Available online at www.sciencedirect.com
ScienceDirect
Comput. Methods Appl. Mech. Engrg. 411 (2023) 116073
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

Automatic stabilization of finite-element simulations using neural networks and hierarchical matrices

Tomasz Służalec[a], Mateusz Dobija[a], Anna Paszyńska[a], Ignacio Muga[b], Marcin Łoś[c], Maciej Paszyński[c,*]

[a] *Jagiellonian University, Kraków, Poland*
[b] *Pontificia Universidad Católica of Valparaíso, Chile*
[c] *AGH University, Kraków, Poland*

Received 26 January 2023; received in revised form 15 April 2023; accepted 15 April 2023
Available online 3 May 2023

Abstract

Petrov–Galerkin formulations with optimal test functions allow for the stabilization of finite element simulations. In particular, given a discrete trial space, the optimal test space induces a numerical scheme delivering the best approximation in terms of a problem-dependent energy norm. This ideal approach has two shortcomings: first, we need to explicitly know the set of optimal test functions; and second, the optimal test functions may have large supports inducing expensive dense linear systems. A concise proposal on how to overcome these shortcomings has been raised during the last decade by the Discontinuous Petrov–Galerkin (DPG) methodology. However, DPG has also some limitations and difficulties: the method requires ultraweak variational formulations, obtained through a hybridization process, which is not trivial to implement at the discrete level.

Our motivation is to offer a simpler alternative for the case of parametric PDEs, which can be used with any variational formulation. Indeed, parametric families of PDEs are an example where it is worth investing some (offline) computational effort to obtain stabilized linear systems that can be solved efficiently in an online stage, for a given range of parameters. Therefore, as a remedy for the first shortcoming, we explicitly compute (offline) a function mapping any PDE parameter, to the matrix of coefficients of optimal test functions (in some basis expansion) associated with that PDE parameter. Next, as a remedy for the second shortcoming, we use the low-rank approximation to hierarchically compress the (non-square) matrix of coefficients of optimal test functions. In order to accelerate this process, we train a neural network to learn a critical bottleneck of the compression algorithm (for a given set of PDE parameters). When solving *online* the resulting (compressed) Petrov–Galerkin formulation, we employ a GMRES iterative solver with inexpensive matrix–vector multiplications thanks to the low-rank features of the compressed matrix. We perform experiments showing that the full online procedure is as fast as an (unstable) Galerkin approach. We illustrate our findings by means of 2D–3D Eriksson–Johnson problems, together with 2D Helmholtz equation.

Keywords: Petrov–Galerkin method; Optimal test functions; Parametric PDEs; Automatic stabilization; Neural networks; Hierarchical matrices

* Corresponding author.
E-mail address: maciej.paszynski@agh.edu.pl (M. Paszyński).

https://doi.org/10.1016/j.cma.2023.116073

1. Introduction

Unstable finite-element simulations solved with the Galerkin method (where we employ the same trial and test space) often generate incorrect numerical results with oscillations or spurious behavior. Examples of such problems are the advection-dominated diffusion equation [1] and the Helmholtz equation [2].

Petrov–Galerkin formulations[1] with optimal test functions [3] allow for automatic stabilization of the discretization of challenging PDEs. This particular Petrov–Galerkin approach is equivalent to the residual minimization (RM) method, whose applications include advection–diffusion [4,5], Navier–Stokes [6], or space–time formulations [7]. The optimal test functions in general may not be even polynomials, but they are approximated in an enriched polynomial (compatible) test space. In practise, for a fixed trial space, RM allows for stabilization by enriching such a discrete test space.[2] Alternatively, having in hand the explicit formulas for the approximated optimal test functions expanded in the enriched discrete test space, one could use the Petrov–Galerkin formulation to solve the same problem in a different manner, recovering the same discrete solution.

This last scenario has two shortcomings. The first problem is that the computation of optimal test functions is expensive. It requires solving a large system of linear equations[3] with multiple right-hand sides (one right-hand side per each basis function of the trial space). The second problem is that the optimal test functions can have global supports, and thus the Petrov–Galerkin method with optimal test functions can generate a dense matrix, expensive to solve. A concise proposal on how to overcome these shortcomings has been raised during the last decade by the Discontinuous Petrov–Galerkin (DPG) methodology [10–12]. However, DPG has also some limitations and difficulties. Namely, the method requires ultraweak variational formulations, obtained through a hybridization process, which is not trivial to implement at the discrete level.

Our motivation is to offer a simpler alternative to the aforementioned shortcomings, for the particular case of parametric PDEs, which can be used with any variational formulation. Indeed, parametric families of PDEs are an example where it is worth investing some (offline) computational effort, to obtain stabilized linear systems that can be solved efficiently in an online stage, for a given range of parameters. Therefore, in the context of a parametric family of PDEs, and as a remedy for the first shortcoming, we explicitly compute (offline) a function mapping any PDE parameter, to the matrix of coefficients of optimal test functions (expanded in the basis of the enriched discrete test space) associated with that PDE parameter. We emphasize that this last procedure is independent of particular right-hand side sources and/or prescribed boundary data of the PDE family.

The obtained matrix $\mathbb{W}$ of optimal test functions coefficients is dense. The Petrov–Galerkin method induces a linear system of the form $\mathbb{B}^T\mathbb{W}\,x = (L^T\mathbb{W})^T$, where L is a right-hand side vector and $\mathbb{B}$ is the matrix associated with the bilinear form of the underlying PDE. To avoid dense matrix computations, we compress $\mathbb{W}$ using the approach of hierarchical matrices [13,14]. Having the matrix $\mathbb{W}$ compressed into a hierarchical matrix $\mathbb{H}$, we employ the GMRES method [15], which involves computations of the residual $R := \mathbb{B}^T\mathbb{H}\,x - (L^T\mathbb{H})^T$ and the hierarchical matrix enables matrix–vector multiplication of $\mathbb{H}\,x$ and $L^T\mathbb{H}$ in a quasi-linear computational cost.

However, compressing the matrix $\mathbb{W}$ for each PDE-parameter is an expensive procedure. Thus, with the help of an artificial neural network, we train (offline) the critical bottleneck of the compression algorithm. We obtain a stabilized method with the additional quasi-linear cost resulting from matrix–vector multiplications within the GMRES solver. From our numerical results with the Eriksson–Johnson and Helmholtz model problems, this cost of stabilization[4] is of the same order as the cost of the solution of the original Galerkin problem without the stabilization.

The hierarchical matrices have been used for preconditioning of multi-grid solvers in context of higher-order FEM in [16]. They have been also employed to speed up the isogeometric boundary method computations [17]. In this paper we employ GMRES solver with the hierarchical matrix working like a preconditioner to stabilize the solution. There are alternative preconditioners for GMRES employed to solve the Helmholtz problem. In [18] the shifted-Laplace preconditioner for the GMRES solver is employed. The paper [19] extends this Helmholtz problem preconditioner into the complex shifted Laplacian combined with deflation techniques. It has been also shown that the refined isogeometric analysis can improve the computational cost of iterative and multi-grid solvers [20]. The

[1] i.e., trial and test spaces are not equal, although they share the same dimension.

[2] Contrary to standard stabilization methods like SUPG [8,9], there are no special stabilization terms modifying the weak formulation.

[3] As large as the dimension of the enriched discrete test space.

[4] i.e., the cost of compression of the hierarchical matrix and the cost of GMRES with hierarchical matrix multiplication by a vector.

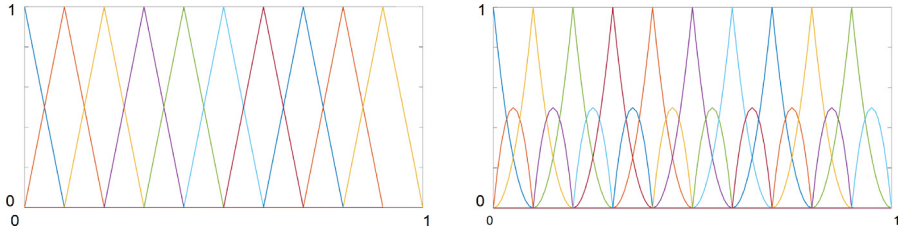

Fig. 1. Discrete spaces U_h (left) and V_h (right).

application of direct solvers [21–23] possibly combined with refined isogeometric analysis [24] for the Petrov–Galerkin formulation with optimal test functions is limited due to the density of the matrix of coefficients of the optimal test functions. However, after the hierarchical compression of the matrix, it may be possible to invert the system using a direct method with a quasi-linear computational cost [25] due to the properties of the hierarchical matrices. This, however, requires further investigation and is not a subject of this paper.

The idea of performing a *double discretization* has been already investigated in the context of the *surrogate matrix* technique [26], which its main goal is to decrease the computational cost by reducing the finite element assembly process, to the evaluation of a small set of functions which can be locally approximated by polynomials. Applications include linear elasticity and nonlinear diffusion problems [27]; isogeometric analysis for the Stokes flow problem [28]; Helmholtz equation, linear elastodynamics and non-linear hyperelastic wave propagation problems [29]. The surrogate matrix technique is another interesting approach, alternative to hierarchical matrices. The question whether this method can be applied for compression of the matrix of coefficients of the optimal test functions is an interesting open problem that may be investigated in a future work.

1.1. One-dimensional illustration of stabilization

Let us illustrate how we stabilize difficult computational problems by means of a one-dimensional example for the advection-dominated diffusion model.

Given $0 < \epsilon \leq 1$, consider the following differential equation:

$$\begin{cases} -\epsilon u'' + u' = 0 & \text{in } (0,1)\,; \\ -\epsilon u'(0) + u(0) = 1 & \text{and } u(1) = 0. \end{cases} \tag{1}$$

In weak-form, Eq. (1) translates into find $u \in H^1_{0)}(0,1) := \{v \in H^1(0,1) : v(1) = 0\}$ such that:

$$\epsilon \int_0^1 u'v' + \int_0^1 u'v + u(0)v(0) = v(0)\,, \qquad \forall v \in H^1_{0)}(0,1). \tag{2}$$

We define discrete spaces U_h and V_h as depicted in Fig. 1. Given a regular mesh, U_h will be the space of piecewise linear and continuous functions; while V_h will be the space of piecewise quadratics and continuous functions. From one side, we discretize formulation (2) using a standard Galerkin method where trial and test spaces are equal to V_h. On the other side, we discretize (2) by means of a residual minimization (RM) technique that uses U_h as the trial space, and V_h as the test space. In the residual minimization method we solve a saddle-point problem. Fig. 2 compares the discrete solutions delivered by these two methods for different values of ϵ, and different meshes parametrized by the number of elements n. We observe a superior performance of the residual minimization method, even though the approximation (trial) space used is poorer than that of the Galerkin method.

1.2. Outline of the paper

The structure of the paper is the following. Section 2 contains all the theoretical ingredients needed to understand our approach. Namely, Petrov–Galerkin formulations with optimal test functions (Section 2.1); optimal test functions for an affine family of PDEs (Section 2.2); the hierarchical compression of the optimal test functions matrix of coefficients (Section 2.3); the fast hierarchical matrix–vector multiplication and related fast implementation of the GMRES solver (Section 2.4); and the neural networks acceleration of the hierarchical matrix compression

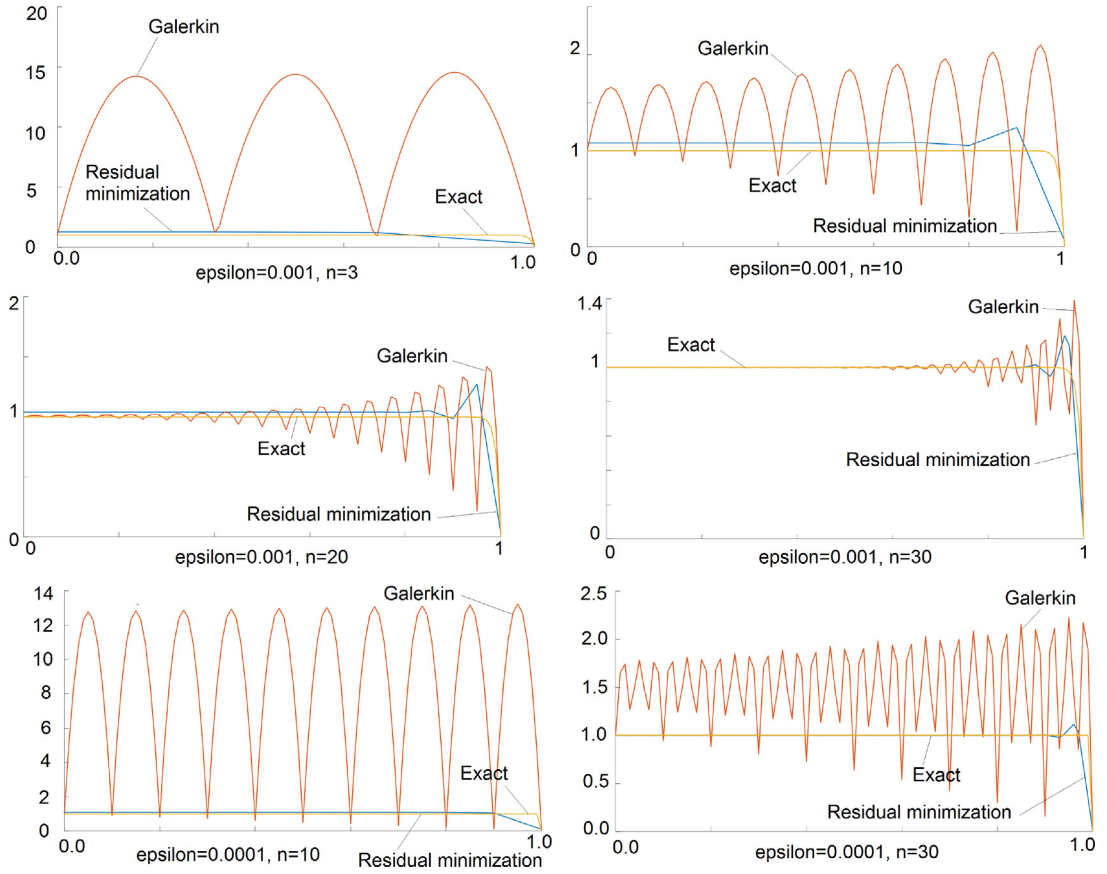

Fig. 2. Comparing the exact solution with the Galerkin method (where trial and test are quadratic B-splines with C^0 separators) and the residual minimization method (with linear B-splines for trial and quadratic B-splines with C^0 separators for test).

(Section 2.5). Next, in Section 3 we apply our automatic stabilization procedure to well-known unstable parametric model problems. First, for the two-dimensional Eriksson–Johnson model problem (Section 3.1); and next, for a two-dimensional Helmholtz equation (Section 3.2). We write our conclusions in Section 4. All the pseudo-code algorithms needed to follow our methodology have been shifted to Appendix A.

2. Theoretical ingredients

2.1. Petrov–Galerkin formulations with optimal test functions

Let U and V be Hilbert spaces. We consider a general PDE variational formulation, which is to find $u \in U$ such that

$$b(u, v) = \ell(v)\,, \quad \forall v \in V\,, \tag{3}$$

where $b : U \times V \to \mathbb{R}$ is a continuous bilinear form, and $\ell : V \to \mathbb{R}$ is a continuous linear functional (i.e., $\ell \in V'$, the dual space of V).

The dual space V' has a norm inherited by the norm of V. Indeed, if $(\cdot, \cdot)_V$ denotes the inner-product of the Hilbert space V, then these norms are given by the following expressions:

$$V \ni v \mapsto \|v\|_V := \sqrt{(v, v)_V} \qquad \text{and} \qquad V' \ni f \mapsto \|f\|_{V'} := \sup_{\|v\|_V = 1} f(v).$$

We will assume well-posedness of problem (3), which translates into the well-known inf-sup conditions (see, e.g., [30, Theorem 2.6]):

$$\exists \gamma > 0 \text{ such that } \|b(w,\cdot)\|_{V'} \geq \gamma \|w\|_U\,, \quad \forall w \in U; \tag{4a}$$

$$\{v \in V : b(w,v) = 0\,, \forall w \in U\} = \{0\}. \tag{4b}$$

Notice that the continuity assumption on the bilinear form $b(\cdot,\cdot)$, also implies the existence of a constant $M \geq \gamma$ such that:

$$\|b(w,\cdot)\|_{V'} \leq M\|w\|_U\,, \quad \forall w \in U. \tag{5}$$

Given a discrete finite-element space $U_h \subset U$, a natural candidate to approximate the solution $u \in U$ to problem (3) is the residual minimizer

$$u_h = \underset{w_h \in U_h}{\operatorname{argmin}} \|b(w_h,\cdot) - \ell(\cdot)\|_{V'}\,. \tag{6}$$

Indeed, combining (4a), (6), and (5), the residual minimizer automatically satisfies the quasi-optimality property:

$$\gamma\|u_h - u\|_U \leq \|b(u_h,\cdot) - \ell(\cdot)\|_{V'} = \inf_{w_h \in U_h} \|b(w_h,\cdot) - \ell(\cdot)\|_{V'} \leq M \inf_{w_h \in U_h} \|w_h - u\|_U.$$

Thus, the residual minimizer inherits stability properties from the continuous problem.

It is well-known (see, e.g., [31]) that the saddle-point formulation of residual minimization (6) becomes the mixed problem that aims to find $u_h \in U_h$, and a residual representative $r \in V$, such that:

$$(r,v)_V - b(u_h,v) = -\ell(v), \qquad \forall v \in V, \tag{7a}$$

$$b(w_h,r) \qquad\quad = 0, \qquad \forall w_h \in U_h. \tag{7b}$$

Although it seems harmless, the mixed problem (7) is still infinite-dimensional in the test space V. To obtain a computable version of it, we introduce a discrete test space $V_h \in V$ that turns (7) into a fully discrete problem that aims to find $u_h \in U_h$,[5], and a discrete residual representative $r_h \in V_h$, such that:

$$(r_h,v_h)_V - b(u_h,v_h) = -\ell(v_h), \qquad \forall v \in V_h, \tag{8a}$$

$$b(w_h,r_h) \qquad\quad = 0, \qquad \forall w_h \in U_h, \tag{8b}$$

Problem (8) corresponds to the saddle-point formulation of a discrete-dual residual minimization, in which the dual norm $\|\cdot\|_{V'}$ in (6) is replaced by the discrete-dual norm $\|\cdot\|_{V_h'}$. Well-posedness and stability of (8) has been extensively studied in [32] and depends on a *Fortin compatibility* condition between the discrete spaces U_h and V_h. Moreover, once this condition is fulfilled (and in order to gain stability), it is possible to enrich the test space V_h without changing the trial space U_h. Obviously, this process will enlarge the linear system (8). Nevertheless, we know that there is an equivalent linear system of the same size of the trial space, delivering the same $u_h \in U_h$ solving (8). This is known as the Petrov–Galerkin method with optimal test functions, which we describe below. Our goal will be to make this (generally impractical) method practical.[6]

Let us introduce now the concept of optimal test functions. For each $w_h \in U_h$, the optimal test function $Tw_h \in V_h$ is defined as the Riesz representative of the functional $b(w_h,\cdot) \in V_h'$, i.e.,

$$(Tw_h,v_h)_V = b(w_h,v_h)\,, \quad \forall v_h \in V_h. \tag{9}$$

If we test Eq. (8a) with optimal test functions Tw_h, then using (9) and (8b), we arrive to the following Petrov–Galerkin system with optimal test functions:

$$\begin{cases} \text{Find } u_h \in U_h \text{ such that} \\ b(u_h, Tw_h) = \ell(Tw_h)\,, \quad \forall w_h \in U_h\,. \end{cases} \tag{10}$$

In order to explicit a matrix expression for (10), let us set $U_h := \operatorname{span}\{w_1,\ldots,w_n\}$ and $V_h := \operatorname{span}\{v_1,\ldots,v_m\}$, with $m > n$. Consider the matrix $\mathbb{B}$ linked to the bilinear form $b(\cdot,\cdot)$ such that its (i,j)-entry is $\mathbb{B}_{ij} = b(w_j,v_i)$.

[5] Notice the abuse of notation. This new $u_h \in U_h$ solution of (8) does not equals the exact residual minimizer solution of (6) or equivalently (7).

[6] We emphasize that DPG is another methodology focused on the same goal; see [10].

Analogously, we consider the Gram matrix $\mathbb{G}$ linked to the inner product $(\cdot,\cdot)_V$ such that $\mathbb{G}_{ki} = (v_k, v_i)_V$. The optimal test space is defined as $V_h^{\text{opt}} := \text{span}\{Tw_1, \ldots, Tw_n\}$. Thus, using (9), we observe that the matrix containing the coefficients of optimal test functions when expanded in the basis of V_h is $\mathbb{W} := \mathbb{G}^{-1}\mathbb{B}$.[7] Moreover, the Petrov–Galerkin system (10) becomes

$$\mathbb{B}^T\mathbb{W}x = (L^T\mathbb{W})^T, \tag{11}$$

where the vector x contains the coefficients of the expansion of u_h in the basis of U_h; $L^T := [\ell(v_1) \ldots \ell(v_m)]$; and we have used the fact that $\mathbb{G}^T = \mathbb{G}$. Therefore, if we aim to solve the Petrov–Galerkin linear system (11) iteratively, an optimized matrix–vector multiplication to perform $\mathbb{W}x$ and $L^T\mathbb{W}$ becomes critical. Section 2.3 is devoted to study the hierarchical compression of $\mathbb{W}$, which allows for fast vector–matrix multiplications.

2.2. Optimal test functions for an affine family of parametric PDEs

Assume we want to solve parametric PDEs in variational form, i.e.,

$$\text{Find} \quad u_\mu \in U \quad \text{such that} \quad b_\mu(u_\mu, v) = \ell_\mu(v), \quad \forall v \in V,$$

where for each set of parameters $\mu \in \mathcal{P} \subset \mathbb{R}^d$, the bilinear form $b_\mu(\cdot,\cdot)$ is continuous and inf-sup stable, with constants that may depend on μ. Moreover, assume that $b_\mu(\cdot,\cdot)$ has the following affine decomposition:

$$b_\mu(\cdot,\cdot) = b_0(\cdot,\cdot) + \sum_{k=1}^{d} \theta_k(\mu) b_k(\cdot,\cdot),$$

where $\theta_k : \mathcal{P} \to \mathbb{R}$ and $b_k : U \times V \to \mathbb{R}$ are accessible and easy to compute. When trial and test spaces are discretized, the bilinear form $b_\mu(\cdot,\cdot)$ induces a matrix of the form:

$$\mathbb{B}_\mu = \mathbb{B}_0 + \sum_{k=1}^{d} \theta_k(\mu)\mathbb{B}_k.$$

Thus, the matrix of coefficients of optimal test functions $\mathbb{W}_\mu := \mathbb{G}^{-1}\mathbb{B}_\mu$ becomes in this case:

$$\mathbb{W}_\mu = \mathbb{G}^{-1}\mathbb{B}_0 + \sum_{k=1}^{d} \theta_k(\mu)\mathbb{G}^{-1}\mathbb{B}_k\,. \tag{12}$$

The upcoming Eq. (17) shows the particular form that expression (12) gets for the Eriksson–Johnson model problem, where knowing $\mathbb{G}^{-1}\mathbb{B}_0$ and $\mathbb{G}^{-1}\mathbb{B}_1$ implies the knowledge of $\mathbb{W}_\epsilon$ for any $\epsilon > 0$.

2.3. Hierarchical compression of the optimal test functions matrix of coefficients

Hierarchical matrices were introduced by Hackbusch [13]. The main idea of the hierarchical compression of a matrix is to store the matrix in a tree-like structure, where:

- the root node corresponds to the whole matrix;
- the root node has some number of sons (in our approach 4 sons) corresponding to submatrices of the main matrix;
- each node can have sons (in our approach 4 sons) corresponding to submatrices (blocks), or it can be a leaf representing the corresponding matrix (block);
- each leaf stores its associated matrix in the SVD compressed form or as a zero matrix;
- at each node, the decision about storing the block in SVD form, or either dividing the block into submatrices, depends on an admissibility condition of the block.

Exemplary hierarchical compression of the matrix in a form of a tree is presented in Fig. 3; while the algorithm for compression of the matrix into the hierarchical matrix format is presented in Algorithm 1.

[7] In DPG, $\mathbb{G}^{-1}$ is available with low computational effort ($\mathbb{G}$ is block-diagonal), at the price of working with a hybrid version of the bilinear form $b(\cdot,\cdot)$, which introduces new variables on the skeleton of the underlying finite element mesh.

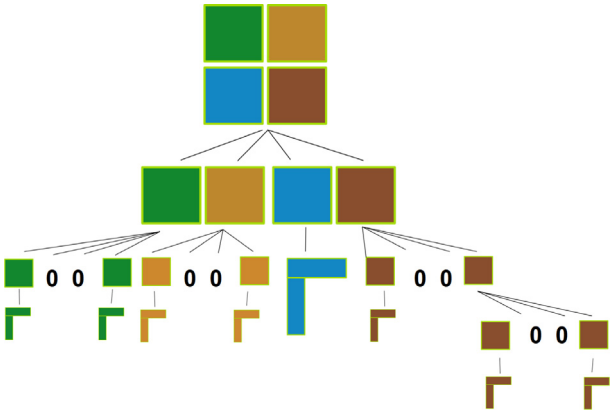

Fig. 3. Hierarchical compression of a matrix.

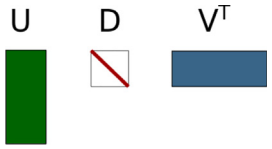

Fig. 4. Reduced singular value decomposition.

The admissibility condition controls the process of creation of the tree, it allows us to decide if the matrix should be divided (or not) into submatrices. In our case, the admissibility condition is established using the following criteria:

1. The size of the matrix: if the matrix is bigger than a pre-defined maximal admissible size $l >> 1$, then the matrix should be divided into submatrices;
2. The first r singular values: if the $r+1$ singular value is greater than a pre-defined threshold $\delta > 0$, then the matrix should be divided into submatrices.

In the leaves of the tree, we perform a reduced Singular Value Decomposition (rSVD). A reduced singular value decomposition of a $(n \times m)$-matrix[8] $\mathbb{M}$ of rank k is a factorization of the form

$$\mathbb{M} = \mathbb{U}\,\mathbb{D}\,\mathbb{V}^T,$$

with unitary matrices $\mathbb{U} \in \mathbb{R}^{n\times k}$ and $\mathbb{V} \in \mathbb{R}^{m\times k}$, and a diagonal matrix $\mathbb{D} \in \mathbb{R}^{k\times k}$ where the diagonal entries are $\mathbb{D}_{11} \geq \mathbb{D}_{22} \geq \cdots \geq \mathbb{D}_{kk} > 0$. (see Fig. 4). The diagonal entries of $\mathbb{D}$ are called the singular values of $\mathbb{M}$. The computational complexity of the reduced SVD is $\mathcal{O}((m+n)k^2)$. The threshold δ in the reduced SVD plays similar role as the required accuracy of convergence of the iterative solver. For example, if we require solutions with accuracy up to 6 digits, like for the Eriksson–Johnson problem with $\epsilon = 10^{-6}$, we need to select δ having order 10^{-7}. The fact that we compress the hierarchical matrix during the offline training means that we can start computations of the reduced SVD from large blocks related to patches of elements, arbitrarily selected for a given problem.

2.4. Matrix–vector multiplication with $\mathbb{H}$-matrices and GMRES solver speedup

The computational cost of matrix–vector multiplication using a compressed $\mathbb{H}$-matrix of rank r and s right-hand side vectors is $\mathcal{O}((m+n)rs)$. This is illustrated in Fig. 5(a).

The multiplication of a matrix compressed into SVD blocks is performed recursively as illustrated in Fig. 5(b). The resulting computational cost of the multiplication is again $\mathcal{O}((m+n)rs)$.

[8] Notice that we are abusing the notation and n, m are not necessarily the dimensions of the discrete trial and test spaces.

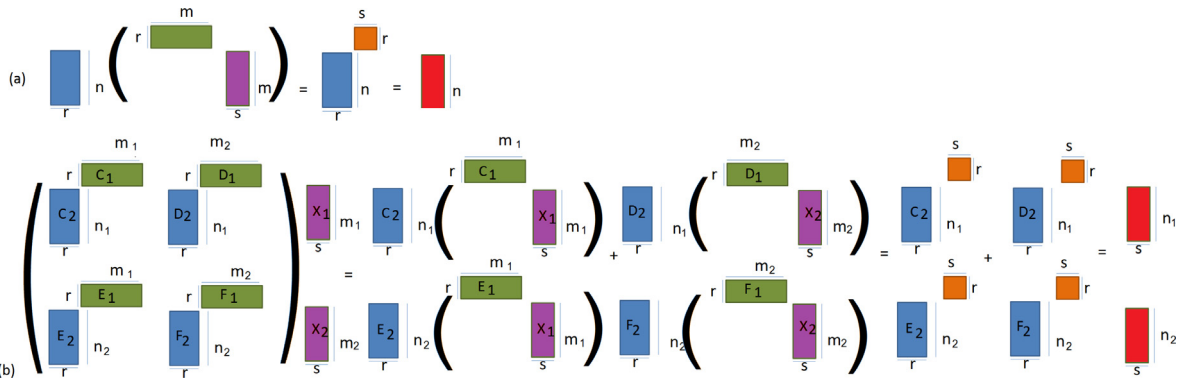

Fig. 5. (a) The complexity of matrix–vector multiplication with compressed $\mathbb{H}$-matrix of arbitrary dimension $n \times m$ and s vectors is $\mathcal{O}((m+n)rs)$. (b) We can also partition the matrix–vector multiplication into four blocks of arbitrary dimensions $n_1 \times m_1$, $n_1 \times m_2$, $n_2 \times m_1$, and $n_2 \times m_2$, where $n = n_1 + n_2, m = m_1 + m_2$. We also partition vectors into m_1 and m_2 rows. When multiplying such the four different SVD compressed blocks by a vector, we can employ the Binet matrix–vector multiplication algorithm $\begin{bmatrix} C_2 * (C_1 * X_1) + D_2 * (D_1 * X_2) \\ E_2 * (E_1 * X_1) + F_2 * (F_1 * X_2) \end{bmatrix}$. The resulting computational cost is $\mathcal{O}((m+n)rs)$.

The GMRES algorithm employed for computing the solution includes multiplications of the problem matrix by vectors (see line 1, line 4, and line 5 in Algorithm 9). These matrix–vector multiplications have a linear cost when using the $\mathbb{H}$-matrix.

2.5. Neural network learning the hierarchical matrices

The hierarchical matrix is obtained by constructing a tree with SVDs of different blocks of the full matrix. The root level corresponds to the entire matrix, and the children correspond to sub-blocks. Only the leaf nodes have blocks stored in the SVD decomposition format. The most expensive part of the compression algorithm is checking the admissibility condition. In particular, checking if a given block has r singular values smaller than δ, and whether we partition or run the SVD. The SVD data for the blocks of different sizes can be precomputed and stored in a list, see Fig. 6. From the set $\mathcal{P} \subset \mathbb{R}^d$ of PDE parameters, we can construct the neural network

$$\mathcal{P} \ni \mu \rightarrow \text{DNN}(\mu) = \{\mathbb{U}_i(\mu), \mathbb{D}_i(\mu), \mathbb{V}_i(\mu)\}_{i=1,\ldots,N_B} \tag{13}$$

where $\text{DNN}(\mu)$ is the list of learned SVDs for all N_B blocks of different dimensions. The associated loss functions are defined as the mean square error (MSE) between the trainable data and the data from the dataset, i.e.,

$$\text{MSE}(\mathbb{U}_i) := \sum_{p=1}^{N_t} \left\| \mathbb{U}_i(\mu_p) - \text{U}_i(\mu_p) \right\|_F^2 / \text{size}(\text{U}_i);$$

$$\text{MSE}(\mathbb{D}_i) := \sum_{p=1}^{N_t} \left\| \mathbb{D}_i(\mu_p) - \text{D}_i(\mu_p) \right\|_2^2 / \text{length}(\text{D}_i);$$

$$\text{MSE}(\mathbb{V}_i) := \sum_{p=1}^{N_t} \left\| \mathbb{V}_i(\mu_p) - \text{V}_i(\mu_p) \right\|_F^2 / \text{size}(\text{V}_i);$$

where $\{\text{U}_i(\mu_p), \text{D}_i(\mu_p), \text{V}_i(\mu_p)\}_{p=1,\ldots,N_t}$ denotes the exact values of the SVDs of a given block, for a sample of N_t training parameters $\{\mu_p\}_{p=1}^{N_t}$; while $\|\cdot\|_2$ and $\|\cdot\|_F$ denote the Euclidean and Frobenius norms, respectively. Notice that each block $i = 1, \ldots, N_B$ is trained in a separate way.

We emphasize that the training of the neural networks $\mathbb{U}_i(\mu)$ and $\mathbb{V}_i(\mu)$ works only if we partition the matrix $\mathbb{W}_\mu$ into blocks corresponding to the mesh dimensions.

Fig. 7 illustrates the changes of values of matrices $\{\text{U}, \text{D}, \text{V}\}$ from the SVD of one block of the matrix $\mathbb{W}_\epsilon$ computed from the two-dimensional Eriksson–Johnson problem. The horizontal axis denotes different values of ϵ (from 10^{-7} to 10^0) and the vertical axis denotes the values of the coefficients of the matrices (as functions of ϵ). In particular, we look at the entries of $\{\text{U}, \text{D}, \text{V}\}$ from the first block of the third partition level (first block from Fig. 12). We only plot the columns of U related with the first and last singular values of D under consideration, as

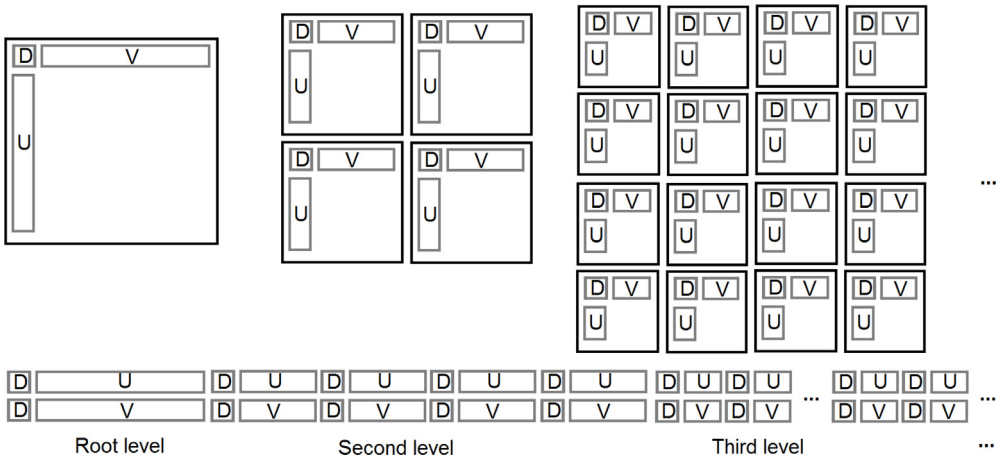

Fig. 6. Compression of reduced SVDs of matrices, starting from the root level, second level, third level, and the following levels.

well as the rows of V related to them. Neural networks are universal approximators, becoming a good choice to approximate these kind of functions.

An illustration of the architecture of the neural network used to learn singular values is depicted in Fig. 8. Such an architecture has been obtained heuristically. We assumed the number of layers varying from 3 to 5. We also randomly checked the number of neurons in the layers from the set $\{2, 4, 6, 8, 10, 12, 14, 2^4, 2^5, 2^6, 2^7, 2^8, 2^9\}$. Then, we checked the MSE error for training of one block of SVD of the matrix of coefficients of the optimal test functions. The selected neural network architecture provides small MSE loss function value. We also compared ReLU activation function with tanh. The ReLU provided better approximation error.

The input values of ϵ are logarithmically scaled before we pass them into the neural network using `np.log10(np_eps)`. We also scale them into $[-1, 1]$ using `MinMaxScaler(feature_range(-1,1))`.

3. Applications

3.1. Two-dimensional eriksson–johnson problem

Given $\Omega = (0, 1)^2 \subset \mathbb{R}^2$ and $\beta = (1, 0)$, we seek the solution of the advection–diffusion problem

$$\begin{cases} -\epsilon\, \Delta u + \beta \cdot \nabla u = 0 & \text{in } \Omega \\ u = \sin(k\pi y)\chi_{\{x=0\}} & \text{over } \partial\Omega\,, \end{cases} \tag{14}$$

where $\chi_{\{x=0\}}$ denotes the characteristic function over the inflow boundary $x = 0$. In our examples, we consider $k = 1$, or $k = 2$, but it can be an arbitrary integer. The problem is driven by the inflow Dirichlet boundary condition and develops a boundary layer of width ϵ near the outflow $x = 1$, as shown in Fig. 9.

The weak form with a general Dirichlet boundary data $g \in H^{\frac{1}{2}}(\partial\Omega)$ will be to find $u^\epsilon \in H^1(\Omega)$ such that

$$\begin{cases} \underbrace{\epsilon \displaystyle\int_\Omega \nabla u^\epsilon \cdot \nabla v + \int_\Omega (\beta \cdot \nabla u^\epsilon) v}_{\epsilon\, b_1(u^\epsilon, v) + b_0(u^\epsilon, v) =: b_\epsilon(u^\epsilon, v)} = 0, \quad \forall v \in H_0^1(\Omega), \\ u^\epsilon = g, \quad \text{over } \partial\Omega. \end{cases} \tag{15}$$

To simplify the discussion, we approximate the solution as tensor products of C^1-continuous one-dimensional B-splines basis functions $\{B_{i;p}(x)B_{j;p}(y)\}_{i,j}$ of uniform order p in all directions. This discrete trial space $U_h^p \subset H^1(\Omega)$ will be split as $U_h^p = U_{h,0}^p + U_{h,\partial\Omega}^p$, where $U_{h,0}^p \subset H_0^1(\Omega)$ contains all the basis functions vanishing at $\partial\Omega$; and $U_{h,\partial\Omega}^p$ is the complementary subspace containing the basis functions associated with boundary nodes. Our discrete solution will be $u_h^\epsilon = u_{h,0}^\epsilon + u_{h,g}^\epsilon$, where $u_{h,0}^\epsilon \in U_{h,0}^p$ is unknown and $u_{h,g}^\epsilon \in U_{h,\partial\Omega}^p$ is directly obtained using the boundary data g.

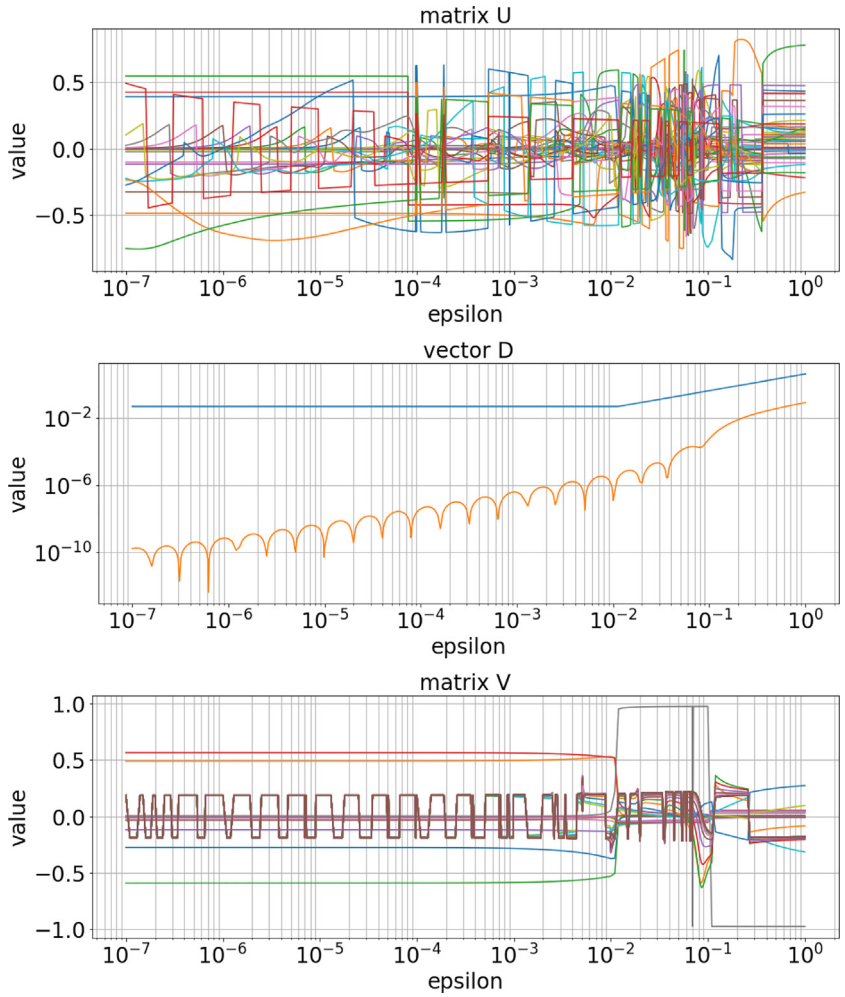

Fig. 7. Some coefficients of matrices $\{\mathrm{U}, \mathrm{D}, \mathrm{V}\}$ from the SVD algorithm executed from the first block of the matrix $\mathbb{W}_\epsilon$ at the third partition level. We only plot the columns of U related with the first and last singular values of D, as well as the rows of V related to them.

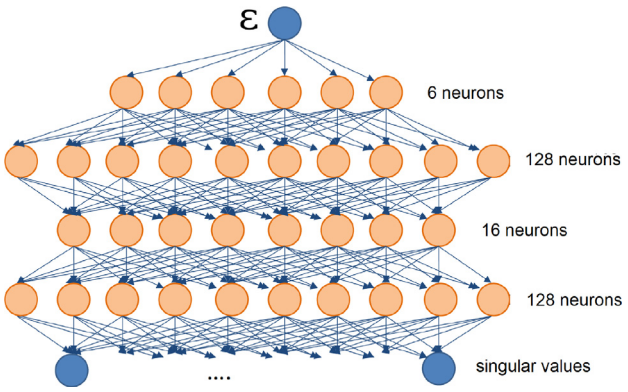

Fig. 8. The architecture of the neural network learning the singular values for blocks of the matrix. The input to the neural network is the ϵ parameter.

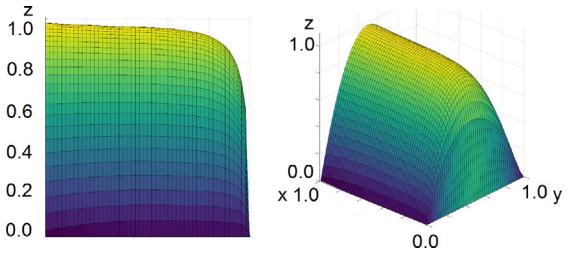

Fig. 9. Boundary layer of Eriksson–Johnson problem for $\epsilon = 0.01$.

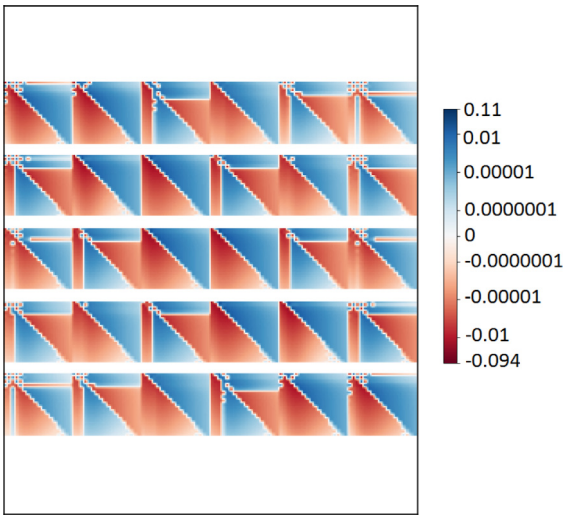

Fig. 10. The entire matrix $\mathbb{W}_\epsilon$ of the coefficients of the optimal test functions as the input of the first hierarchical level.

We build the test space using the same polynomial order p but with C^0 separators inserted between elements, which makes the test space larger than the trial space. This discrete test space will be denoted by $V^p_{h,0} \subset H^1_0(\Omega)$. The discrete residual minimization problem will be to find $u^\epsilon_{h,0} \in U^p_{h,0}$ and $r_h \in V^p_{h,0}$ such that

$$(\nabla r_h, \nabla v_h)_{L^2(\Omega)} - b_\epsilon(u^\epsilon_{h,0}, v_h) = -b_\epsilon(u^\epsilon_{h,g}, v_h)\,, \quad \forall v_h \in V^p_{h,0}\,, \tag{16a}$$

$$b_\epsilon(w_h, r_h) = 0\,, \quad \forall w_h \in U^p_{h,0}\,. \tag{16b}$$

The reduced matrix system associated with (16) takes the form:

$$\mathbb{B}^T_\epsilon \mathbb{W}_\epsilon x = (L^T \mathbb{W}_\epsilon)^T, \qquad \text{where } \mathbb{W}_\epsilon = \mathbb{G}^{-1}\mathbb{B}_\epsilon = \epsilon\, \mathbb{G}^{-1}\mathbb{B}_1 + \mathbb{G}^{-1}\mathbb{B}_0. \tag{17}$$

We train a neural network $\{\mathbb{U}(\epsilon), \mathbb{D}(\epsilon), \mathbb{V}(\epsilon)\}$ for the SVD of the entire matrix $\mathbb{W}_\epsilon$ (see Fig. 10), as well as the SVD decompositions $\{\mathbb{U}_{ij}(\epsilon), \mathbb{D}_{ij}(\epsilon), \mathbb{V}_{ij}(\epsilon)\}_{i=1,\dots,j^2}$ of sub-matrices obtained by $j \times j$ partitions of $\mathbb{W}_\epsilon$, for $j = 2, 4, 8, 16$. The successful training requires that the partitions of the matrix corresponds with the mesh dimensions. Fig. 11 shows a 2×2 partition; while Fig. 12 shows a 4×4 partition. The white parts correspond to the boundary nodes, where we have enforced the boundary conditions.

For the training of SVDs we employ the following training set of epsilons

$$\begin{aligned}\{1,\ & [9,8,7,6,5,4,3,2,1]\cdot 10^{-1},\\ & [9,8,7,6,5,4,3,2,1]\cdot 10^{-2},\\ & [9,8,7,6,5,4,3,2,1]\cdot 10^{-3},\\ & [9,8,7,6,5,4,3,2,1]\cdot 10^{-4},\\ & [9,8,7,6,5,4,3,2,1]\cdot 10^{-5},\end{aligned}$$

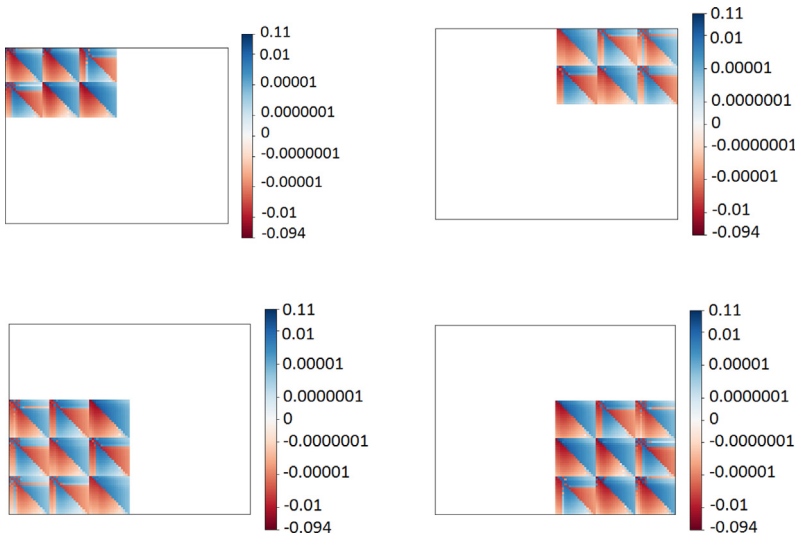

Fig. 11. The four blocks of the matrix $\mathbb{W}_\epsilon$ of the coefficients of the optimal test functions as the input of the second hierarchical level.

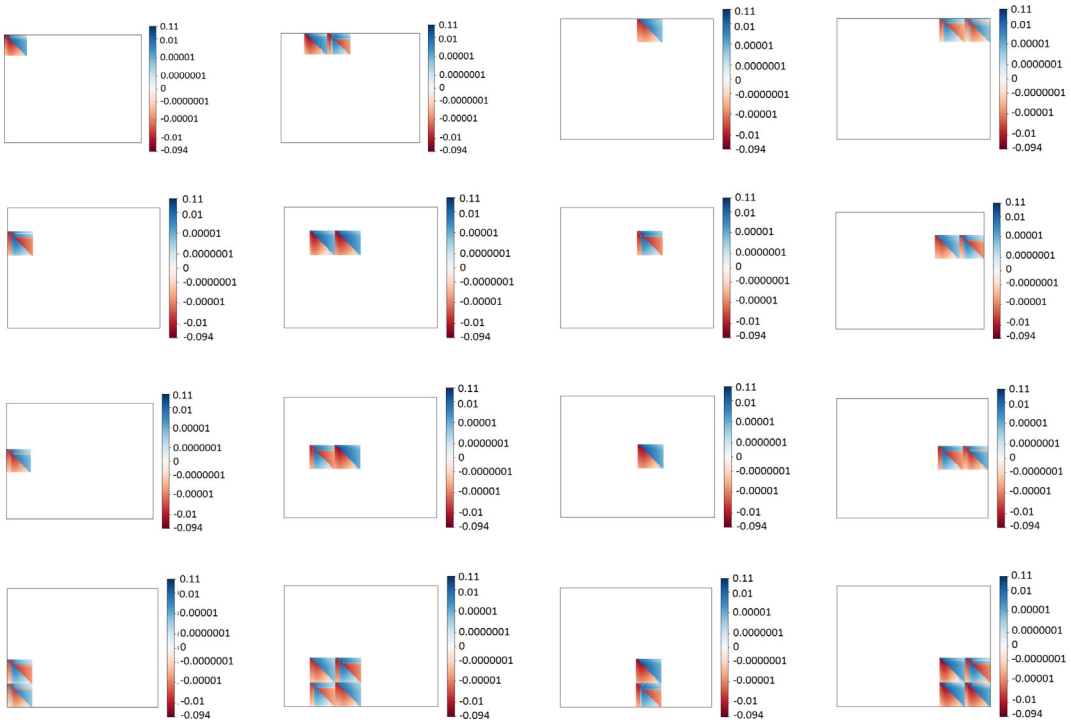

Fig. 12. The sixteen blocks of the matrix $\mathbb{W}_\epsilon$ of the coefficients of the optimal test functions as the input of the third hierarchical level.

$$[9, 8, 7, 6, 5, 4, 3, 2, 1] \cdot 10^{-6},$$
$$[9, 8, 7, 6, 5, 4, 3] \cdot 10^{-7}\}$$

Output from the neural network are the approximated coefficients of the SVD. We partitioned the data set for training and testing data: odd epsilons are used for training, while even epsilons are used for testing. The examples of convergence of the training procedures are presented in Figs. 13–14.

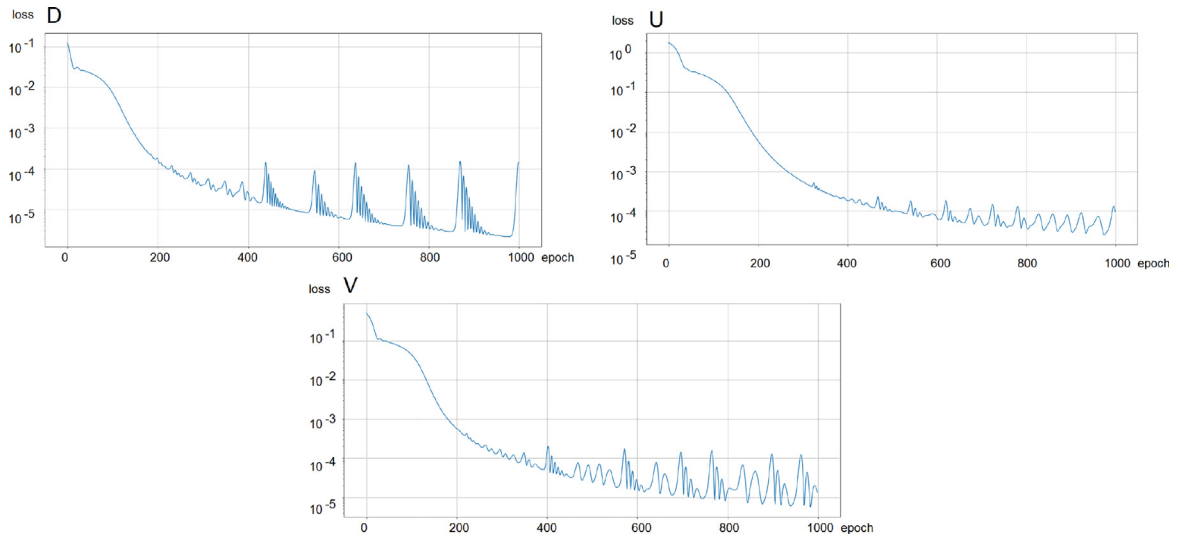

Fig. 13. Eriksson–Johnson problem. Training of the SVD matrices $\{\mathbb{U}, \mathbb{D}, \mathbb{V}\}$ related with the full matrix $\mathbb{W}_\epsilon$.

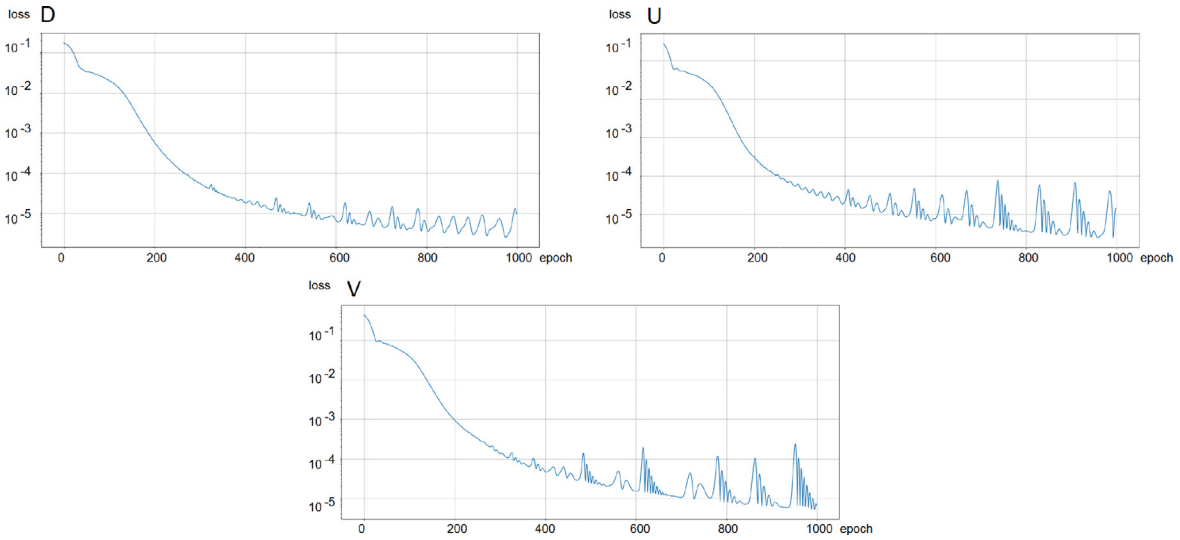

Fig. 14. Eriksson–Johnson problem. Training of the SVD matrices $\{\mathbb{U}, \mathbb{D}, \mathbb{V}\}$ related with the first block at the second hierarchical level.

In an online stage, for a given diffusion coefficient ϵ, we perform the compression of the matrix $\mathbb{W}_\epsilon$ into the hierarchical matrix $\mathbb{H}_\epsilon$ using Algorithm 6, where the admissibility condition is now provided by the trained neural network. The compressed hierarchical matrices are illustrated in Fig. 15.

Having the compressed matrix $\mathbb{H}_\epsilon$, we employ the GMRES algorithm [15] for solving the linear system (17). To avoid the computation with a dense $\mathbb{B}_\epsilon^T \mathbb{H}_\epsilon$ matrix, we note that the GMRES method involves computations of the residual $R = \mathbb{B}_\epsilon^T \mathbb{H}_\epsilon\, x - (L^T \mathbb{H}_\epsilon)^T$ and the hierarchical matrix $\mathbb{H}_\epsilon$ enables matrix–vector multiplications of $\mathbb{H}_\epsilon\, x$ and $L^T \mathbb{H}_\epsilon$ in a quasi-linear computational cost.

In Table 1 we summarize the computational costs of our solver for two different values of the ϵ parameter, namely $\epsilon = 10^{-1}$ and $\epsilon = 10^{-6}$. The computational trial mesh was a tensor product of quadratic B-splines with

Table 1
Computational costs of the stabilized Eriksson–Johnson solver using neural networks, hierarchical matrices and GMRES solver.

ϵ	Compress $\mathbb{W}_\epsilon$ flops with DNN	Compress $\mathbb{W}_\epsilon$ flops without DNN	$\mathbb{H}_\epsilon * x$ flops	$\mathbb{B}^T * \mathbb{H}_\epsilon x$ flops	# iter GMRES $\mathbb{H}$-matrix	Total flops
0.1	14,334	158,946	41,163	9,152	90	3,728,166
10^{-6}	31,880	117,922	33,100	9,002	76	3,232,392

Table 2
Computational costs of Galerkin method for the Eriksson–Johnson problem using GMRES solver.

ϵ	# iter GMRES Galerkin	Flops per iteration	Total flops
0.1	89	17,536	1,560,704
10^{-6}	65	17,536	1,139,840

Table 3
Execution times for (second column) stabilized Eriksson–Johnson problem using GMRES solver using hierarchical matrices and (third column) Eriksson–Johnson solver without matrix compression, as well as (fourth column) GMRES for Galerkin.

ϵ	# GMRES Petrov–Galerkin compressed $\mathbb{H}_\epsilon$ time	GMRES Petrov–Galerkin no compression time	GMRES Galerkin time
0.1	7.65 s	150 s	3.96 s
10^{-6}	1.77 s	109 s	0.77 s

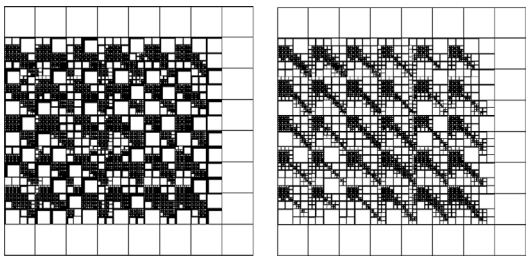

Fig. 15. Matrix of the coefficients of the optimal test functions compressed by using recursive SVD algorithm for $\epsilon = 10^{-1}$ and $\epsilon = 10^{-6}$. The compression algorithm removes singular values smaller than $\delta = 10^{-7}$, and employs 5 levels of hierarchy.

26 elements along x-axis and quadratic B-splines with 10 elements along y-axis. As we can read from the second and third columns, the DNN speeds up the compression process of the matrix of optimal test function's coefficients around ten times. We employ the GMRES solver that computes the residual. The cost of multiplication of the $\mathbb{H}_\epsilon * x$ and the cost of multiplication of $\mathbb{B}^T * \mathbb{H}_\epsilon x$ is included in the fourth and fifth column in Table 1. The total cost of the GMRES with hierarchical matrices augmented by DNN compression is equal to the compression cost of $\mathbb{H}_\epsilon$ with DNN, plus the number of iterations times the multiplication cost of $\mathbb{H}_\epsilon * x$ plus the multiplication cost of $\mathbb{B}^T * \mathbb{H}_\epsilon x$. The total cost is presented in the last column of Table 1.

For comparisons, we run the GMRES algorithm on the Galerkin method. The comparison is summarized in Table 2. The number of iterations, the cost per iteration, and the total cost are presented there. We can observe that the cost of the stabilized solution is two times larger than the cost of the Galerkin solution, in terms of floating-point operations and execution time. We are comparing here the costs of the solution obtained from the stable Petrov–Galerkin method, with the cost of the solution from the unstable Galerkin method.

We present in Table 3 the execution times of the GMRES solver using Petrov–Galerkin formulation with hierarchical matrix compression and without the compression. Additionally, we present the execution time of the

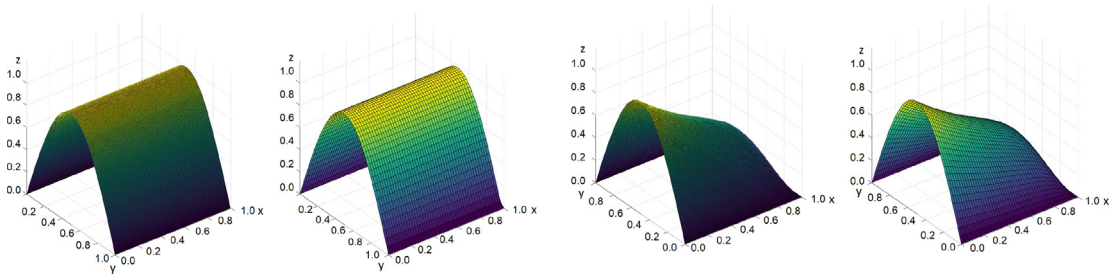

Fig. 16. Case inflow data $g = \sin(\pi y)$. Comparison of exact solutions (first and third column) and solutions obtained from GMRES solver using H-matrix (second and four column) for $\epsilon = 10^{-6}$ (first row) and $\epsilon = 10^{-1}$ (second row). Iterative solver executed with accuracy 10^{-10}.

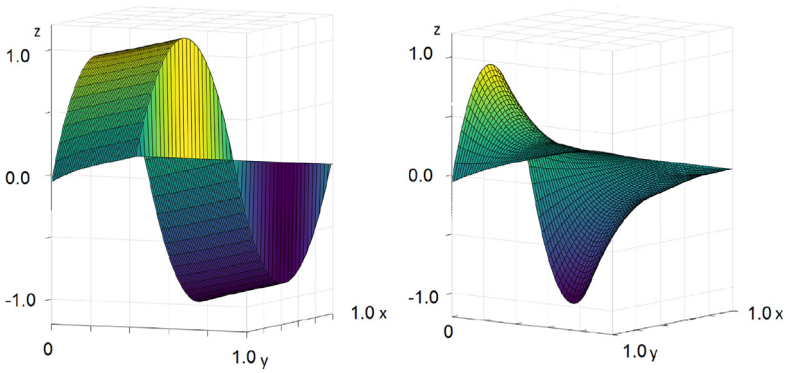

Fig. 17. Case inflow data $g = \sin(2\pi y)$. Solutions obtained from the Petrov–Galerkin formulation with $\epsilon = 10^{-6}$ and $\epsilon = 10^{-1}$.

GMRES for Galerkin method. For the comparison we use Octave implementation of GMRES algorithm with and without hierarchical matrix compression.

Numerical results comparing with the exact solution are depicted in Fig. 16 for inflow data $g = \sin(\pi y)$, and Fig. 17 for inflow data $g = \sin(2\pi y)$. Moreover, the Euclidean distance between discrete solutions of compressed and uncompressed methodologies is equal to $7.22 \cdot 10^{-10}$ for $\epsilon = 10^{-1}$, and it is equal to $1.45 \cdot 10^{-5}$ for $\epsilon = 10^{-6}$. Notice that these errors are induced by the hierarchical matrix compression error, as well as the approximation error of those matrix entries performed by the neural network. We emphasize that the error of the hierarchical matrix compression involves the difference between $\mathbb{H}_\epsilon$ and $\mathbb{W}_\epsilon$; while the error of the trained neural network concerns the difference between $\mathbb{H}_\epsilon$ and the neural network approximation of it. Detailed perturbation analysis on how these matrix errors translate to the discrete solutions will be the subject of another paper.

3.2. Helmholtz Problem

Given $\Omega = (0, 1)^2 \subset \mathbb{R}^2$ and $\kappa \in [1, 10]$, we seek the solution of the Helmholtz problem

$$\begin{cases} \Delta u + \kappa^2 u = f & \text{in } \Omega \\ u = g & \text{over } \partial\Omega\,, \end{cases} \tag{18}$$

with right-hand sides f and g such that the exact solution is $u(x, y) = \sin(\kappa\pi x)\sin(\kappa\pi y)$.

We employ 20×20 finite elements mesh. The trial space is constructed from quadratic B-splines. The test space is obtained with quadratic B-splines with C^0 separators (equivalent to Lagrange polynomials). The dependence of

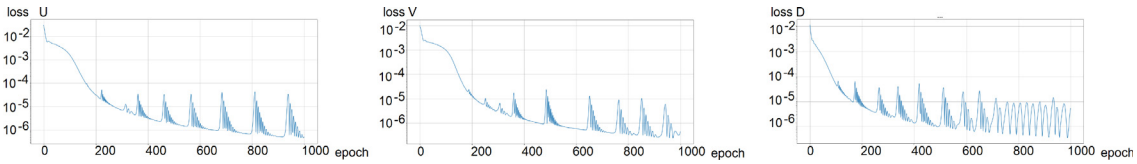

Fig. 18. Helmholtz problem. Training of the neural network for the SVD matrices for one block.

Table 4
Computational costs of the stabilized Helmholtz solver using neural networks, hierarchical matrices and GMRES solver.

κ	Compress $\mathbb{W}_\kappa$ flops with DNN	Compress $\mathbb{W}_\kappa$ flops without DNN	$\mathbb{H}_\kappa * x$ flops	$\mathbb{A} * \mathbb{H}_\kappa x$ flops	# iter GMRES $\mathbb{H}$-matrix	Total flops
1	0	129,788	47,649	8,901	10	565,500
10	0	129,788	47,649	8,877	31	1,752,306

the coefficients of the optimal test functions on κ for the Helmholtz problem has the affine structure described in Section 2.2, thus we can *offline* construct the function

$$\mathcal{P} \ni \kappa \rightarrow \mathbb{W}_\kappa, \tag{19}$$

where $\mathbb{W}_\kappa$ is the matrix of the coefficients of the optimal test functions. We fix the trial and test spaces used for approximation of the solution and stabilization of the Petrov–Galerkin formulation. Next, we consider blocks of different size of matrix $\mathbb{W}_\kappa$, and we train the SVD for these different blocks as a function of κ, i.e.,

$$\mathcal{P} \ni \kappa \rightarrow \text{DNN}(\kappa) = \{\mathbb{U}_i(\kappa), \mathbb{D}_i(\kappa), \mathbb{V}_i(\kappa)\}_{i=1,\ldots,N_B} . \tag{20}$$

The convergence of the training procedure is presented in Fig. 18. Knowing $\mathbb{D}_i(\kappa)$ a priori for a given κ allows us to construct the structure of the hierarchical matrix, and we obtain the $\mathbb{U}_i(\kappa)$ and $\mathbb{V}_i(\kappa)$ from the neural networks. Fig. 20 depicts the exemplary resulting hierarchical matrices. The hierarchical matrix compression of $\mathbb{W}_\kappa$ (for a given κ) is obtained from the neural network. It has been computed offline, and the online cost is just the evaluation of the neural network.

Table 4 summarizes the computational costs of our solver for two values of $\kappa = \{1, 10\}$. The computational mesh was a tensor product of quadratic B-splines with 10 elements along each of the axes.

We employ the GMRES solver that computes the residual. The cost of multiplication of the $\mathbb{H}_\kappa * x$ and the cost of multiplication of $\mathbb{B}^T * \mathbb{H}_\kappa x$ is included in the fourth and fifth column in Table 4. The total cost of the GMRES with hierarchical matrices augmented by DNN compression is equal to the **number of iterations** times the **multiplication cost of** $\mathbb{H}_\kappa * x$ **plus multiplication cost of** $\mathbb{B}^T * \mathbb{H}_\kappa x$. The total cost is presented in the last column of Table 4.

We present in Table 5 the cost of the GMRES algorithm executed on the Galerkin method. We present the number of iterations, the cost per iteration, and the total cost. We can see that the number of floating-point operations to obtain the stabilized solution is 4 times larger than the number of floating-point operations to obtain the Galerkin solution. Namely, for $\kappa = 1$, we have 144,440 flops of Galerkin method versus 565,500 flops of Petrov–Galerkin method. For $\kappa = 10$, we have 447,764 flops of Galerkin method versus 1,752,306 flops of Petrov–Galerkin method. The comparison of the solution obtained with the Petrov–Galerkin formulation with the optimal test functions generated by DNN and the exact solution is presented in Fig. 19.

We also show in Table 6 the execution times of the GMRES solver for the Helmholtz problem stabilized with Petrov–Galerkin formulation using hierarchical matrix compression, without the hierarchical matrix compression, and the execution time of the GMRES with the Galerkin formulation (resulting in incorrect solution). We employ the Octave implementation of GMRES algorithm with and without hierarchical matrix compression. The Euclidean distance between discrete solutions of compressed and uncompressed methodologies is equal to $4 \cdot 10^{-14}$ for $\kappa = 1$, and it is equal to $3.84 \cdot 10^{-13}$ for $\kappa = 8$.

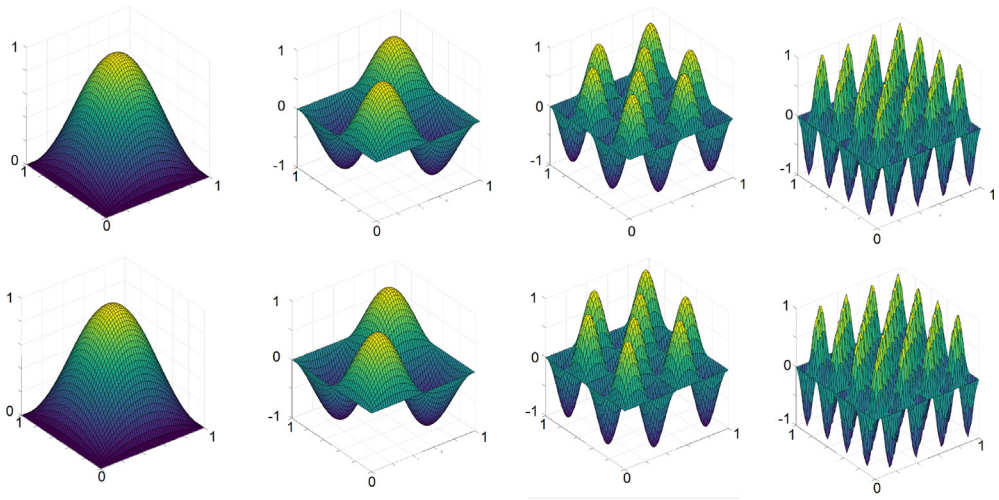

Fig. 19. The solutions obtained for the Helmholtz problem for $\kappa = 1, 2, 4, 8$. The solution obtained from the Petrov–Galerkin formulation with the optimal test functions provided by the neural network (first row). The exact solution (second row).

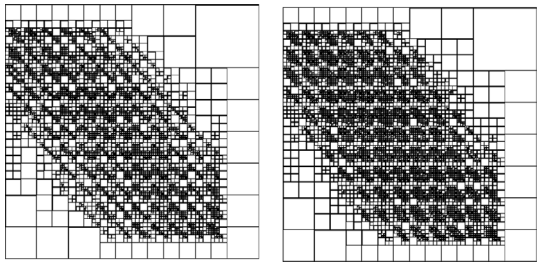

Fig. 20. The hierarchical matrices for $\kappa = 1$ (left panel) and $\kappa = 10$ (right panel). The compression algorithm removes the singular values smaller than $\delta = 10^{-7}$ and employs 5 levels of hierarchy.

Table 5
Computational costs of Galerkin method for the Helmholtz problem using GMRES solver.

ϵ	# iter GMRES Galerkin	Flops per iteration	Total flops
1	10	14,444	144,440
10	31	14,444	447,764

3.3. Generalization of the Eriksson–Johnson problem into 3D

We have generalized the Eriksson–Johnson problem into three-dimensions. Given $\Omega = (0,1)^3 \subset \mathbb{R}^3$ and $\beta = (1,0,0)$, we seek the solution of the advection–diffusion problem

$$\begin{cases} -\epsilon\, \Delta u + \beta \cdot \nabla u = 0 & \text{in } \Omega \\ u = \sin(k\pi y)\sin(k\pi z)\chi_{\{x=0\}} & \text{over } \partial\Omega\,, \end{cases} \tag{21}$$

Table 6
Execution times for (second column) stabilized Helmholtz problem using GMRES solver with hierarchical matrices and (third) GMRES solver without matrix compression, as well as (fourth column) GMRES for Galerkin method.

κ	# GMRES Petrov–Galerkin compressed $\mathbb{H}_\epsilon$ time	GMRES Petrov–Galerkin no compression time	GMRES Galerkin time
1	6.5 s	22 s	1.62 s
10	9.2 s	48 s	2.34 s

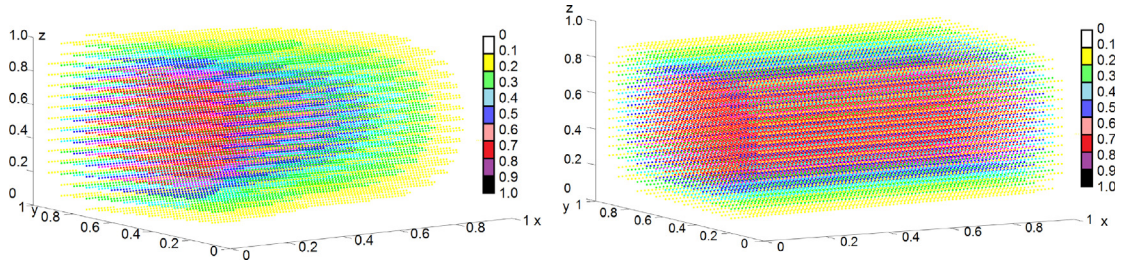

Fig. 21. Solutions to the generalized 3D Eriksson–Johnson problem (21), for $\epsilon = 0.1$ (left panel) and $\epsilon = 10^{-3}$ (right panel)..

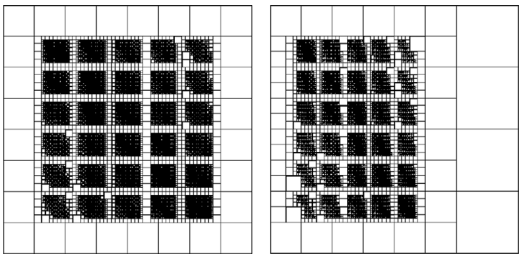

Fig. 22. Eriksson–Johnson problem generalized into 3D. Hierarchical matrix of coefficients of the optimal test functions, for $\epsilon = 10^{-1}$ and $\epsilon = 10^{-3}$. Compression with $\delta = 10^{-6}$ and 5 levels of hierarchy.

where $\chi_{\{x=0\}}$ denotes the characteristic function over the inflow boundary $x = 0$. The associated weak form will be the same as in (15).

We illustrate the solution in Fig. 21. We employ 26 intervals along x axis, and 10 intervals along y and z axis. We use quadratic B-splines for trial and cubic B-splines for test.

Fig. 22 illustrates the hierarchical matrix of coefficients of the optimal test functions compressed with $\delta = 10^{-6}$ and 5 levels of hierarchy. We have performed the training for the following set of ϵ:

$$\{1, 0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1,$$
$$0.09, 0.08, 0.07, 0.06, 0.05, 0.04, 0.03, 0.02, 0.01,$$
$$0.009, 0.008, 0.007, 0.006, 0.005, 0.004, 0.003, 0.002, 0.001\}.$$

The odd values of ϵ are used for the training, while the even values are employed for validation. The examples of convergence of the training as expressed by the MSE value for some SVD blocks are illustrated in Figs. 23–24.

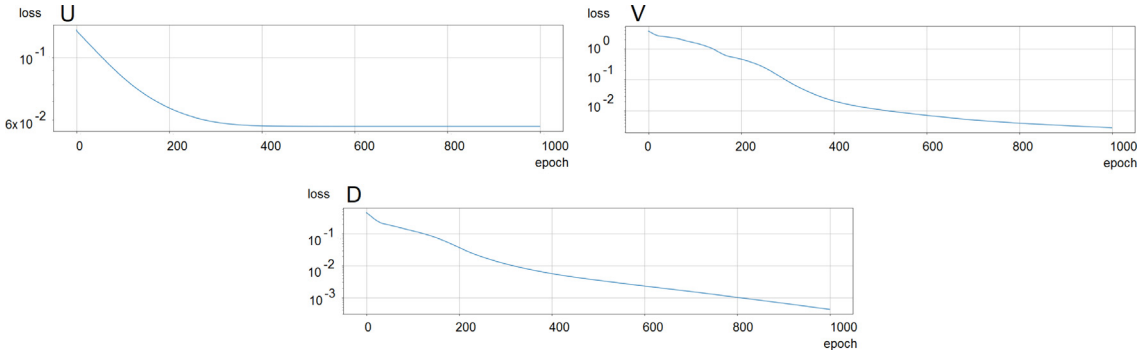

Fig. 23. Eriksson–Johnson problem generalized into 3D. Training of the neural network the SVD matrices for the first one of the 2×2 blocks. The MSE for entries of $\{\mathbb{U}, \mathbb{D}, \mathbb{V}\}$ computed for the validation set.

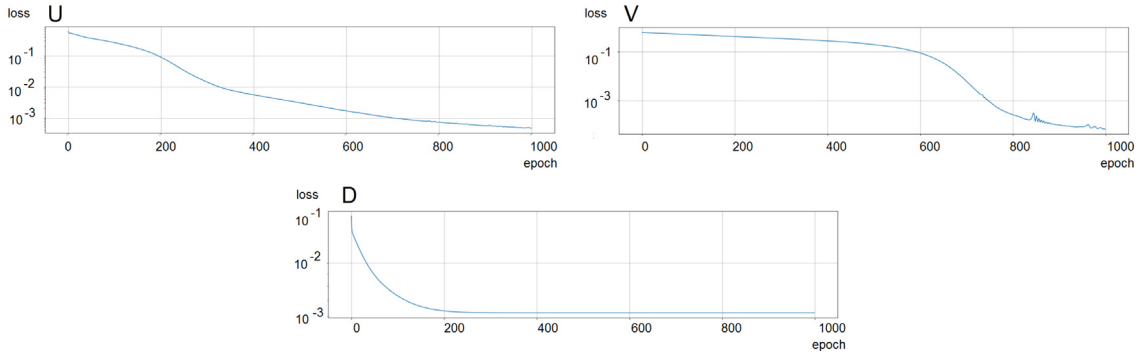

Fig. 24. Eriksson–Johnson problem generalized into 3D. Training of the neural network the SVD matrices for the first one of the 4×4 blocks. The MSE for entries of $\{\mathbb{U}, \mathbb{D}, \mathbb{V}\}$ computed for the validation set.

Table 7
Computational costs of the stabilized 3D extension of Eriksson–Johnson solver using neural networks, hierarchical matrices and GMRES solver.

ϵ	GMRES Petrov-Galerkin flops	GMRES Galerkin flops	GMRES Petrov-Galerkin # iter	GMRES Galerkin # iter	GMRES Petrov-Galerkin time	GMRES Galerkin time
0.1	2,337,300	385,815	100	85	160 s	11 s
0.001	2,789,836	462,978	119	102	164 s	12 s

In Table 7 we present general comparison of the number of floating-point operations, number of iterations and total execution times for GMRES solver for Galerkin formulation (resulting in incorrect solution), GMRES solver stabilized with Petrov–Galerkin formulation with hierarchical matrix compression, and without hierarchical matrix compression. The Euclidean distance between discrete solutions of compressed and uncompressed methodologies is equal to $8.88 \cdot 10^{-4}$ for $\epsilon = 0.1$, and it is equal to $1.31 \cdot 10^{-4}$ for $\epsilon = 0.001$.

4. Conclusions

We have employed the Petrov–Galerkin formulation with optimal test functions for the stabilization of the discretization of challenging PDEs. We have focused on advection-dominated diffusion and Helmholtz problems. During the *offline* phase, we explicitly compute the matrix of coefficients of optimal test functions for any PDE parameter. We have also trained neural networks to compute for each PDE parameter the bottleneck of hierarchical matrix compression. During the *online* phase, we rapidly compute the matrix compression using the neural networks, and we perform the GMRES iterative solver on the reduced Petrov–Galerkin linear system, where vector–matrix multiplications are done in a quasi-linear computational cost, due to the hierarchical structure of the low-rank decomposition used.

As future work, we plan to extend this idea to time-dependent problems, starting with a non-stationary advection-dominated diffusion problem [5]. We also plan to implement the method for time-dependent Navier–Stokes equations [4]. Additionally, we want to experiment with more complex parametric PDEs, where the parameters are piecewise constant through the domain, by means of a domain decomposition technique.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

Maciej Paszyński was supported by the program "Excellence initiative – research university" for the AGH University of Science and Technology. Ignacio Muga was supported by the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No. 777778 (MATHROCKS); and the Chilean National Agency for Research & Development through the Fondecyt Project #1230091.

Appendix A. Algorithms

A.1. Recursive hierarchical compression of the matrix

Require: $t_{\min}, t_{\max}, s_{\min}, s_{\max} \in \mathbb{N}$ (row and column index ranges),
$1 \leq t_{\min} \leq t_{\max} \leq n$, $1 \leq s_{\min} \leq s_{\max} \leq m$ where $n \times m$ is the size of the matrix to be compressed
if Admissible($t_{\min}, t_{\max}, s_{\min}, s_{\max}, r, \delta$) **then**
v = CompressMatrix($t_{\min}, t_{\max}, s_{\min}, s_{\max}, r$)
else
// Create a new node with four sons corresponding to four //quarters of the matrix
create new node v
AppendChild(v,CreateTree($t_{\min}, t_{\text{newmax}}, s_{\min}, s_{\text{newmax}}$))
AppendChild(v,CreateTree($t_{\min}, t_{\text{newmax}}, s_{\text{newmax}}+1, s_{\max}$))
AppendChild(v,CreateTree($t_{\text{newmax}}+1, t_{\max}, s_{\min}, s_{\text{newmax}}$))
AppendChild(v,CreateTree($t_{\text{newmax}}+1, t_{\max}, s_{\text{newmax}}+1, s_{\max}$))
end if
RETURN v

Algorithm 1: Recursive hierarchical compression of the matrix: **CreateTree**(r, δ) where r is the rank used for the compression, and δ is the threshold for the singular values.

A.2. Checking of the admissibility condition

Require: $t_{\min}, t_{\max}, s_{\min}, s_{\max}$ - range of indexes of block, δ compression threshold, r maximum rank
  if block $(t_{\min}, t_{\max}, s_{\min}, s_{\max})$ consist of zeros **then**
    return true;
  end if
  $[\mathbb{U}, \mathbb{D}, \mathbb{V}] \leftarrow$ reducedSVD$(t_{\min}, t_{\max}, s_{\min}, s_{\max}, r+1)$; $\sigma \leftarrow$ diag$(\mathbb{D})$;
  if $\sigma(r+1) < \delta$ **then**
    return true;
  end if
  return false;

Algorithm 2: Checking of the admissibility condition: result = Admissible$(t_{\min}, t_{\max}, s_{\min}, s_{\max}, r, \delta)$

A.3. Matrix–vector multiplication

Require: node v representing *compressed matrix* $\mathbb{H}(v) \in \mathcal{M}^{m\times n}$, $X \in \mathcal{M}^{n\times c}$ *vectors to multiply*
  if $v.nr\_sons == 0$ **then**
    if $v.rank == 0$ **then**
      return $zeros(size(A).rows)$
    end if
    return $v.U * (v.V * X)$
  end if
  $rows = size(X).rows$
  $X_1 = X(1 : \frac{rows}{2}, *)$
  $X_2 = X(\frac{rows}{2} + 1 : size(A).rows, *)$
  $C_2 = v.son(1).U$; $C_1 = v.son(1).V$
  $D_2 = v.son(2).U$; $D_1 = v.son(2).V$
  $E_2 = v.son(3).U$; $E_1 = v.son(3).V$
  $F_2 = v.son(4).U$; $F_1 = v.son(4).V$
  return $\begin{bmatrix} C_2 * (C_1 * X_1) + D_2 * (D_1 * X_2) \\ E_2 * (E_1 * X_1) + F_2 * (F_1 * X_2) \end{bmatrix}$

Algorithm 3: Matrix–vector multiplication: $Y =$ **matrix_vector_mult**(v, X)

A.4. rSVD compression of a block

Require: $t_{\min}, t_{\max}, s_{\min}, s_{\max}$ - range of indexes of block, δ *compression threshold*, r *maximum rank*
  if block $(t_{\min}, t_{\max}, s_{\min}, s_{\max})$ consist of zeros **then**
    create new node v; $v.rank \leftarrow 0$; $v.size \leftarrow size(t_{\min}, t_{\max}, s_{\min}, s_{\max})$; **return** v;
  end if
  $[\mathbb{U}, \mathbb{D}, \mathbb{V}] \leftarrow$ reducedSVD$(t_{\min}, t_{\max}, s_{\min}, s_{\max}, r)$; $\sigma \leftarrow$ diag$(\mathbb{D})$;
  $rank \leftarrow rank(\mathbb{D})$
  create new node v; $v.rank \leftarrow rank$;
  $v.singularvalues \leftarrow \sigma(1 : rank)$;
  $v.U \leftarrow U(*, 1 : rank)$;
  $v.V \leftarrow D(1 : rank, 1 : rank) * V(1 : rank, *)$;
  $v.sons \leftarrow \emptyset$; $v.size \leftarrow size(t_{\min}, t_{\max}, s_{\min}, s_{\max})$;
  return v;

Algorithm 4: rSVD compression of a block: $node =$ **CompressMatrix**$(t_{\min}, t_{\max}, s_{\min}, s_{\max}, r)$

A.5. Pseudo-code of the GMRES algorithm

Require: A matrix, b right-hand-side vector, x_0 starting point
Compute $r_0 = b - Ax_0$
Compute $v_1 = \frac{r_0}{\|r_0\|}$
for $j = 1, 2, ..., k$
Compute $h_{i,j} = \left(Av_j, v_i\right)$ for $i = 1, 2, ..., j$
Compute $\hat{v}_{j+1} = Av_j - \sum_{i=1,...,j} h_{i,j} v_i$
Compute $h_{j+1,j} = \|\hat{v}_{j+1}\|_2$
Compute $v_{j+1} = \hat{v}_{j+1} / h_{j+1,j}$
end for
Form solution $x_k = x_0 + V_k y_k$ where $V_k = [v_1 ... v_k]$, and y_k minimizes $J(y) = \|\beta e_1 - \hat{H}_k y\|$ where

$$\hat{H} = \begin{bmatrix} h_{1,1} & h_{1,2} \cdots h_{1,k} & & \\ h_{2,1} & h_{2,2} \cdots h_{2,k} & & \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & h_{k,k-1} & h_{k,k} \\ 0 & \cdots & 0 & h_{k+1,k} \end{bmatrix}$$

Algorithm 5: Pseudo-code of the GMRES algorithm

A.6. Recursive hierarchical compression of the matrix augmented by neural network

Require: $t_{\min}, t_{\max}, s_{\min}, s_{\max}, \in \mathbb{N}$ (row and column index ranges), r rank of the blocks, δ accuracy of compression, μ PDE parameter
$1 \leq t_{\min} \leq t_{\max} \leq n, 1 \leq s_{\min} \leq s_{\max} \leq m$ where $n \times m$ is the size of the matrix to be compressed
i = block index for $(t_{\min}, t_{\max}, s_{\min}, s_{\max})$
if $\mathbb{D}_i(\mu)[r+1] < \delta$ *(asking NN for block singularvalues)* **then**
if block $(t_{\min}, t_{\max}, s_{\min}, s_{\max})$ consist of zeros **then**
create new node v; $v.rank \leftarrow 0$; $v.size \leftarrow size(t_{\min}, t_{\max}, s_{\min}, s_{\max}, s, t)$; **return** v;
end if
$[\mathbb{U}_i(\mu), \mathbb{D}_i(\mu), \mathbb{V}_i(\mu)]$*(asking NN for SVD);*$\sigma \leftarrow \text{diag}(\mathbb{D}_i(\mu))$;
$rank \leftarrow rank(\mathbb{D}_i(\mu))$
create new node v; $v.rank \leftarrow rank$;
$v.singularvalues \leftarrow \sigma(1 : rank)$;
$v.U \leftarrow \mathbb{U}_i(\mu)(*, 1 : rank)$;
$v.V \leftarrow \mathbb{D}_i(\mu)(1 : rank, 1 : rank) * \mathbb{V}_i(\mu)(1 : rank, *)$;
$v.sons \leftarrow \emptyset$; $v.size \leftarrow size(t_{\min}, t_{\max}, s_{\min}, s_{\max})$;
return v;
else
// Create a new node with four sons corresponding to four //quarters of the matrix
create new node v
AppendChild(v,CreateTreeNN($t_{\min}, t_{\text{newmax}}, s_{\min}, s_{\text{newmax}}, r, \delta, \mu$))
AppendChild(v,CreateTreeNN($t_{\min}, t_{\text{newmax}}, s_{\text{newmax}} + 1, s_{\max}, r, \delta, \mu$))
AppendChild(v,CreateTreeNN($t_{\text{newmax}} + 1, t_{\max}, s_{\min}, s_{\text{newmax}}, r, \delta, \mu$))
AppendChild(v,CreateTreeNN($t_{\text{newmax}} + 1, t_{\max}, s_{\text{newmax}} + 1, s_{\max}, r, \delta, \mu$))
end if
RETURN v

Algorithm 6: Recursive hierarchical compression of the matrix augmented by neural network: **CreateTreeNN(1,rowsof(**A**),1,columnsof(**A**),**r, δ, μ) where r is the rank used for the compression, and δ is the threshold for the r singular values, μ is the PDE parameter.

References

[1] K.W. Morton, Numerical Solution of Convection-Diffusion Problems, CRC Press, 2019.
[2] O.G. Ernst, M.J. Gander, Why it is difficult to solve Helmholtz problems with classical iterative methods, Numer. Anal. Multiscale Probl. (2012) 325–363.
[3] L. Demkowicz, J. Gopalakrishnan, A class of discontinuous Petrov–Galerkin methods. II. Optimal test functions, Numer. Methods Part. Differ. Equ. 27 (1) (2011) 70–105.
[4] V.M. Calo, M. Łoś, Q. Deng, I. Muga, M. Paszyński, Isogeometric residual minimization method (iGRM) with direction splitting preconditioner for stationary advection-dominated diffusion problems, Comput. Methods Appl. Mech. Engrg. 373 (2021) 113214.
[5] M. Łoś, J. Munoz-Matute, I. Muga, M. Paszyński, Isogeometric residual minimization method (iGRM) with direction splitting for non-stationary advection–diffusion problems, Comput. Math. Appl. 79 (2) (2020) 213–229.
[6] M. Łoś, I. Muga, J. Munoz-Matute, M. Paszyński, Isogeometric residual minimization (iGRM) for non-stationary Stokes and Navier–Stokes problems, Comput. Math. Appl. 95 (2021) 200–214.
[7] R. Stevenson, J. Westerdiep, Minimal residual space–time discretizations of parabolic equations: Asymmetric spatial operators, Comput. Math. Appl. 101 (2021) 107–118.
[8] V.M. Calo, Residual-Based Multiscale Turbulence Modeling: Finite Volume Simulations of Bypass Transition, Stanford University, 2005.
[9] T.J.R. Hughes, L.P. Franca, M. Mallet, A new finite element formulation for computational fluid dynamics: VI. Convergence analysis of the generalized supg formulation for linear time-dependent multidimensional advectivediffusive systems, Comput. Methods Appl. Mech. Eng. 63 (1) (1987) 97–112.
[10] L. Demkowicz, J. Gopalakrishnan, An Overview of the Discontinuous Petrov Galerkin Method, Springer International Publishing, 2014.
[11] L. Demkowicz, J. Gopalakrishnan, I. Muga, J. Zitelli, Wavenumber explicit analysis of a DPG method for the multidimensional Helmholtz equation, Comput. Methods Appl. Mech. Eng. 213-216 (2012) 126–138.
[12] L. Demkowicz, N. Heuer, Robust DPG method for convection-dominated diffusion problems, SIAM J. Numer. Anal. 51 (5) (2013) 2514–2537.
[13] W. Hackbusch, Hierarchical Matrices: Algorithms and Analysis, Springer, 2015.
[14] W. Hackbusch, A sparse matrix arithmetic based on H-matrices. Part I: Introduction to h-matrices, Computing (1999) 89–108.
[15] Y. Saad, Iterative Methods for Sparse Linear Systems, SIAM, 2003.
[16] S. Le Borne, Hierarchical preconditioners for high-order FEM, in: T. Dickopf, J.M. Gander, L. Halpern, R. Krause, L.F. Pavarino (Eds.), Domain Decomposition Methods in Science and Engineering XXII, Springer International Publishing, Cham, 2016, pp. 559–566.
[17] J. Zechner, B. Marussig, G. Beer, F. Thomas-Peter, Isogeometric boundary element method with hierarchical matrices, 2014, pp. 1–10, arXiv:1406.2817.
[18] G.C. Diwan, M.S. Mohamed, Iterative solution of Helmholtz problem with high-order isogeometric analysis and finite element method at midrange frequencies, Comput. Methods Appl. Mech. Engrg. 363 (2020) 112855.
[19] V. Dwarka, R. Tielen, M. Möller, C. Vuik, Towards accuracy and scalability: Combining isogeometric analysis with deflation to obtain scalable convergence for the Helmholtz equation, Comput. Methods Appl. Mech. Engrg. 377 (2021) 113694.
[20] D. Garcia, D. Pardo, L. Dalcin, V.M. Calo, Refined isogeometric analysis for a preconditioned conjugate gradient solver, Comput. Methods Appl. Mech. Engrg. 335 (2018) 490–509.
[21] P.R. Amestoy, I.S. Duff, J.-Y. L'Excellent, Multifrontal parallel distributed symmetric and unsymmetric solvers, Comput. Methods Appl. Mech. Engrg. 184 (2–4) (2000) 501–520.
[22] P.R. Amestoy, I.S. Duff, J.-Y. L'Excellent, J. Koster, A fully asynchronous multifrontal solver using distributed dynamic scheduling, SIAM J. Matrix Anal. Appl. 23 (1) (2001) 15–41.
[23] P.R. Amestoy, A. Guermouche, J.-Y. L'Excellent, S. Pralet, Hybrid scheduling for the parallel solution of linear systems, Parallel Comput. 32 (2) (2006) 136–156.
[24] D. Garcia, D. Pardo, L. Dalcin, M. Paszyński, N. Collier, V.M. Calo, The value of continuity: Refined isogeometric analysis and fast direct solvers, Comput. Methods Appl. Mech. Engrg. 316 (2017) 586–605, Special Issue on Isogeometric Analysis: Progress and Challenges.
[25] G. Chavez, G. Turkiyyah, D.E. Keyes, A direct elliptic solver based on hierarchically low-rank Schur complements, in: C.-O. Lee, X.-C. Cai, D.E. Keyes, H.H. Kim, A. Klawonn, E.-J. Park, O.B. Widlund (Eds.), Domain Decomposition Methods in Science and Engineering XXIII, Springer International Publishing, Cham, 2017, pp. 135–143.
[26] S. Bauer, M. Mohr, U. Rüde, J. Weismüller, M. Wittmann, B. Wohlmuth, A two-scale approach for efficient on-the-fly operator assembly in massively parallel high performance multigrid codes, Appl. Numer. Math. 122 (2017) 14–38.
[27] D. Drzisga, B. Keith, B. Wohlmuth, The surrogate matrix methodology: A priori error estimation, SIAM J. Sci. Comput. 41 (6) (2019) A3806–A3838.
[28] D. Drzisga, B. Keith, B. Wohlmuth, The surrogate matrix methodology: Accelerating isogeometric analysis of waves, Comput. Methods Appl. Mech. Engrg. 372 (2020) 113322.
[29] D. Drzisga, B. Keith, B. Wohlmuth, The surrogate matrix methodology: Low-cost assembly for isogeometric analysis, Comput. Methods Appl. Mech. Engrg. 361 (2020) 112776.
[30] A. Ern, J.-L. Guermond, Theory and Practice of Finite Elements, Vol. 159, Springer, 2013.
[31] J. Chan, J.A. Evans, W. Qiu, A dual Petrov–Galerkin finite element method for the convection–diffusion equation, Comput. Math. Appl. 68 (11) (2014) 1513–1529.
[32] I. Muga, K.G. van Der Zee, Discretization of linear problems in Banach spaces: Residual minimization, nonlinear Petrov–Galerkin, and monotone mixed methods, SIAM J. Numer. Anal. 58 (6) (2020) 3406–3426.