14, and 16, we present the inverse of the Gram matrix corresponding to the configuration from Figures 5, 7, 9, 11, 13, and 15, respectively. On the second panel in Figures 6, 8, 10, 12, 14, and 16 we present the result of the multiplication $\mathbf{G}^{-1}\mathbf{G} \approx \mathbf{I}$. On the third panel of the Figures, we present the zoom towards the inverse matrix using the logarithmic scale (cropped to interval $[0, 0.1]$) that is focused on showing the tiny differences in the small values).

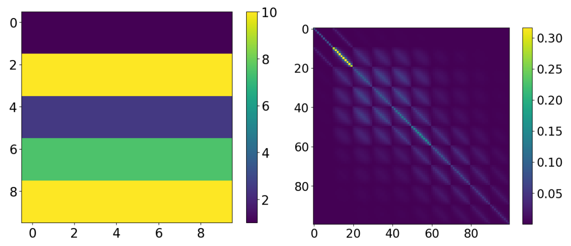

Fig. 5: The computational domain with diffusion coefficients in five uniform layers $\{1, 10, 2.5, 7.5, 10\}$ (left panel) and the calculated inverse of the Gram matrix $\mathbf{G}^{-1}$ (right panel).

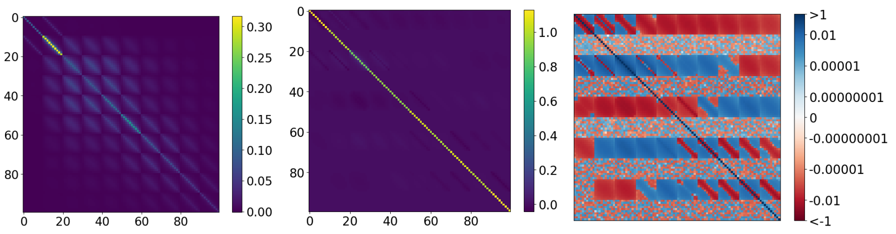

Fig. 6: The predicted inverse matrix (first panel), the result of $\mathbf{G}^{-1}\mathbf{G} \approx \mathbf{I}$ (second panel), and the logarithmic zoom towards the approximation of the inverse matrix (third panel). The computations for the computational domain with diffusion coefficients in five uniform layers $\{1, 10, 2.5, 7.5, 10\}$.

As we see in the numerical results, the predicted inverse of the Gram matrix $\mathbf{G}^{-1}$ is well recreated. The only difference is if we look closely at the values in the computation of the eye of the identity matrix. The other values that are supposed to be equal to 0 are very well approximated.

Our trained DNN has shown good generalization capabilities across different diffusion coefficients and moderately varying material settings within the considered class of problems and range of values for material setting. However, extensive variations in problem parameters (such as drastic changes in material properties, geometry, or boundary conditions) require retraining or even a different architecture.

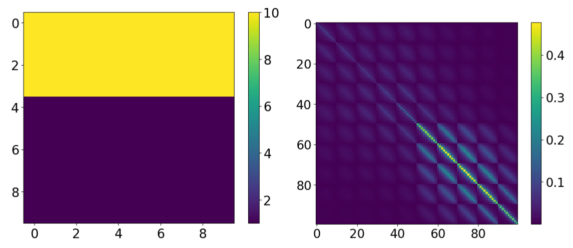

Fig. 7: The computational domain with diffusion coefficients in two layers $\{10, 1\}$, span over $2/5$ and $3/5$ of the domain (left panel) and the calculated inverse of the Gram matrix $\mathbf{G}^{-1}$ (right panel).

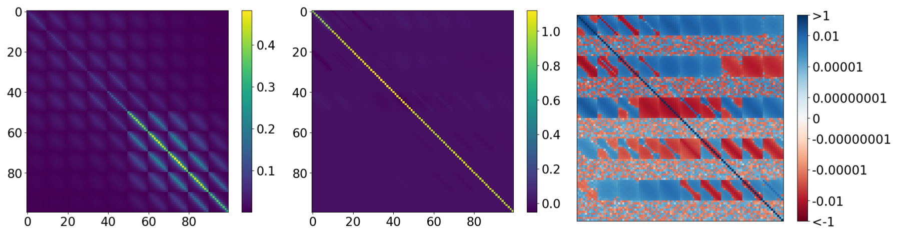

Fig. 8: The predicted inverse matrix (first panel), the result of $\mathbf{G}^{-1}\mathbf{G} \approx \mathbf{I}$ (second panel), and the logarithmic zoom towards the approximation of the inverse matrix (third panel). The computations for the computational domain with diffusion coefficients in two layers $\{10, 1\}$, span over $2/5$ and $3/5$ of the domain.

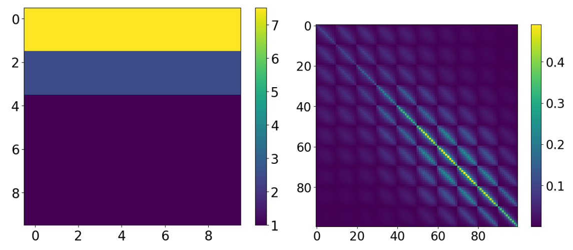

Fig. 9: The computational domain with diffusion coefficients in three layers $\{10, 2.5, 1\}$, span of $1/5$, $1/5$, and $3/5$ of the domain (left panel) and the calculated inverse of the Gram matrix $\mathbf{G}^{-1}$ (right panel).

5 Conclusions

We have shown that it is possible to efficiently train the neural network the inverse of the Gram matrix. This inverse of the Gram matrix can be applied for the computation of the matrix of coefficients of the optimal test functions $\mathbf{W} = \mathbf{G}^{-1}\mathbf{B}$, and this matrix is independent on the right-hand side vector F. Additionally, in the novel Robust Variational Physics Informed Neural Network

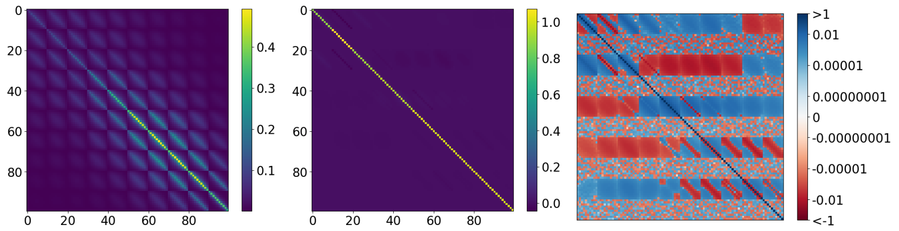

Fig. 10: The predicted inverse matrix (first panel), the result of $\mathbf{G}^{-1}\mathbf{G} \approx \mathbf{I}$ (second panel), and the logarithmic zoom towards the approximation of the inverse matrix (third panel). The computations for the computational domain with diffusion coefficients in three layers $\{10, 2.5, 1\}$, span of $1/5$, $1/5$, and $3/5$ of the domain.

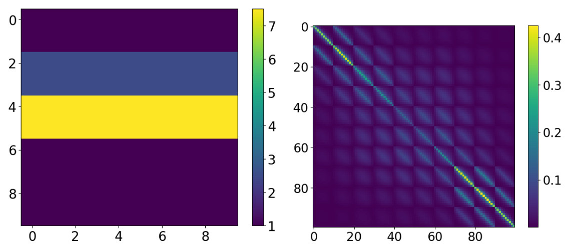

Fig. 11: The computational domain with diffusion coefficients in four layers $\{1, 2.5, 7, 1\}$ span over $1/5$, $1/5$, $1/5$ and $2/5$ of the domain (left panel) and the calculated inverse of the Gram matrix $\mathbf{G}^{-1}$ (right panel).

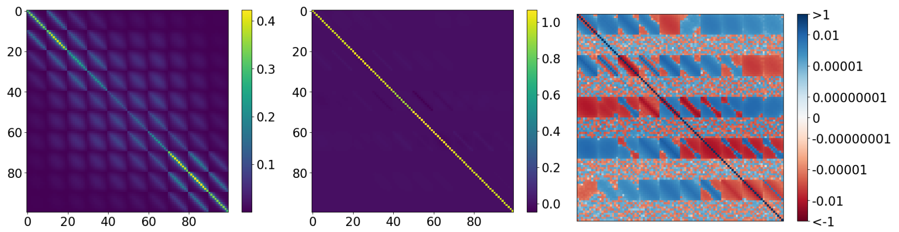

Fig. 12: The predicted inverse matrix (first panel), the result of $\mathbf{G}^{-1}\mathbf{G} \approx \mathbf{I}$ (second panel), and the logarithmic zoom towards the approximation of the inverse matrix (third panel). The computations for the computational domain with diffusion coefficients in four layers $\{1, 2.5, 7, 1\}$ span over $1/5$, $1/5$, $1/5$ and $2/5$ of the domain.

method [14] this matrix can be employed to obtain the robust loss function, by multiplication of the residual vectors, namely $RES^T(u_\theta)\mathbf{G}^{-1}RES(u_\theta)$. Similar inverse of the Gram matrix can be also employed for making the loss functon of the collocation-based RVPINN robust [10]. The inverse of the Gram matrix ob-

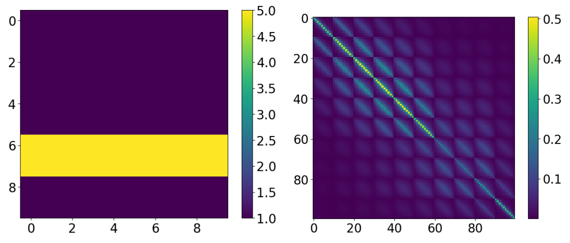

Fig. 13: The computational domain with diffusion coefficients in three layers $\{1, 2.5, 1\}$, span over 3/5, 1/5 and 1/5 of the domain (left panel) and the calculated inverse of the Gram matrix $\mathbf{G}^{-1}$ (right panel).

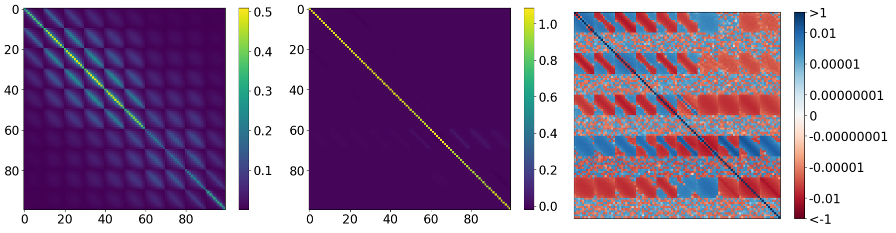

Fig. 14: The predicted inverse matrix (first panel), the result of $\mathbf{G}^{-1}\mathbf{G} \approx \mathbf{I}$ (second panel), and the logarithmic zoom towards the approximation of the inverse matrix (third panel). The computational domain with diffusion coefficients in three layers $\{1, 2.5, 1\}$, span over 3/5, 1/5 and 1/5 of the domain.

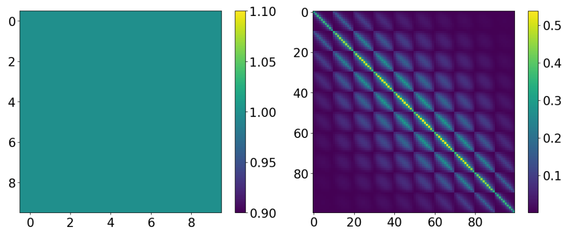

Fig. 15: The computational domain with uniform diffusion coefficient $\epsilon = 1$ (left panel) and the calculated inverse of the Gram matrix $\mathbf{G}^{-1}$ (right panel).

tained from neural networks allows obtaining high numerical accuracy results, as presented in Figure 17 for the Petrov-Galerkin formulation. There are some local variations of the concentration field resulting from local layers of the diffusion. The presented approach can be generalizable to other convection-diffusion problems and potentially other classes of PDEs beyond the isogeometric formulation considered here. The key requirement is the availability of representative training datasets that capture the essential variability of the Gram matrix inverses for the new problem class. Adapting our methodology to other Galerkin formu-

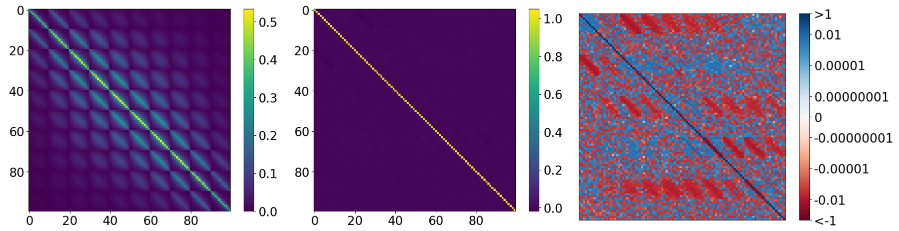

Fig. 16: The predicted inverse matrix (first panel), the result of $\mathbf{G}^{-1}\mathbf{G} \approx \mathbf{I}$ (second panel), and the logarithmic zoom towards the approximation of the inverse matrix (third panel). The computational domain with uniform diffusion coefficient $\epsilon = 1$.

lations is straightforward in principle, as the DNN-based approximation does not fundamentally depend on the specific basis type or discretization method, provided the Gram matrices exhibit similar structural properties. Extending the proposed method to other PDE formulations involves primarily generating suitable training data sets from the target PDE and retraining the neural network accordingly.

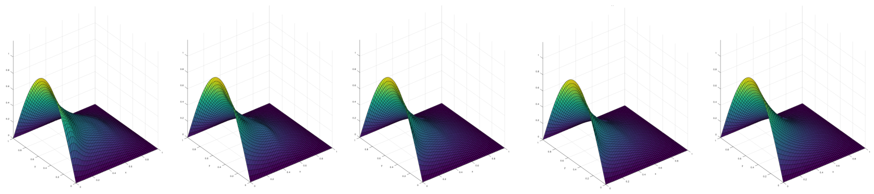

Fig. 17: The solutions of the advection-diffusion problem with ϵ maps from Figures 5, 7, 9, 11, 13, and 15, obtained with Petrov-Galerkin formulation using the inverse of the Gram matrix as provided by the neural network.

6 Acknowledgment

This work was supported by the program "Excellence initiative - research university" for the AGH University of Krakow.

References

1. Calo, V.M., Collier, N.O., Niemi, A.H.: Analysis of the discontinuous petrov–galerkin method with optimal test functions for the reissner–mindlin plate bending model. Computers & Mathematics with Applications **66**(12), 2570–2586

(2014). https://doi.org/https://doi.org/10.1016/j.camwa.2013.07.012, https://www.sciencedirect.com/science/article/pii/S0898122113004409

2. Calo, V., Łoś, M., Deng, Q., Muga, I., Paszyński, M.: Isogeometric residual minimization method (igrm) with direction splitting preconditioner for stationary advection-dominated diffusion problems. Computer Methods in Applied Mechanics and Engineering **373**, 113214 (2021). https://doi.org/https://doi.org/10.1016/j.cma.2020.113214, https://www.sciencedirect.com/science/article/pii/S0045782520303996
3. Cottrell, J., Hughes, T., Reali, A.: Studies of refinement and continuity in isogeometric structural analysis. Computer Methods in Applied Mechanics and Engineering **196**(41), 4160–4183 (2007). https://doi.org/https://doi.org/10.1016/j.cma.2007.04.007, https://www.sciencedirect.com/science/article/pii/S0045782507001703
4. Demkowicz, L., Gopalakrishnan, J.: A class of discontinuous petrov–galerkin methods. part i: The transport equation. Computer Methods in Applied Mechanics and Engineering **199**(23), 1558–1572 (2010). https://doi.org/https://doi.org/10.1016/j.cma.2010.01.003, https://www.sciencedirect.com/science/article/pii/S0045782510000125
5. Demkowicz, L., Kurtz, J., Pardo, D., Paszynski, M., Rachowicz, W., Zdunek, A.: Computing with hp-adaptive finite element method. Vol. I. One and Two Dimensional Elliptic and Maxwell Problems. Chapmann & Hall/CRC Applied Mathematics & Nonlinear Science (2006)
6. Demkowicz, L.F., Gopalakrishnan, J.: An overview of the discontinuous petrov galerkin method. Recent Developments in Discontinuous Galerkin Finite Element Methods for Partial Differential Equations: 2012 John H Barrett Memorial Lectures pp. 149–180 (2014)
7. Hughes, T., Cottrell, J., Bazilevs, Y.: Isogeometric analysis: Cad, finite elements, nurbs, exact geometry and mesh refinement. Computer Methods in Applied Mechanics and Engineering **194**(39), 4135–4195 (2005). https://doi.org/https://doi.org/10.1016/j.cma.2004.10.008, https://www.sciencedirect.com/science/article/pii/S0045782504005171
8. Kharazmi, E., Zhang, Z., Karniadakis, G.E.: hp-vpinns: Variational physics-informed neural networks with domain decomposition. Computer Methods in Applied Mechanics and Engineering **374**, 113547 (2021). https://doi.org/https://doi.org/10.1016/j.cma.2020.113547, https://www.sciencedirect.com/science/article/pii/S0045782520307325
9. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. CoRR **abs/1412.6980** (2014), https://api.semanticscholar.org/CorpusID:6628106
10. Łoś, M., Służalec, T., Maczuga, P., Vilkha, A., Uriarte, C., Paszyński, M.: Collocation-based robust variational physics-informed neural networks (crvpinn). arXiv:2401.02300v3 [cs.LG] (2024)
11. Niemi, A.H., Bramwell, J.A., Demkowicz, L.F.: Discontinuous petrov–galerkin method with optimal test functions for thin-body problems in solid mechanics. Computer Methods in Applied Mechanics and Engineering **200**(9), 1291–1300 (2011). https://doi.org/https://doi.org/10.1016/j.cma.2010.10.018, https://www.sciencedirect.com/science/article/pii/S0045782510002963
12. Paszyński, M., Grzeszczuk, R., Pardo, D., Demkowicz, L.: Deep learning driven self-adaptive hp finite element method. In: Paszynski, M., Kranzlmüller, D., Krzhizhanovskaya, V.V., Dongarra, J.J., Sloot, P.M.A. (eds.) Computational Science – ICCS 2021. pp. 114–121. Springer International Publishing, Cham (2021)

13. Roberts, N.V., Bui-Thanh, T., Demkowicz, L.: The dpg method for the stokes problem. Computers & Mathematics with Applications **67**(4), 966–995 (2014). https://doi.org/https://doi.org/10.1016/j.camwa.2013.12.015, https://www.sciencedirect.com/science/article/pii/S0898122113007116, high-order Finite Element Approximation for Partial Differential Equations
14. Rojas, S., Maczuga, P., Muñoz-Matute, J., Pardo, D., Paszyński, M.: Robust variational physics-informed neural networks. Computer Methods in Applied Mechanics and Engineering **425**, 116904 (2024). https://doi.org/https://doi.org/10.1016/j.cma.2024.116904, https://www.sciencedirect.com/science/article/pii/S0045782524001609
15. Służalec, T., Dobija, M., Paszyńska, A., Muga, I., Łoś, M., Paszyński, M.: Automatic stabilization of finite-element simulations using neural networks and hierarchical matrices. Computer Methods in Applied Mechanics and Engineering **411**, 116073 (2023)
16. Służalec, T., Paszyński, M.: Fast solver for advection dominated diffusion using residual minimization and neural networks. In: Mikyška, J., de Mulatier, C., Paszynski, M., Krzhizhanovskaya, V.V., Dongarra, J.J., Sloot, P.M. (eds.) Computational Science – ICCS 2023. pp. 517–531. Springer Nature Switzerland, Cham (2023)
17. Służalec, T., Grzeszczuk, R., Rojas, S., Dzwinel, W., Paszyński, M.: Quasi-optimal hp-finite element refinements towards singularities via deep neural network prediction. Computers & Mathematics with Applications **142**, 157–174 (2023). https://doi.org/https://doi.org/10.1016/j.camwa.2023.04.023, https://www.sciencedirect.com/science/article/pii/S0898122123001591
18. Łoś, M., Muñoz-Matute, J., Muga, I., Paszyński, M.: Isogeometric residual minimization method (igrm) with direction splitting for non-stationary advection–diffusion problems. Computers & Mathematics with Applications **79**(2), 213–229 (2020). https://doi.org/https://doi.org/10.1016/j.camwa.2019.06.023, https://www.sciencedirect.com/science/article/pii/S0898122119303268