Operator compression with deep neural networks

Fabian Kröpfl, Roland Maier, Daniel Peterseim

Angaben zur Veröffentlichung / Publication details:

Kröpfl, Fabian, Roland Maier, and Daniel Peterseim. 2022. "Operator compression with deep neural networks." *Advances in Continuous and Discrete Models* 2022 (1): 29. https://doi.org/10.1186/s13662-022-03702-y.

Advances in Continuous and Discrete Models

RESEARCH Open Access

Operator compression with deep neural networks

Fabian Kröpfl[1*], Roland Maier[2] and Daniel Peterseim[1,3]

*Correspondence:
fabian.kroepfl@uni-a.de
[1]Institute of Mathematics, University of Augsburg, Universitätsstr. 12a, 86159 Augsburg, Germany
Full list of author information is available at the end of the article

Abstract

This paper studies the compression of partial differential operators using neural networks. We consider a family of operators, parameterized by a potentially high-dimensional space of coefficients that may vary on a large range of scales. Based on the existing methods that compress such a multiscale operator to a finite-dimensional sparse surrogate model on a given target scale, we propose to directly approximate the coefficient-to-surrogate map with a neural network. We emulate local assembly structures of the surrogates and thus only require a moderately sized network that can be trained efficiently in an offline phase. This enables large compression ratios and the online computation of a surrogate based on simple forward passes through the network is substantially accelerated compared to classical numerical upscaling approaches. We apply the abstract framework to a family of prototypical second-order elliptic heterogeneous diffusion operators as a demonstrating example.

MSC: 68T07; 65N30; 35J15

Keywords: Deep learning; Neural networks; Numerical homogenization; Model order reduction

1 Introduction

The remarkable success of machine learning technology, especially deep learning, in classical AI disciplines such as image recognition and natural language processing has led to an increased research interest in leveraging the power of these approaches in other science and engineering disciplines over the last years. In the field of numerical modeling and simulation, promising approaches are emerging that try to integrate machine learning algorithms and traditional physics-based approaches, combining the advantages of the data-driven regime with known physics and domain knowledge. In this spirit, many different approaches to approximating solutions of partial differential equations (PDEs) with neural networks have been proposed, for example so-called physics-informed neural networks (PINNs) [60], the deep Galerkin method [63], or the deep Ritz method [19]. It has become evident that the strategy of using neural networks as ansatz functions for the approximation of a PDE's solution is especially advantageous for high-dimensional problems that are outside the reach of classical mesh-based methods [17, 18, 33]. For some classes

of PDEs, e.g., Kolmogorov PDEs and semilinear heat equations, it has even been proven that neural networks break the curse of dimensionality [9, 39].

In this spirit, we strongly believe that the strength of neural networks lies in scenarios where one deals with a whole family of PDEs rather than one single equation, for example in the context of so-called parametric PDEs, i.e., settings where a family of partial differential operators parameterized by some coefficient is considered, see, e.g., [62]. This is particularly true for multiscale problems, where one is interested in computing coarse-scale surrogates for problems involving a range of scales that cannot be resolved in a direct numerical simulation.

In this paper, we study the problem of approximating a coefficient-to-surrogate map with a neural network in a very general setting of parameterized PDEs with arbitrarily rough coefficients that may vary on a microscopic scale. In other words, we are not trying to directly approximate the parameter-to-solution map, but rather compress the fine-scale information contained in the continuous operator to a finite-dimensional sparse object that is able to replicate the effective behavior of the solution on a macroscopic scale of interest even in the presence of unresolved oscillations of the underlying coefficient.

The output surrogate models are based on the idea of modern numerical homogenization techniques such as localized orthogonal decomposition [46, 49, 56], gamblets [52], rough polyharmonic splines [53], the multiscale finite element method [21, 38], or the generalized finite element method [7, 20]; see [5] and the references therein for a comprehensive overview. These methods have demonstrated high performance in many relevant applications such as porous media flow or wave scattering in heterogeneous media to mention only a few. In particular, they typically do not require explicit assumptions on the existence of lower-dimensional structures in the underlying family of PDE coefficients and yield sparse system matrices that ensure uniform approximation properties of the resulting surrogate. Moreover, the computation of the system matrices mimics the standard assembly procedure from finite element theory, consisting of the generation of local system matrices and their combination by local-to-global mappings, which is exploited to reduce the size of the network architecture and its complexity considerably.

The possibility of fast computation of the surrogates has high potential for multi-query problems, such as in uncertainty quantification, and time-dependent or inverse multiscale problems, which require the computation of surrogates for many different a priori unknown coefficients. Though the aforementioned numerical homogenization methods lead to accurate surrogates for the whole class of coefficients, their computation requires the resolution of all scales locally which marks a severe limitation when it has to be performed many times for the solution of a multi-query problem. There have been attempts to tackle this problem, but the results so far are only applicable to small perturbation regimes [35, 50] or settings where the parameterization fulfills additional smoothness requirements [3].

To overcome this problem, we propose to learn the whole nonlinear coefficient-to-surrogate map from a training set consisting of pairs of coefficients and their corresponding surrogates with a deep neural network. In other words, we are combining the domain knowledge from numerical homogenization with a data-driven deep learning approach by essentially learning a numerical homogenization method from data. To this end, we propose using an offline-online approach. In the offline phase, the neural network is trained based on data generated with existing numerical homogenization techniques. In the on-

line phase, the compression of previously unseen operators can then be reduced to a simple forward pass of the neural network, which eliminates the computational bottleneck encountered in multi-query settings.

Our method is conceptually different from the existing approaches that try to integrate ideas from homogenization with neural networks. In [6] for example, the authors propose to learn a homogenized PDE from simulation data by linking deep learning with an equation-free multiscale approach. Other papers in the context of uncertainty quantification suggest training a neural network to identify suitable multiscale basis functions for the finite volume method given a porous random medium [13, 54]. In [31], the authors consider the problem of elasticity with history-dependent material properties, where a recurrent deep neural network connects microscopic and macroscopic material parameters. In deep multiscale model learning [65], learning techniques are used to predict the evolution from one time step to another within a given coarse multiscale space. The goal of this approach is to obtain a reasonable coarse operator for the successive approximation of a time-dependent PDE. Furthermore, several experimental and theoretical works on the approximation of the coefficient-to-solution map [11, 28, 30, 43] or other quantities of interest such as the ground state energy in Schrödinger equations [41] by deep neural networks have been published in the context of parametric PDEs.

This paper is structured as follows: in Sect. 2, we introduce and motivate the abstract framework for a very general class of linear differential operators. After that, we study the problem of elliptic homogenization as an example of how to apply the general methodology in practice. In Sect. 4, we conduct numerical experiments that show the feasibility of our ideas developed in the previous two sections. We conclude this work with an outlook on further research questions.

2 Abstract framework

In this section, we describe the general abstract problem of finding discrete compressed surrogates to a family of differential operators that allow us to satisfactorily approximate the original operators on a target scale of interest, given only the underlying coefficients but *not* a high resolution representation of the operators. We elaborate on how to speed up the online computation of those compressed representatives using deep neural networks after an initial offline training phase.

2.1 Setting

Let $D \subseteq \mathbb{R}^d$, $d \in \{1, 2, 3\}$ be a bounded Lipschitz domain and $H_0^1(D)$ be the Sobolev space of L^2-functions with weak first derivatives in $L^2(D)$ that vanish on the boundary of D. We write $H^{-1}(D)$ for the dual space of $H_0^1(D)$ and $\langle \cdot, \cdot \rangle$ for the duality pairing between $H^{-1}(D)$ and $H_0^1(D)$. Consider a family of linear differential operators

$$\mathfrak{L} := \left\{ \mathfrak{L}_A : H_0^1(D) \to H^{-1}(D) \mid A \in \mathfrak{A} \right\}$$

that is parameterized by some class $\mathfrak{A} \subseteq L^\infty(D)$ of admissible coefficients. We emphasize that we do not pose any assumptions on the structure of the coefficients $A \in \mathfrak{A}$ such as periodicity or scale separation and explicitly allow for arbitrarily rough coefficients that may vary on a continuum of scales up to some microscale $\varepsilon \ll \operatorname{diam}(D)$. We assume that for every $A \in \mathfrak{A}$ the associated operator $\mathfrak{L}_A$ is symmetric ($\langle \mathfrak{L}_A u, v \rangle = \langle u, \mathfrak{L}_A v \rangle$), local ($\langle \mathfrak{L}_A u, v \rangle = 0$ if u and v have disjoint supports) and bijective.

Bijectivity implies that for any given $A \in \mathfrak{A}$ and $f \in H^{-1}(D)$ there exists a unique $u \in H_0^1(D)$ that solves the equation

$$\mathfrak{L}_A u = f \tag{2.1}$$

in a weak sense, i.e., the solution satisfies

$$\langle \mathfrak{L}_A u, v \rangle = \langle f, v \rangle \quad \text{for all } v \in H_0^1(D). \tag{2.2}$$

For given problem data A and f, we are interested in computing an approximation to u on some target scale in reasonable time.

2.2 Discretization

In order to be able to solve this problem computationally, we choose a finite-dimensional subspace $V_h \subseteq H_0^1(D)$ of dimension $m = \dim(V_h)$. As a standard example, one could take V_h to be a classical finite element space based on some mesh $\mathcal{T}_h$ with characteristic mesh size h and approximate (2.2) with a Galerkin method. However, in the very general setting with A possibly having fine oscillations on a scale that is not resolved by the mesh size h, this approach leads to unreliable approximations of u. Then again, the resolution of these fine-scale features can be prohibitively expensive in terms of computational resources if ε is very small. Note that resolution here may mean that the actual mesh size is significantly smaller than ε, depending on the oscillations of the coefficient and its regularity [8, 58]. This means that more advanced discretization techniques are required to still obtain reasonable approximations in the unresolved setting. In practice, the challenge is therefore to compress the fine-scale information that is contained in the operator $\mathfrak{L}_A$ to a suitable surrogate $\mathfrak{S}_A$ on the target scale h, i.e., the surrogate $\mathfrak{S}_A$ must be chosen in such a way that it is still able to capture the characteristic behavior of the operator $\mathfrak{L}_A$ on the scale of interest. Moreover, we require $\mathfrak{S}_A$ to be a bijection that maps the space V_h to itself. This ensures that for any $A \in \mathfrak{A}$ and $f \in H^{-1}(D)$ we can find unique $u_h \in V_h$ that weakly solves the discretized equation

$$\mathfrak{S}_A u_h = f_h, \tag{2.3}$$

with $f_h = \mathfrak{M} f$, where $\mathfrak{M}$ is a quadrature-type operator that maps a function in H^{-1} to an appropriate approximation in V_h. Problem (2.3) needs to be understood as finding u_h that satisfies

$$\langle \mathfrak{S}_A u_h, v_h \rangle = \langle f_h, v_h \rangle \quad \text{for all } v_h \in V_h.$$

The choice of the surrogate is obviously highly dependent on the problem at hand, see for example Sect. 3.2 for possible choices in the case of second-order elliptic diffusion operators.

2.3 Characterization in terms of a system matrix

We restrict our discussion to choices of surrogates that can be represented by an $m \times m$ system matrix S_A that is often called the *effective* system matrix in the following. We

assume that $S_A \in \mathbb{R}^{m\times m}$ is of the form $(S_A)_{ij} = \langle \mathfrak{S}_A \lambda_j, \lambda_i \rangle$ for a basis $\lambda_1, \dots, \lambda_m$ of V_h. Note that the basis should be chosen as localized as possible in order for the resulting system matrix to be sparse. The process of operator compression can then be formalized by a compression operator

$$\mathfrak{C}\colon \mathfrak{A} \to \mathbb{R}^{m\times m}$$

that maps a given coefficient A to the system matrix S_A representing the compressed surrogate $\mathfrak{S}_A$ of the operator $\mathfrak{L}_A$. Once $\mathfrak{C}$ has been evaluated for given $A \in \mathfrak{A}$, the solution to (2.2) can then be approximated with a function $u_h \in V_h$ for any right-hand side $f \in H^{-1}(D)$ by solving the linear system $S_A U = F$, where $F \in \mathbb{R}^m$ is the vector with entries $F_i := \langle \mathfrak{M} f, \lambda_i \rangle$ and $U \in \mathbb{R}^m$ contains the coefficients of the basis representation $u_h = \sum_{i=1}^m U_i \lambda_i$.

2.4 Multi-query scenarios

For many classes of coefficients $\mathfrak{A}$ and based on the choice of the surrogate, evaluating $\mathfrak{C}$ requires solving local auxiliary problems, during which the finest scale ε has to be resolved at some point. While this is acceptable if one wants to compress only a few operators in an offline computation, it becomes a major problem once $\mathfrak{C}$ has to be evaluated for many different coefficients A in an online phase, as for example in certain inverse problems, uncertainty quantification, or the simulation of evolution equations with time-dependent coefficients. This motivates a data-driven offline–online approach, where the offline phase consists of training a neural network to approximate the compression operator $\mathfrak{C}$, such that in the subsequent online phase the evaluation of $\mathfrak{C}$ can be replaced with a simple forward pass through the network, thus eliminating the computational bottleneck.

2.5 System matrix decomposition

In principle, one could try to directly approximate the global operator $\mathfrak{C}$ with a neural network. If the coefficient involves oscillations on some fine scale ε, this would lead to a network architecture with an input layer of size $\mathcal{O}(\varepsilon^{-d})$, an output layer of size $\mathcal{O}(m)$, and possible hidden layers. Particularly for small ε, this leads to very large networks and, thus, requires a huge amount of free parameters and therefore extraordinary amounts of training data and storage space in order to preserve good generalization capabilities.

To reduce the necessary size of the network, one can exploit available information on the compression operator $\mathfrak{C}$ by means of a certain structure in the resulting effective matrices S_A. To this end, we think of S_A as a matrix composed of multiple inflated sub-matrices, i.e.,

$$S_A = \sum_{j\in J} \Phi_j(S_{A,j}), \tag{2.4}$$

where J denotes some given index set and $S_{A,j} \in \mathbb{R}^{s\times t}$, $s, t \ll m$, are (typically dense) local matrices of equal size. The functions

$$\Phi_j\colon \mathbb{R}^{s\times t} \to \mathbb{R}^{m\times m} \tag{2.5}$$

represent local-to-global mappings inspired by classical finite element assembly processes as further explained below. More precisely, there exist index transformations π_j and φ_j,

such that

$$\Phi_j(S)\big[\pi_j(k), \varphi_j(l)\big] = S[k,l], \quad 1 \le k \le s, 1 \le l \le t,$$

where $M[i_r, i_c]$ denotes the entry of a matrix M in the i_r^{th} row and the i_c^{th} column. Note that π_j and φ_j can also map to zero, indicating that the corresponding entry should be disregarded. Let us also emphasize that the mappings Φ_j (and, in turn, the index transformations π_j and φ_j) are completely independent of coefficients A and solely depend on the domain D as well as the geometry of an allotted discretization. The precise definitions of the maps π_j and φ_j as well as the index set J are usually determined in a canonical way by the choice of the computational mesh and the compression operator $\mathfrak{C}$. In Sect. 3.4, we present an example of how such mappings may look like.

Depending on the compression operator $\mathfrak{C}$ and decomposition (2.4), we can expect that all the local matrices $S_{A,j}$ are created in a similar fashion and only depend on a local subsample of the coefficient. This can be understood as a generalization of the assembly process that underlies classical finite element system matrices: these matrices are composed of local system matrices that are computed on each element separately and only require knowledge about the coefficient on the respective element. In that context, the local submatrices all have a similar structure and the mapping by the functions Φ_j leads to overlapping contributions on the global level. Going back to the abstract setting, we generalize these properties and assume the existence of a lower-dimensional reduced compression operator

$$\mathfrak{C}_{\text{red}} \colon \mathbb{R}^r \to \mathbb{R}^{s \times t}, \tag{2.6}$$

such that the contributions $S_{A,j}$ are of the form

$$S_{A,j} = (\mathfrak{C}_{\text{red}} \circ R_j)(A), \tag{2.7}$$

where the operators

$$R_j \colon \mathfrak{A} \to \mathbb{R}^r \tag{2.8}$$

extract r relevant features of a given global coefficient. In the context of, e.g., finite element matrices, the operators R_j correspond to the restriction of a coefficient to an element-based piecewise constant approximation and $\mathfrak{C}_{\text{red}}$ incorporates the computation of a local system matrix based on such a sub-sample of the coefficient. To achieve a uniform length r of the output for the operators R_j, these operators may include artificially introduced zeros depending on the respective geometric configurations (e.g., at the boundary). An example for a quadrilateral mesh in two dimensions is shown in Fig. 1.

The problem of evaluating $\mathfrak{C}$ can now be decomposed into multiple evaluations of the reduced operator $\mathfrak{C}_{\text{red}}$ that takes the local information $R_j(A)$ of A and outputs a corresponding local matrix as described in (2.7). In our setting of a coefficient A that is potentially unresolved by the target scale h, evaluating $\mathfrak{C}_{\text{red}}$ is nontrivial and might become a bottleneck in multi-query scenarios as already indicated in Sect. 2.4. In such cases, we

propose to approximate the operator $\mathfrak{C}_{\text{red}}$ with a deep neural network

$$\Psi(\cdot,\theta) : \mathbb{R}^r \to \mathbb{R}^{s\times t},$$

where $\theta \in \mathbb{R}^p$ is a set of p trainable parameters of moderate size such that, for given $A \in \mathfrak{A}$, the effective system matrix $\mathfrak{C}(A) = S_A$ can be efficiently approximated by

$$\widehat{S}_A := \sum_{j\in J} \Phi_j\big(\Psi\big(R_j(A),\theta\big)\big),$$

which requires just a single forward pass of the minibatch $(R_j(A))_{j\in J}$ through the network. Note that the approximation $\widehat{S}_A$ possesses the same sparsity structure as the matrix S_A, since the neural network yields only approximations to the local sub-matrices $S_{A,j}$, whereas the assembling process which determines the sparsity structure of the global matrix is determined by the mappings Φ_j, which are independent of the network Ψ.

We emphasize that a decomposition of S_A as described in (2.4)–(2.8) does not necessarily require a uniform operator $\mathfrak{C}_{\text{red}}$. If multiple reduced operators are required for such a decomposition, the idea of approximating them by one single neural network can still be applied. It is, however, necessary for the ability of the network to generalize well beyond data seen during training that the reduced operators at least involve certain similarities.

2.6 Network training

In practice, the neural network $\Psi(\cdot,\theta)$ has to be trained in an offline phase from a set of training examples before it can be used for approximating the mapping $\mathfrak{C}_{\text{red}}$. We propose to draw N global coefficients $(A^{(i)})_{i=1}^N$ from $\mathfrak{A}$, extracting the relevant information $(A_j^{(i)}) := (R_j(A^{(i)}))$ from them and compressing it into the corresponding effective matrices $(S_{A,j}^{(i)})$ with $\mathfrak{C}_{\text{red}}$. This results in a total of $|J|\cdot N$ training samples available for the neural network to train on, namely $(A_j^{(i)}, S_{A,j}^{(i)})$, $i = 1,\dots N$, $j \in J$. In order to learn the parameters of the network, we then minimize the loss functional

$$\mathcal{J}(\theta) = \frac{1}{N\cdot|J|}\sum_{i=1}^{N}\sum_{j\in J}\frac{1}{2}\frac{\|\Psi(A_j^{(i)},\theta) - S_{A,j}^{(i)}\|_{\mathbb{R}^{s\times t}}^2}{\|S_{A,j}^{(i)}\|_{\mathbb{R}^{s\times t}}^2} \tag{2.9}$$

over the parameter space $\mathbb{R}^p$ using iterative gradient-based optimization on minibatches of the training data. This can be very efficiently implemented within modern deep learning frameworks such as *TensorFlow* [1], *PyTorch* [55], or *Flux* [40], which allow for the automatic differentiation of the loss functional with respect to the network parameters.

2.7 Full algorithm

After having established all the conceptual pieces, we now put them together and return to the abstract variational problem (2.2) from the beginning of the section. Suppose that we want to solve (2.2) for a large number of given coefficients $A^{(i)}$, $i = 1,\dots,M$, and a given right-hand side $f \in H^{-1}(D)$. For ease of notation, we restrict ourselves to a single right-hand side, which is, however, not necessarily required for our approach. The proposed procedure is summarized in Algorithm 1, divided into the offline and online stages of the method.

Algorithm 1 Operator compression with neural network

(i) **Offline phase**

Input: number of samples N, index set J, operators R_j, error functional $\mathcal{J}$ reduced compression operator $\mathfrak{C}_{\mathrm{red}}$, initial neural network $\Psi(\cdot,\theta)$,

Output: trained neural network $\Psi(\cdot,\theta)$

Draw N samples $(A^{(i)})_{i=1}^{N}$ from $\mathfrak{A}$
for $i = 1,\dots,N$ **do**
  for $j \in J$ **do**
    Extract relevant information from coefficient: $A_j^{(i)} = R_j(A^{(i)})$
    Compress to local matrix: $S_{A,j}^{(i)} = \mathfrak{C}_{\mathrm{red}}(A_j^{(i)})$
  end for
end for
Train neural network $\Psi(\cdot,\theta)$: update parameters θ based on data $(A_j^{(i)}, S_{A,j}^{(i)})_{i,j}$, $i = 1,\dots,N$, $j \in J$, by minimizing the error functional $\mathcal{J}$ over the parameter space $\mathbb{R}^p$
return $\Psi(\cdot,\theta)$

(ii) **Online phase**

Input: coefficients $(A^{(i)})_{i=1}^{M}$, index set J, operators R_j, trained network $\Psi(\cdot,\theta)$, right-hand side vector F, local-to-global mappings Φ_j

Output: coefficient vectors $(U^{(i)})_{i=1}^{M}$

for $i = 1,\dots,M$ **do**
  for $j \in J$ **do**
    Extract relevant information from coefficient: $A_j^{(i)} = R_j(A^{(i)})$
  end for
  Fast compression: $\widehat{S}_A^{(i)} = \sum_{j\in J} \Phi_j(\Psi(A_j^{(i)},\theta))$
  Solve: $U^{(i)} = (\widehat{S}_A^{(i)})^{-1} F$
end for
return $(U^{(i)})_{i=1}^{M}$

3 Application to elliptic homogenization

In this section, we specifically consider a family of prototypical elliptic diffusion operators as a demonstrating example of how to apply the abstract framework laid down in Sect. 2 in practice.

3.1 Setting

From now on let the domain D be polyhedral. We consider the family of linear second-order diffusion operators

$$\mathfrak{L} := \left\{ -\operatorname{div}(A\nabla\cdot) : H_0^1(D) \to H^{-1}(D) \mid A \in \mathfrak{A} \right\},$$

parameterized by the following set of admissible coefficients which possibly encode microstructures:

$$\mathfrak{A} := \left\{ A \in L^\infty(D) \mid \exists\, 0 < \alpha \le \beta < \infty : \alpha \le A(x) \le \beta \text{ for almost all } x \in D \right\}. \tag{3.1}$$

For the sake of simplicity, we restrict ourselves to scalar coefficients here. Note, however, that also the consideration of matrix-valued coefficients is not an issue from a numerical homogenization viewpoint. We remark that the family of operators $\mathfrak{L}$ fulfills the assumptions of locality and symmetry from the abstract framework. In this setting, the abstract problem (2.1) amounts to solving the following linear elliptic PDE with homogeneous Dirichlet boundary condition:

$$\begin{cases} -\operatorname{div}(A\nabla u) = f & \text{in } D, \\ u = 0 & \text{on } \partial D, \end{cases}$$

which possesses a unique weak solution $u \in H_0^1(D)$ for every $f \in H^{-1}(D)$ and $A \in \mathfrak{A}$. The corresponding counterpart to the weak formulation (2.2) can be written as: find $u \in H_0^1(D)$ such that

$$a_A(u,v) := \int_D A\nabla u \cdot \nabla v \,\mathrm{d}x = \langle f, v\rangle \quad \text{for all } v \in H_0^1(D) \tag{3.2}$$

by using integration by parts on the divergence term.

3.2 Discretization and compression

Let now $\mathcal{T}_h$ be a Cartesian mesh with characteristic mesh size h and denote with $Q^1(\mathcal{T}_h)$ the corresponding space of piecewise bilinear functions. We consider the conforming finite element space $V_h := Q^1(\mathcal{T}_h) \cap H_0^1(D)$ of dimension $m := \dim(V_h)$. Generally, also other types of meshes and finite element spaces could be employed, but we restrict ourselves to the above choice for the moment. As we have already mentioned in Sect. 2.2, if the mesh $\mathcal{T}_h$ does not resolve the fine-scale oscillations of A, approximating u with a pure finite element ansatz of seeking $u_h \in V_h$ such that

$$a_A(u_h, v_h) = \langle f, v_h\rangle \quad \text{for all } v_h \in V_h$$

will not yield satisfactory results. In a setting where resolving A with the mesh is computationally too demanding, we are therefore interested in suitable choices for a compression operator $\mathfrak{C}$. In particular, we want $\mathfrak{C}$ to produce effective system matrices on the target scale h that can be used to obtain appropriate approximations on this scale. In the following, we briefly comment on possible choices for this operator that are based on the finite element space V_h.

3.2.1 Compression by analytical homogenization

The idea of analytical homogenization is to replace an oscillating A with an appropriate homogenized coefficient $A_{\mathrm{hom}} \in L^\infty(D, \mathbb{R}^{d\times d})$. The mathematical theory of homogenization can treat very general nonperiodic coefficients in the framework of G- or H-convergence [14, 51, 64]. However, apart from being nonconstructive in many cases, homogenization in the classical analytical sense considers a sequence of operators $-\operatorname{div}(A_\varepsilon \nabla \cdot)$ indexed by $\varepsilon > 0$ and aims to characterize the limit as ε tends to zero. In many realistic applications, such a sequence of models can hardly be identified or may not be available in the first place. Assuming that the necessary requirements on the coefficient A are met, a homogenized coefficient A_{hom} exists and does not involve oscillations on a fine scale. The coefficient

A_{hom} can then be used in combination with a classical finite element ansatz, since A_{hom} does no longer include troublesome fine-scale quantities. In practice, the homogenized coefficients cannot be computed easily and need to be approximated. This is, for instance, done with the heterogeneous multiscale method (HMM) [2, 15, 16], which in the end replaces A with a computable approximation $A_h \in L^\infty(D, \mathbb{R}^{d\times d}_{\text{sym}})$ of A_{hom} with $(A_h)|_T \in \mathbb{R}^{d\times d}_{\text{sym}}$ for all $T \in \mathcal{T}_h$. With this piecewise constant approximation of A, we obtain a possible compression operator $\mathfrak{C}$. Given an enumeration $1,\dots,m$ of the inner nodes in $\mathcal{T}_h$ and writing $\lambda_1,\dots,\lambda_m$ for the associated nodal basis of V_h, the compressed operator $\mathfrak{C}(A)$ can be defined as

$$\big(\mathfrak{C}(A)\big)_{i,j} = (S_A)_{i,j} := \sum_{T\in\mathcal{T}_h} \int_T \big((A_h)|_T \nabla\lambda_j\big)\cdot\nabla\lambda_i \,\mathrm{d}x. \tag{3.3}$$

That is, one takes the classical finite element stiffness matrix corresponding to the homogenized coefficient A_{hom} as an effective system matrix. In this case, decomposition (2.4) corresponds to a partition into element-wise stiffness matrices (with constant coefficient, respectively) that are merged with a simple finite element assembly routine.

We emphasize that approaches based on analytical homogenization – such as (3.3) – are able to provide reasonable approximations on the target scale h but are subject to structural assumptions, in particular scale separation and local periodicity. The goal to overcome these restrictions has led to a new class of numerical methods that are specifically tailored to treating general coefficients with minimal assumptions. These methods are known as *numerical homogenization approaches* and typically only require a boundedness condition as in (3.1).

3.2.2 Compression by numerical homogenization

The general idea of numerical homogenization methods is to replace the trial space V_h with a suitable *multiscale space* $\tilde{V}_h$, see for instance the references [5, 7, 21, 38, 46, 52, 53]. One possible construction uses a one-to-one correspondence of $\tilde{V}_h$ to the space V_h, which implies that the two spaces possess the same number of degrees of freedom. Typically, the multiscale space is chosen in a problem-adapted way. We indicate this dependence by defining the new space $\tilde{V}_h := \mathcal{P}_A V_h$, where $\mathcal{P}_A\colon V_h \to H^1_0(D)$ particularly depends on A. Therefore, another possible choice of the operator $\mathfrak{C}$ leads to the effective matrix $\mathfrak{C}(A)$ given by

$$\big(\mathfrak{C}(A)\big)_{i,j} = (S_A)_{i,j} := a_A(\mathcal{P}_A\lambda_j, \lambda_i). \tag{3.4}$$

A prominent example for such an approach – and, thus, the operator $\mathfrak{C}$ – is the Petrov–Galerkin version of the localized orthogonal decomposition (LOD) method which explicitly constructs a suitable operator $\mathcal{P}_A$. The LOD was introduced in [46] and theoretically and practically works for very general coefficients. It has also been successfully applied to other problem classes, for instance, wave propagation problems in the context of Helmholtz and Maxwell equations [26, 27, 45, 57, 61] or the wave equation [4, 29, 44, 59], eigenvalue problems [47, 48], and in connection with time-dependent nonlinear Schrödinger equations [37]. However, it requires a slight deviation from locality. That is, while the classical finite element method and the HMM result in a system

matrix that only includes neighbor-to-neighbor communication between the degrees of freedom, the multiscale approach (3.4) moderately increases this communication to effectively incorporate the fine-scale information in A for a broader range of coefficients, which is a common property of modern homogenization techniques. As indicated in [12], this slightly increased communication indeed seems to be necessary to handle very general coefficients.

Since we consider a class $\mathfrak{A}$ of arbitrarily rough coefficients, the compression operator (3.4) corresponding to the operator $\mathcal{P}_A$ constructed in the LOD method is a suitable choice for our discussion as well as for the numerical experiments of Sect. 4. In the following subsection, we therefore have a closer look into its construction and summarize some main results. Note that we restrict ourselves to an elliptic model problem with homogeneous Dirichlet boundary conditions, but the compression approach can generally be extended to more involved settings such as the ones mentioned above.

3.3 Localized orthogonal decomposition

The method is based on a projective quasi-interpolation operator $\mathcal{I}_h\colon H_0^1(D) \to V_h$ with the following approximation and stability properties: for an element $T \in \mathcal{T}_h$, we require that

$$\left\|h^{-1}(v - \mathcal{I}_h v)\right\|_{L^2(T)} + \|\nabla \mathcal{I}_h v\|_{L^2(T)} \leq C\|\nabla v\|_{L^2(\mathbb{N}(T))}$$

for all $v \in H_0^1(D)$, where the constant C is independent of h, and $\mathbb{N}(S) := \mathbb{N}^1(S)$ is the neighborhood (of order 1) of $S \subseteq D$ defined by

$$\mathbb{N}^1(S) := \bigcup\{\overline{K} \in \mathcal{T}_h | \overline{S} \cap \overline{K} \neq \emptyset\}.$$

For a particular choice of $\mathcal{I}_h$, we refer to [24].

For given $\mathcal{I}_h$ with the above properties, we can define the so-called fine-scale space $\mathcal{W}$, which contains all functions that are not well captured by the finite element functions in V_h. It is defined as the kernel of $\mathcal{I}_h$ with respect to $H_0^1(D)$, i.e.,

$$\mathcal{W} := \ker \mathcal{I}_h|_{H_0^1(D)},$$

and its local version, for any $S \subseteq D$, is given by

$$\mathcal{W}(S) := \left\{w \in \mathcal{W} | \operatorname{supp}(w) \subseteq S\right\}.$$

In order to incorporate fine-scale information contained in the coefficient A, the idea is now to compute coefficient-dependent local corrections of functions $v_h \in V_h$. To this end, we define the neighborhood of order $\ell \in \mathbb{N}$ iteratively by $\mathbb{N}^\ell(S) := \mathbb{N}(\mathbb{N}^{\ell-1}(S))$, $\ell \geq 2$. For any function $v_h \in V_h$, its element corrector $\mathcal{Q}_{A,T}^\ell v_h \in \mathcal{W}(\mathbb{N}^\ell(T))$, $T \in \mathcal{T}_h$, is defined by

$$a_A\left(\mathcal{Q}_{A,T}^\ell v_h, w\right) = \int_T A\nabla v_h \cdot \nabla w \,\mathrm{d}x \quad \text{for all } w \in \mathcal{W}\left(\mathbb{N}^\ell(T)\right). \tag{3.5}$$

Note that in an implementation, the element corrections $\mathcal{Q}_{A,T}^\ell$ have to be computed on a sufficiently fine mesh that resolves the oscillations of the coefficient A. Since the algebraic realization of the correctors and guidelines for an efficient implementation of the

LOD method in general are not within the scope of the article, we refer to [23] for details. We emphasize that, by construction, the supports of the correctors $\mathcal{Q}_{A,T}^{\ell} v_h$ are limited to $\mathrm{N}^{\ell}(T)$. The global correction $\mathcal{Q}_A^{\ell}\colon V_h \to \mathcal{W}$ then consists of a summation of these local contributions and is given by

$$\mathcal{Q}_A^{\ell} := \sum_{T\in\mathcal{T}_h} \mathcal{Q}_{A,T}^{\ell}.$$

Note that the choice $\ell = \infty$ corresponds to a computation of the element corrections on the entire domain D and leads to the orthogonality property

$$a_A\big(\big(1-\mathcal{Q}_A^{\infty}\big)v_h, w\big) = 0 \quad \text{for all } w \in \mathcal{W}, \tag{3.6}$$

that defines an a_A-orthogonal splitting $H_0^1(D) = (1-\mathcal{Q}_A^{\infty})V_h \oplus \mathcal{W}$. This particularly explains the name *orthogonal decomposition*. The use of localized element corrections is motivated by the decay of the corrections $\mathcal{Q}_{A,T}^{\infty}$ away from the element T. This is, for instance, shown in [36, 56] (based on [46]) and reads

$$\big\|\nabla\big(\mathcal{Q}_A^{\infty} - \mathcal{Q}_A^{\ell}\big)v_h\big\|_{L^2(D)} \le Ce^{-c_{\mathrm{dec}}\ell}\|\nabla v_h\|_{L^2(D)}$$

with a constant c_{dec} which is independent of h and ℓ.

Motivated by decomposition (3.6) and the localized approximations in (3.5), we choose $\mathcal{P}_A := 1 - \mathcal{Q}_A^{\ell}$ in (3.4). The space $\tilde{V}_h := \mathcal{P}_A V_h = (1-\mathcal{Q}_A^{\ell})V_h$, which has the same number of degrees of freedom as V_h, can then be used as an ansatz space for the discretization of (3.2). Note that the original LOD method introduced in [46] considers a discretization where $\tilde{V}_h$ is also used as test space. We, however, consider the Petrov–Galerkin variant of the method as analyzed in [22] that uses the classical finite element space V_h as test space instead, i.e., we seek $u_h \in V_h$ such that

$$a_A\big(\big(1-\mathcal{Q}_A^{\ell}\big)u_h, v_h\big) = \langle f_h, v_h\rangle \quad \text{for all } v_h \in V_h, \tag{3.7}$$

where $f_h = \mathfrak{M}f \in V_h$ is again a suitable approximation of $f \in H^{-1}(D)$. This defines a compressed operator $\mathfrak{S}_A$ as in (2.3) that maps $u_h \in V_h$ to $f_h \in V_h$. As it turns out, the Petrov–Galerkin formulation has some computational advantages over the classical method, in particular in terms of memory requirement. For details, we again refer to [23]. The theory in [22] shows that the approximation u_h defined in (3.7) is first-order accurate in $L^2(D)$ provided that $\ell \gtrsim |\log h|$ and, additionally, $f \in L^2(D)$. More precisely, it holds

$$\begin{aligned}\big\|\big(-\operatorname{div}(A\nabla)\big)^{-1} - \mathfrak{S}_A^{-1}\big\|_{L^2(D)\to L^2(D)} &= \sup_{f\in L^2(D)} \frac{\|(-\operatorname{div}(A\nabla))^{-1}(f) - \mathfrak{S}_A^{-1}(\mathfrak{M}f)\|_{L^2(D)}}{\|f\|_{L^2(D)}} \\ &\lesssim h + e^{-c_{\mathrm{dec}}\ell},\end{aligned}$$

where $\mathfrak{M}f$ denotes the L^2-projection of f onto V_h. Note that the methodology can actually be applied to more general settings beyond the elliptic case, see for instance [49] for an overview.

3.4 System matrix surrogate

We now return to the discussion of the compression operator $\mathfrak{C}$ introduced in (3.4) that maps coefficients $A \in \mathfrak{A}$ to the effective system matrices

$$(S_A)_{i,j} := a_A\big(\big(1 - \mathcal{Q}_A^\ell\big)\lambda_j, \lambda_i\big)$$

obtained from the Petrov–Galerkin LOD method. Once S_A has been computed for a given A, an approximation $u_h = \sum_{j=1}^m U_j\lambda_j$ can be computed by solving the following linear system for the coefficients $U = (U_1, \dots, U_m)^T$:

$$S_A U = F,$$

where $F := (\langle \mathfrak{M}f, \lambda_1\rangle, \dots, \langle \mathfrak{M}f, \lambda_m\rangle)^T$. Since u_h is equivalently characterized by the solution of (3.7), it captures the effective behavior of the solution to the continuous problem (3.2) on the target scale h as discussed in the previous subsection.

The remainder of this section is dedicated to showing how the abstract decomposition (2.4) translates to the present LOD setting and how it can be implemented in practice. Writing $\mathcal{N}(S)$ for the set of inner mesh nodes on some subdomain $S \subseteq D$ and denoting $N_S = |\mathcal{N}(S)|$, the effective system matrix S_A can be decomposed as

$$S_A = \sum_{T\in\mathcal{T}_h} \Phi_T(S_{A,T}), \tag{3.8}$$

where the matrices $S_{A,T}$ are local system matrices of the form

$$(S_{A,T})_{i,j} = \int_{\mathbb{N}^\ell(T)} A\ \nabla\big(1 - \mathcal{Q}_{A,T}^\ell\big)\lambda_j \cdot \nabla\lambda_i \,\mathrm{d}x, \quad j \in \mathcal{N}(T),\ i \in \mathcal{N}\big(\mathbb{N}^\ell(T)\big), \tag{3.9}$$

i.e., they correspond to the interaction of the localized ansatz functions $(1 - \mathcal{Q}_{A,T}^\ell)\lambda_j$ associated with the nodes of the element T with the classical first order nodal basis functions whose supports overlap with the *element neighborhood* $\mathbb{N}^\ell(T)$. This means that $S_{A,T}$ is an $N_{\mathbb{N}^\ell(T)} \times N_T$ matrix. In practice, the coefficient A in (3.9) is often replaced with an elementwise constant approximation A_ε on a finer mesh $\mathcal{T}_\varepsilon$ that resolves all the oscillations of A and that we assume to be a uniform refinement of $\mathcal{T}_h$.

As already explained in the abstract framework, the mappings Φ_T in (3.8) are local-to-global mappings that assemble the contributions $S_{A,T}$ on an element neighborhood to a global matrix. In particular, given an enumeration $1, \dots, N_{\mathbb{N}^\ell(T)}$ of the nodes in $\mathcal{N}(\mathbb{N}^\ell(T))$, one considers a mapping $g_T(\cdot)$ that assigns to a given node index i in the element neighborhood $\mathbb{N}^\ell(T)$ its global node index $g_T(i) \in \{1, \dots, m\}$. This mapping can be represented by an $m \times N_{\mathbb{N}^\ell(T)}$ sparse matrix with entries

$$\pi_T[i,j] = \begin{cases} 1, & \text{if } i = g_T(j), \\ 0, & \text{otherwise.} \end{cases}$$

Analogously, given an enumeration $1, \dots, N_T$ of the nodes in $\mathcal{N}(T)$, there exists a mapping $\tilde{g}_T(\cdot)$ – represented by an $m \times N_T$-matrix – that assigns to a given node in $\mathcal{N}(T)$ with index

i its global representative with index $\tilde{g}_T(i)$. The corresponding matrix is given by

$$\varphi_T[i,j] = \begin{cases} 1, & \text{if } i = \tilde{g}_T(j), \\ 0, & \text{otherwise.} \end{cases}$$

Using these matrices, decomposition (3.8) reads

$$S_A = \sum_{T \in \mathcal{T}_h} \pi_T \, S_{A,T} \, \varphi_T', \tag{3.10}$$

where φ_T' denotes the transpose of the matrix φ_T.

From the definition of the local contributions $S_{A,T}$ introduced in (3.9), it directly follows that $S_{A,T}$ does only depend on the restriction of A, respectively A_ε, to the element neighborhood $\mathbb{N}^\ell(T)$. Let now $\mathcal{T}_\varepsilon(\mathbb{N}^\ell(T))$ be the restriction of the mesh $\mathcal{T}_\varepsilon$ to $\mathbb{N}^\ell(T)$, consisting of $r = |\mathcal{T}_\varepsilon(\mathbb{N}^\ell(T))|$ elements. Enumerating the elements then leads to the following operators that correspond to the abstract reduction operators in (2.8):

$$R_T : \mathfrak{A} \to \mathbb{R}^r \tag{3.11}$$

that map a global coefficient A to a vector that contains the values of A_ε in the respective cells of $\mathcal{T}_\varepsilon(\mathbb{N}^\ell(T))$.

As already mentioned in the abstract section above, we aim for a uniform output size of the operators R_T, since the outputs of the operators R_T will later on be fed into a neural network with a fixed number of input neurons. In order to achieve that, we artificially extend the domain D and the mesh $\mathcal{T}_h$ by ℓ layers of outer elements around the boundary elements of $\mathcal{T}_h$, thus ensuring that the element neighborhood $\mathbb{N}^\ell(T)$ always consists of the same number of elements regardless of the respective location of the central element $T \in \mathcal{T}_h$ relative to the boundary. Further, we extend the piecewise constant coefficient A_ε by zero on those outer elements. Figure 1 illustrates this procedure for the case $d = 2$ and $\ell = 1$ for an element T that lies in a corner of the computational domain. In this figure, $\mathcal{T}_h$ is a uniform quadrilateral mesh on the domain D and $\mathcal{T}_\varepsilon$ is obtained from $\mathcal{T}_h$ by a single

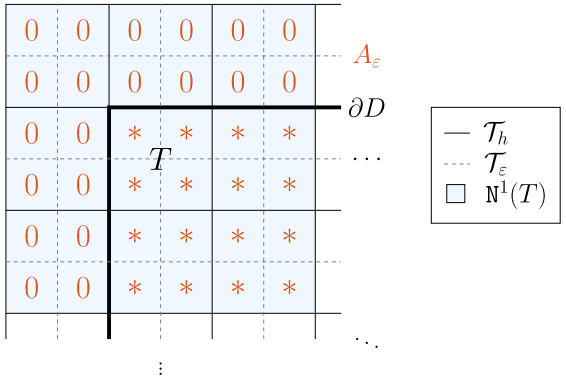

Figure 1 Illustration of the extended element neighborhood $\mathbb{N}^1(T)$ around a corner element $T \in \mathcal{T}_h$. An asterisk indicates that $A_\varepsilon|_K \in [\alpha, \beta]$, a zero that $A_\varepsilon|_K = 0$ in the respective cell K of the refined mesh $\mathcal{T}_\varepsilon$

uniform refinement step. The asterisks indicate the coefficient A_ε taking a regular value in the interval $[\alpha, \beta]$, whereas in the cells outside of D, we set A_ε to zero.

Note that this enlargement of the mesh $\mathcal{T}_h$ to obtain equally sized element neighborhoods $\mathbb{N}^\ell(T)$ also introduces artificial mesh nodes that lie outside of D and that are all formally considered as *inner* nodes for the definition of $N_S = |\mathcal{N}(S)|$ with a subset S in the extended domain. This implies that the local system matrices $S_{A,T}$ of dimension $N_{\mathbb{N}^\ell(T)} \times N_T$ introduced in (3.9) are all of equal size as well and the rows of $S_{A,T}$ corresponding to test functions associated with nodes that are attached to outer elements contain only zeros. During the assembly process of the local contributions to a global matrix, these zero rows are disregarded (which is also consistent with our definition of the matrices π_T, φ_T).

Finally, in order to unify the computation of local contributions, we use an abstract mapping $\mathfrak{C}_{\text{red}}$ with fixed input dimension r and fixed output dimension $N_{\mathbb{N}^\ell(T)} \times 2^d$ as proposed for the abstract framework in Sect. 2.5. The mapping takes the restriction of A_ε to an element neighborhood $\mathbb{N}^\ell(T)$ as input data and outputs the corresponding approximation of a local effective matrix $S_{A,T}$ that will be determined by an underlying neural network $\Psi(\cdot, \theta)$.

4 Numerical experiments

In this section, we demonstrate the feasibility of our proposed approach by performing numerical experiments in the setting of Sect. 3. For all experiments, we consider the two-dimensional computational domain $D = (0,1)^2$, which we discretize with a uniform quadrilateral mesh $\mathcal{T}_h$ of characteristic mesh size $h = 2^{-5}$. The coefficients are allowed to vary on the finer unresolved scale $\varepsilon = 2^{-8}$.

4.1 Coefficient family

In order to test our method's ability to deal with coefficients that show oscillating behavior across multiple scales, we introduce a hierarchy of meshes $\mathcal{T}^k$, $k = 0, 1, \dots, 8$, where the initial mesh $\mathcal{T}^0$ consists only of a single element, and the subsequent meshes are obtained by uniform refinement, i.e., $\mathcal{T}^k$ is obtained from $\mathcal{T}^{k-1}$ by subdiving each element of $\mathcal{T}^{k-1}$ into four equally sized elements. This implies that the characteristic mesh size of $\mathcal{T}^k$ is given by 2^{-k}. In the following, we refer to the parameter k as the *mesh level*. In particular, the computational mesh $\mathcal{T}_h = \mathcal{T}^5$ corresponds to the mesh level 5, whereas the coefficients may vary on the mesh level 8 and are therefore only resolved by the finest mesh $\mathcal{T}^8$. We thus have a scenario where an information gap of 3 mesh levels has to be bridged. Based on the mesh hierarchy, we now define the coefficient family $\mathfrak{A}$ of interest. Let $\mathfrak{A}_k$ denote the set of element-wise constant coefficients on $\mathcal{T}^k$ whose values in the respective cells are iid uniformly distributed on the interval $[\alpha, \beta] := [1, 5]$, i.e.,

$$\mathfrak{A}_k := \left\{ A \in Q^0(\mathcal{T}^k) | A|_T \overset{\text{iid}}{\sim} U([1,5]) \text{ for all } T \in \mathcal{T}^k \right\}.$$

Furthermore, let $\mathfrak{A}_{\text{ms}}$ denote the set of coefficients of the form

$$A = \frac{1}{9} \sum_{k=0}^{8} A_k, \quad A_k \in \mathfrak{A}_k.$$

These multiscale coefficients are especially interesting, since they vary on all considered scales simultaneously and are therefore very hard to deal with using classical homogeniza-

tion techniques due to their unstructured nature. The total set of interest $\mathfrak{A}$ is then defined as

$$\mathfrak{A} := \bigcup_{k=0}^{8} \mathfrak{A}_k \cup \mathfrak{A}_{\mathrm{ms}}.$$

In the following, we will frequently index coefficients sampled from $\mathfrak{A}$ by their corresponding level, i.e., write $A_k,\ k \in \{0,\dots,8,\mathrm{ms}\}$ instead of a plain A.

4.2 Data generation and preprocessing

In order to train the network, we sample 500 coefficients $A_k^{(i)},\ i = 1,\dots,500$, from each $\mathfrak{A}_k,\ k \in \{0,\dots,8,\mathrm{ms}\}$, where the individual samples $A_k^{(i)}$ on the coarser mesh levels $k = 0,\dots,7$ are prolongated to the finest mesh $\mathcal{T}^8$ in order to achieve a uniform dimension across all scales. The set of all sample coefficients is subsequently divided into a training, validation, and test set according to a 80–10–10 split. In order to achieve an identical distribution in all three sets, the splitting is performed separately on every level (including ms), i.e., for every $k \in \{0,\dots,8,\mathrm{ms}\}$, the first 400 coefficients $A_k^{(i)}, i = 1,\dots,400$, get assigned to the training set $\mathcal{D}_{\mathrm{train}}$, the samples with indices $401,\dots,450$ to the validation set $\mathcal{D}_{\mathrm{val}}$, and those with indices $451,\dots,500$ to the test set $\mathcal{D}_{\mathrm{test}}$. Then, we individually split each sample $A_k^{(i)}$, using the reduction operators R_T introduced in (3.11), into sub-samples $A_{k,T}^{(i)}$ based on element neighborhoods $\mathtt{N}^\ell(T)$ for $\ell = 2$ that are centered around the elements $T \in \mathcal{T}_h$, also taking into account the artificial extension of the element neighborhoods around the boundary of D. Since our target scale of interest is $h = 2^{-5}$ and $\mathcal{T}_h$ is a uniform quadrilateral mesh, this yields 1024 sub-samples $A_{k,T}^{(i)} \in \mathbb{R}^{1600}$ per sample $A_k^{(i)} \in \mathbb{R}^{65,536}$. Note that the size of the sub-samples is obtained from the construction of the local neighborhoods $\mathtt{N}^\ell(T)$. Here, each neighborhood consists of $(2\ell+1)^2 = 25$ elements in $\mathcal{T}_h = \mathcal{T}^5$, which corresponds to $64 \cdot 25 = 1600$ elements in the mesh $\mathcal{T}_\varepsilon = \mathcal{T}^8$.

The corresponding "labels", i.e., the local effective system matrices $S_{A,k,T}^{(i)} \in \mathbb{R}^{36\times 4}$, are then computed with the Petrov–Galerkin LOD according to (3.9) and flattened column-wise to vectors in $\mathbb{R}^{144}$. In total, we obtain $10 \cdot 400 \cdot 1024 = 4{,}096{,}000$ pairs $(A_{k,T}^{(i)}, S_{A,k,T}^{(i)}) \in \mathcal{D}_{\mathrm{train}}$ to train our network with, and 512,000 pairs in $\mathcal{D}_{\mathrm{val}}$ and $\mathcal{D}_{\mathrm{test}}$ each.

4.3 Network architecture and training

Given the above dataset, we now try to fit it with a suitable neural network Ψ. As network architecture, we consider a dense feedforward network with a total of eight layers including the input and output layer. As activation function, we choose the standard ReLU activation given by $\rho(x) := \max(0,x)$ in the first seven layers and the identity function in the last layer. By convention, the activation function acts component-wise on vectors. The network output is thus of the form

$$\Psi(x) = W^{(8)}\rho\big(W^{(7)}\big(\dots\rho\big(W^{(2)}\rho\big(W^{(1)}x + b^{(1)}\big) + b^{(2)}\big)\dots\big) + b^{(7)}\big) + b^{(8)}, \tag{4.1}$$

where the weight matrices and bias vectors have the following dimensions:

$$W^{(1)} \in \mathbb{R}^{1600\times 1600}, \quad W^{(2)} \in \mathbb{R}^{800\times 1600}, \quad W^{(3)} \in \mathbb{R}^{800\times 800}, \quad W^{(4)} \in \mathbb{R}^{400\times 800},$$
$$W^{(5)} \in \mathbb{R}^{400\times 400}, \quad W^{(6)} \in \mathbb{R}^{144\times 400}, \quad W^{(7)} \in \mathbb{R}^{144\times 144}, \quad W^{(8)} \in \mathbb{R}^{144\times 144},$$

$$b^{(1)} \in \mathbb{R}^{1600}, \quad b^{(2)}, \in \mathbb{R}^{800}, \quad b^{(3)} \in \mathbb{R}^{800}, \quad b^{(4)} \in \mathbb{R}^{400},$$
$$b^{(5)} \in \mathbb{R}^{400}, \quad b^{(6)} \in \mathbb{R}^{144}, \quad b^{(7)} \in \mathbb{R}^{144}, \quad b^{(8)} \in \mathbb{R}^{144},$$

yielding a total of 5,063,504 trainable parameters. The idea behind this architecture is that in the first six layers, information about the coefficient in the input element neighborhood is gathered and combined by allowing communication between all inputs in layers with odd indices, whereas in the layers with even indices, this information is repeatedly compressed. That is, every other layer is built in such a way that the input and output dimension are equal. If the neurons in that layer are understood as some sort of degrees of freedom in a mesh, this refers to having communication among all of these degrees of freedoms, while the layers in between reduce the number of degrees of freedom, which can be interpreted as transferring information to a coarser mesh. In the last two layers, this compressed information is taken and assembled to the local effective system matrix. Note that this logarithmic dependence of the number of layers on the number of scales that need to be bridged by the network (two layers per mesh level to be bridged plus two layers to assemble the local effective matrix) yielded the most reliable results in our experiments. Shallower networks had difficulties fitting the complex training set consisting of coefficients varying on different scales, whereas deeper networks were more prone to overfitting the training set. More involved architectures, for example the ones that include skip connections between layers like in the classic ResNet [34], are also conceivable; however, this seems not to be necessary to obtain good results. The key message here is that the coefficient-to-surrogate map can be satisfyingly approximated by a simple feedforward architecture, whose size does depend only on the scales ε and h, but not on any finer discretization scales. The implementation of the network as well as the training is performed using the library *Flux* [40] for the open-source scientific computing language *Julia* [10].

After initializing all parameters in the network according to a Glorot uniform distribution [32], network (4.1) is trained on minibatches of 1000 samples for a total of 20 epochs on $\mathcal{D}_{\text{train}}$, using the ADAM optimizer [42] with a step size of 10^{-4} for the first 5 epochs before reducing it to 10^{-5} for the subsequent 15 epochs. It could be observed that further training led to a stagnation of the validation error, whereas the error on the training set continued to decrease (very slowly but gradually), indicating overfitting of the network. The development of the loss functional $\mathcal{J}$ defined in (2.9) over the epochs is shown in Fig. 2. Note that training and validation loss stay very close to each other during the whole training process since $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{val}}$ have the same sample distribution due to our chosen splitting procedure. The development of the loss during the training and an average loss of $7.78 \cdot 10^{-5}$ on the test set $\mathcal{D}_{\text{test}}$ indicates that the network has at least learned to approximate the local effective system matrices. In applications, however, we are mostly concerned about how well this translates to the global level, when computing solutions to problem (3.2) using a global system matrix assembled from network outputs. In order to investigate this question, the next three subsections are dedicated to evaluating the performance of the trained network at exactly this task for several coefficients unseen during training. For a given right-hand side f and coefficient A, we denote with u_h the solution of (3.7), obtained with the Petrov–Galerkin LOD matrix S_A defined in (3.10), and with $\widehat{u}_h$ the solution obtained by using the neural network approximation of this matrix, i.e.,

$$\widehat{S}_A := \sum_{T \in \mathcal{T}_h} \pi_T \, \Psi\big(R_T(A)\big) \, \varphi_T'. \tag{4.2}$$

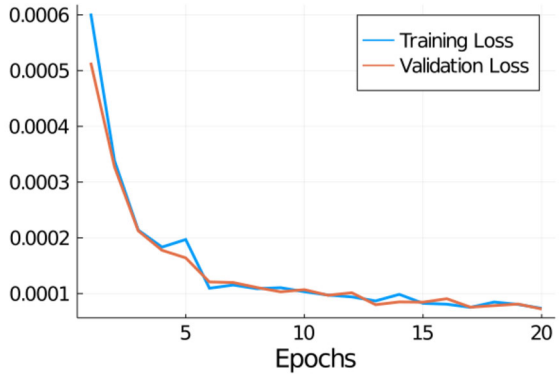

Figure 2 Development of loss functional $\mathcal{J}$ during 20 epochs of the network training

The spectral norm difference $\|S_A - \widehat{S}_A\|_2$, the L^2-error $\|u_h - \widehat{u}_h\|_{L^2(D)}$ as well as the visual discrepancy between the two solutions are then considered as a measure of the network's global performance. We emphasize once more that computing approximate surrogates via (4.2) is significantly faster compared to (3.8) and (3.9). This is due to the fact that no corrector problems of the form (3.5) have to be solved to obtain the surrogate model. As pointed out, the solution of these local problems requires inversion on a very fine discretization scale that is significantly smaller than the scale ε on which the coefficient varies. In order to compute the system matrix S_A, one has to solve $N_{\mathcal{T}_h}$ fine-scale linear systems, where $N_{\mathcal{T}_h}$ denotes the number of elements in $\mathcal{T}_h$. In contrast, the main computational effort of evaluating our trained network consists of N_L matrix-matrix multiplications, where N_L is the number of layers in the network (not taking into account bias vectors and the activation function).

4.4 Experiment 1

For our first experiment, we consider an unstructured multiscale coefficient sampled from $\mathfrak{A}_{\text{ms}}$ that was not a part of the training set and a constant right-hand side $f \equiv 1$. The coefficient (top left), the error $|u_h(x) - \widehat{u}_h(x)|$ (top right) as well as representative cross-sections along $x_1 = 0.5$ (bottom left) and $x_2 = 0.5$ (bottom right) of the two solutions u_h and $\widehat{u}_h$ are shown in Fig. 3. The spectral norm difference $\|S_A - \widehat{S}_A\|_2 \approx 6.58 \cdot 10^{-2}$ and the L^2-error $\|u_h - \widehat{u}_h\|_{L^2(D)} \approx 2.13 \cdot 10^{-4}$ confirm the visual impression – the network has successfully learned to produce a system matrix that is able to capture the behavior of the solution on the target scale well.

4.5 Experiment 2

Next, we test the network's performance for smoother and more regular coefficients than the ones it has been trained with. As a demonstrating example, we consider the coefficient $A(x) = 2 + \sin(2\pi x_1)\sin(2\pi x_2)$. The network's input is obtained by evaluating A on the midpoints of the mesh $\mathcal{T}_\varepsilon$ on the fine unresolved scale ε. In this example, we choose the function $f(x) = x_1 \chi_{\{x_1 \geq 0.5\}}$ as a right-hand side, where χ_S denotes the characteristic function of the set $S \subseteq D$. The results are shown in Fig. 4. We obtain an even better L^2-error of $\|u_h - \widehat{u}_h\|_{L^2(D)} \approx 7.56 \cdot 10^{-5}$ and a spectral norm difference of $\|S_A - \widehat{S}_A\|_2 \approx 3.91 \cdot 10^{-2}$. A comparison between the solutions at the cross-sections shows that there is almost no

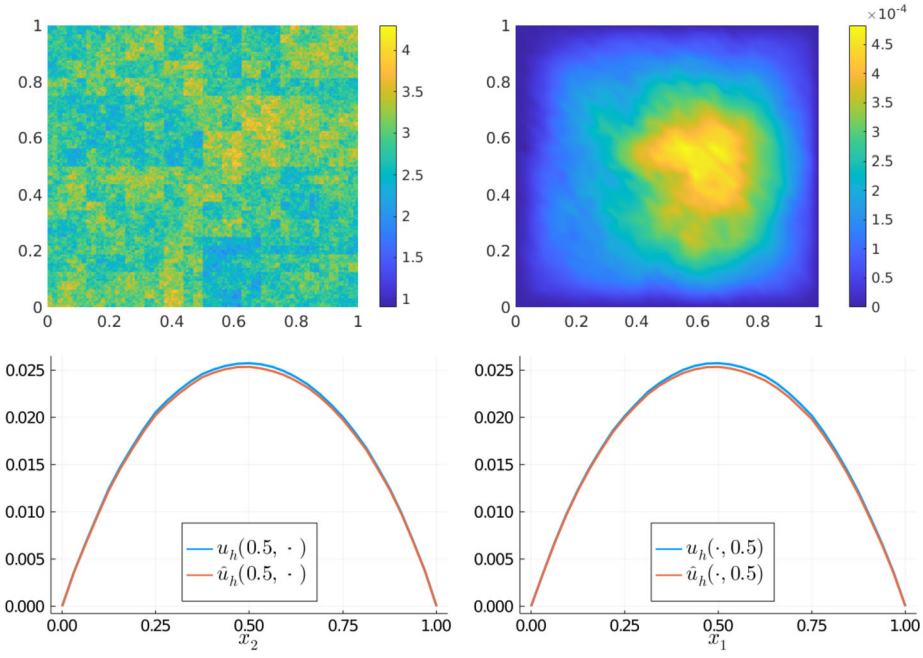

Figure 3 Results of Experiment 1: Unstructured multiscale coefficient sampled from $\mathfrak{A}_{\text{ms}}$ (top left), $|u_h(x) - \widehat{u}_h(x)|$ (top right), and comparison of u_h vs $\widehat{u}_h$ along the cross sections $x_1 = 0.5$ (bottom left) and $x_2 = 0.5$ (bottom right)

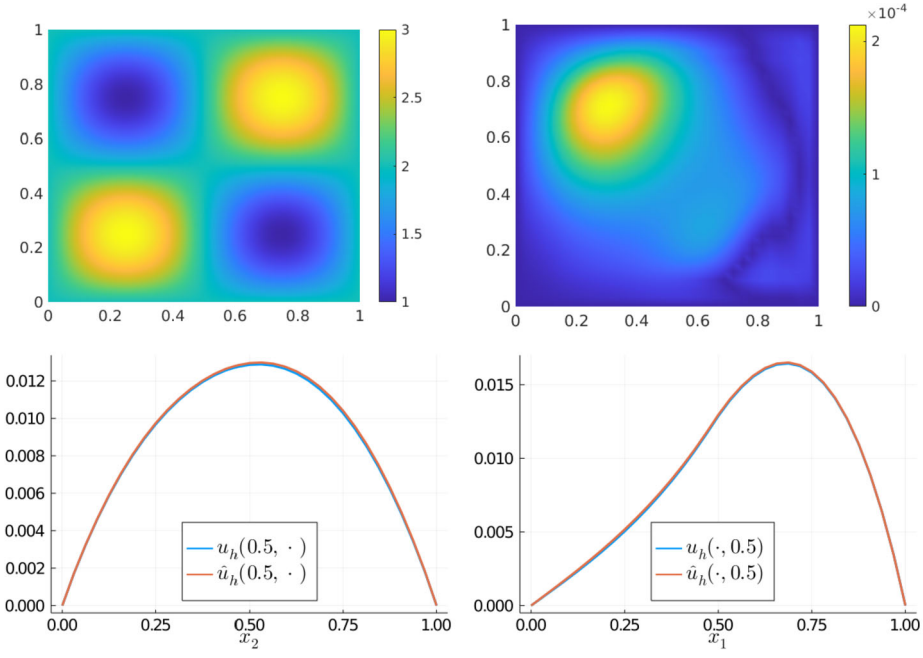

Figure 4 Results of Experiment 2: Smooth coefficient (top left), $|u_h(x) - \widehat{u}_h(x)|$ (top right), and comparison of u_h vs $\widehat{u}_h$ along the cross sections $x_1 = 0.5$ (bottom left) and $x_2 = 0.5$ (bottom right)

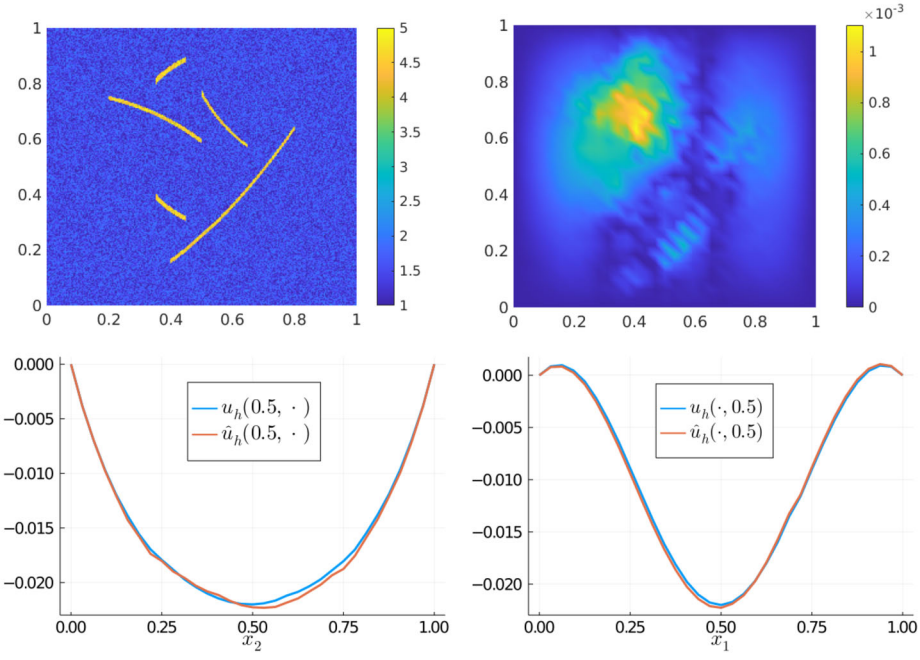

Figure 5 Results of Experiment 3: Coefficient with cracks (top left), $|u_h(x) - \widehat{u}_h(x)|$ (top right), and comparison of u_h vs $\widehat{u}_h$ along the cross sections $x_1 = 0.5$ (bottom left) and $x_2 = 0.5$ (bottom right)

discernible visual difference between the LOD-solution and the approximation obtained using our trained network.

4.6 Experiment 3

As a third experiment, we choose another coefficient that possesses an unfamiliar structure not seen by the network during the training phase, this time a less regular one. The coefficient is shown in the top left of Fig. 5. It is composed of a background part (blue region), which is obtained by sampling uniformly and independently on each element of $\mathcal{T}_\varepsilon$ from the interval $[1, 2]$, and several "cracks" (yellow regions), in which the coefficient varies uniformly in $[4, 5]$. The right-hand side here is $f(x) = \cos(2\pi x_1)$. A computation of the L^2-error $\|u_h - \widehat{u}_h\|_{L^2(D)} \approx 2.72 \cdot 10^{-4}$ shows that the overall error is still moderate; a closer visual inspection of the solutions along the cross-sections however reveals more prominent deviations of the neural network approximation to the ground truth. The spectral norm difference $\|S_A - \widehat{S}_A\|_2 \approx 2.81 \cdot 10^{-1}$ is also one order of magnitude larger than in the previous examples. Nevertheless, it might be possible that performing a few corrective training steps including samples of this nature would be sufficient to fix this issue. A thorough investigation of this hypothesis, along with the extension to other coefficient classes is subject of future work.

5 Conclusion and outlook

We proposed an approach to the compression of linear differential operators – parameterized by PDE coefficients that may depend on microscopic quantities that are not resolved by a target discretization scale of interest – to lower-dimensional surrogates that are based on a combination of existing model reduction methods with a data-driven deep learning framework. Our method is motivated by the fact that the computation of the

surrogates (represented by effective system matrices) via classical methods is nontrivial and requires significant computational resources in multi-query settings. To overcome this problem, we showed how the compression process can be approximated by a neural network based on given training data that can be generated using existing compression approaches. Importantly, we avoid a global approximation by a neural network and instead first decompose the compression map into local contributions, which can then be approximated by one single unified network. As an example, we studied a class of second-order elliptic diffusion operators. We showed how to approximate the compression map based on the Petrov–Galerkin formulation of the localized orthogonal decomposition method with a neural network. The proposed ansatz has been numerically validated for a large set of piecewise constant and highly oscillatory multiscale coefficients. Furthermore, it has been shown that the approach also generalizes well, in the sense that a well-trained network is able to produce reasonable results even for classes of coefficients that it has not been trained on.

For future research, many possible research questions building on the present work are conceivable. Straightforward extensions would be to consider stochastic settings with differential operators parameterized by random fields or settings with high contrast. Another question to investigate is to what degree the method can be made robust against changes in geometry, for example, by training the network not only on coefficients that are sampled on a fixed domain, but rather on reference patches with varying geometries. Mimicking a hierarchical discretization approach, one may also try to directly approximate the inverse operator which can be represented by a sparse matrix [25]. On a more theoretical level, the approximation properties of neural networks for various existing compression operators could be investigated, along with the question of the number of training samples required to faithfully approximate those for a given family of coefficients.

Acknowledgements
Not applicable.

Funding
The work of F. Kröpfl and D. Peterseim is part of the project that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant agreement No. 865751 - *Computational Random Multiscale Problems*). R. Maier acknowledges support by the Göran Gustafsson Foundation for Research in Natural Sciences and Medicine. Open Access funding enabled and organized by Projekt DEAL.

Availability of data and materials
The dataset generated and used for the numerical experiments in this work are available from the corresponding author F. Kröpfl on reasonable request.

Declarations

Competing interests
The authors declare that they have no competing interests.

Authors' contributions
The work emerged from a close collaboration between the authors. All authors read and approved the final manuscript.

Author details
[1]Institute of Mathematics, University of Augsburg, Universitätsstr. 12a, 86159 Augsburg, Germany. [2]Institute of Mathematics, Friedrich Schiller University Jena, Ernst-Abbe-Platz 2, 07743 Jena, Germany. [3]Centre for Advanced Analytics and Predictive Sciences (CAAPS), University of Augsburg, Universitätsstr. 12a, 86159 Augsburg, Germany.

Publisher's Note

Received: 14 July 2021 Accepted: 26 March 2022 Published online: 09 April 2022

References

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., TensorFlow, X.Z.: Large-Scale Machine Learning on Heterogeneous Systems (2015) Software available from tensorflow.org
2. Abdulle, A., E, W., Engquist, B., Vanden-Eijnden, E.: The heterogeneous multiscale method. Acta Numer. **21**, 1–87 (2012)
3. Abdulle, A., Henning, P.: A reduced basis localized orthogonal decomposition. J. Comput. Phys. **295**, 379–401 (2015)
4. Abdulle, A., Henning, P.: Localized orthogonal decomposition method for the wave equation with a continuum of scales. Math. Comput. **86**(304), 549–587 (2017)
5. Altmann, R., Henning, P., Peterseim, D.: Numerical homogenization beyond scale separation. Acta Numer. **30**, 1–86 (2021)
6. Arbabi, H., Bunder, J.E., Samaey, G., Roberts, A.J., Kevrekidis, I.G.: Linking machine learning with multiscale numerics: data-driven discovery of homogenized equations. JOM **72**(12), 4444–4457 (2020)
7. Babuška, I., Lipton, R.: Optimal local approximation spaces for generalized finite element methods with application to multiscale problems. Multiscale Model. Simul. **9**(1), 373–406 (2011)
8. Babuška, I., Osborn, J.E.: Can a finite element method perform arbitrarily badly? Math. Comput. **69**(230), 443–462 (2000)
9. Berner, J., Dablander, M., Grohs, P.: Numerically solving parametric families of high-dimensional Kolmogorov partial differential equations via deep learning. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.F., Lin, H. (eds.) Advances in Neural Information Processing Systems, vol. 33, pp. 16615–16627. Curran Associates, Red Hook (2020)
10. Bezanson, J., Edelman, A., Karpinski, S., Shah, V.B.: Julia: a fresh approach to numerical computing. SIAM Rev. **59**(1), 65–98 (2017)
11. Bhattacharya, K., Hosseini, B., Kovachki, N.B., Stuart, A.M.: Model reduction and neural networks for parametric PDEs. SMAI J. Comput. Math. **7**, 121–157 (2021)
12. Caiazzo, A., Maier, R., Peterseim, D.: Reconstruction of quasi-local numerical effective models from low-resolution measurements. J. Sci. Comput. **85**(1), Article ID 10 (2020)
13. Chan, S., Elsheikh, A.H.: A machine learning approach for efficient uncertainty quantification using multiscale methods. J. Comput. Phys. **354**, 493–511 (2018)
14. De Giorgi, E.: Sulla convergenza di alcune successioni d'integrali del tipo dell'area. Rend. Mat. **6**(8), 277–294 (1975)
15. E, W., Engquist, B.: The heterogeneous multiscale methods. Commun. Math. Sci. **1**(1), 87–132 (2003)
16. E, W., Engquist, B.: The heterogeneous multi-scale method for homogenization problems. In: Multiscale Methods in Science and Engineering. Lect. Notes Comput. Sci. Eng., vol. 44, pp. 89–110. Springer, Berlin (2005)
17. E, W., Han, J., Jentzen, A.: Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. Commun. Math. Stat. **5**(4), 349–380 (2017)
18. E, W., Han, J., Jentzen, A.: Algorithms for solving high dimensional PDEs: from nonlinear Monte Carlo to machine learning. Nonlinearity **35**(1), 278–310 (2021)
19. E, W., Yu, B.: The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems. Commun. Math. Stat. **6**(1), 1–12 (2018)
20. Efendiev, Y.R., Galvis, J., Wu, X.-H.: Multiscale finite element methods for high-contrast problems using local spectral basis functions. J. Comput. Phys. **230**(4), 937–955 (2011)
21. Efendiev, Y.R., Hou, T.Y.: Multiscale Finite Element Methods: Theory and Applications. Surveys and Tutorials in the Applied Mathematical Sciences, vol. 4. Springer, New York (2009)
22. Elfverson, D., Ginting, V., Henning, P.: On multiscale methods in Petrov-Galerkin formulation. Numer. Math. **131**(4), 643–682 (2015)
23. Engwer, C., Henning, P., Målqvist, A., Peterseim, D.: Efficient implementation of the localized orthogonal decomposition method. Comput. Methods Appl. Mech. Eng. **350**, 123–153 (2019)
24. Ern, A., Guermond, J.-L.: Finite element quasi-interpolation and best approximation. ESAIM: Math. Model. Numer. Anal. **51**(4), 1367–1385 (2017)
25. Feischl, M., Peterseim, D.: Sparse compression of expected solution operators. SIAM J. Numer. Anal. **58**(6), 3144–3164 (2020)
26. Gallistl, D., Henning, P., Verfürth, B.: Numerical homogenization of H(curl)-problems. SIAM J. Numer. Anal. **56**(3), 1570–1596 (2018)
27. Gallistl, D., Peterseim, D.: Stable multiscale Petrov–Galerkin finite element method for high frequency acoustic scattering. Comput. Methods Appl. Math. **295**, 1–17 (2015)
28. Gao, H., Sun, L., Wang, J.-X.: Phygeonet: physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state PDEs on irregular domain. J. Comput. Phys. **428**, 110079 (2021)
29. Geevers, S., Maier, R.: Fast mass lumped multiscale wave propagation modelling. IMA J. Numer. Anal. (2022) To appear
30. Geist, M., Petersen, P., Raslan, M., Schneider, R., Kutyniok, G.: Numerical solution of the parametric diffusion equation by deep neural networks. J. Sci. Comput. (2022) To appear
31. Ghavamian, F., Simone, A.: Accelerating multiscale finite element simulations of history-dependent materials using a recurrent neural network. Comput. Methods Appl. Math. **357**, 112594 (2019)
32. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the 13th International Conference on Artificial Intelligence and Statistics. JMLR Workshop and Conference Proceedings, pp. 249–256 (2010)
33. Han, J., Jentzen, A., E, W.: Solving high-dimensional partial differential equations using deep learning. Proc. Natl. Acad. Sci. **115**(34), 8505–8510 (2018)
34. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
35. Hellman, F., Keil, T., Målqvist, A.: Numerical upscaling of perturbed diffusion problems. SIAM J. Sci. Comput. **42**(4), A2014–A2036 (2020)

36. Henning, P., Peterseim, D.: Oversampling for the multiscale finite element method. Multiscale Model. Simul. **11**(4), 1149–1175 (2013)
37. Henning, P., Wärnegård, J.: Superconvergence of time invariants for the Gross-Pitaevskii equation. Math. Comput. **91**(334), 509–555 (2022)
38. Hou, T.Y., Wu, X.-H.: A multiscale finite element method for elliptic problems in composite materials and porous media. J. Comput. Phys. **134**(1), 169–189 (1997)
39. Hutzenthaler, M., Jentzen, A., Kruse, T., Nguyen, T.A.: A proof that rectified deep neural networks overcome the curse of dimensionality in the numerical approximation of semilinear heat equations. Ser. Partial Differ. Equ. Appl. **1**(2), 1–34 (2020)
40. Innes, M.: Flux: elegant machine learning with Julia. J. Open Sour. Softw. (2018)
41. Khoo, Y., Lu, J., Ying, L.: Solving parametric PDE problems with artificial neural networks. Eur. J. Appl. Math. **32**(3), 421–435 (2021)
42. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization (2014). Preprint arXiv:1412.6980
43. Kutyniok, G., Petersen, P., Raslan, M., Schneider, R.: A theoretical analysis of deep neural networks and parametric PDEs. Constr. Approx. (2022) To appear
44. Maier, R., Peterseim, D.: Explicit computational wave propagation in micro-heterogeneous media. BIT Numer. Math. **59**(2), 443–462 (2019)
45. Maier, R., Verfürth, B.: Multiscale scattering in nonlinear Kerr-type media. Math. Comput. (2022) To appear
46. Målqvist, A., Peterseim, D.: Localization of elliptic multiscale problems. Math. Comput. **83**(290), 2583–2603 (2014)
47. Målqvist, A., Peterseim, D.: Computation of eigenvalues by numerical upscaling. Numer. Math. **130**(2), 337–361 (2015)
48. Målqvist, A., Peterseim, D.: Generalized finite element methods for quadratic eigenvalue problems. ESAIM: Math. Model. Numer. Anal. **51**(1), 147–163 (2017)
49. Målqvist, A., Peterseim, D.: Numerical Homogenization by Localized Orthogonal Decomposition. SIAM Spotlights, vol. 5. SIAM, Philadelphia (2020)
50. Målqvist, A., Verfürth, B.: An offline-online strategy for multiscale problems with random defects. ESAIM: Math. Model. Numer. Anal. (2022) To appear
51. Murat, F., Tartar, L.: H-convergence. In: Séminaire d'Analyse Fonctionnelle et Numérique de l'Université d'Alger (1978)
52. Owhadi, H.: Multigrid with rough coefficients and multiresolution operator decomposition from hierarchical information games. SIAM Rev. **59**(1), 99–149 (2017)
53. Owhadi, H., Zhang, L., Berlyand, L.: Polyharmonic homogenization, rough polyharmonic splines and sparse super-localization. ESAIM: Math. Model. Numer. Anal. **48**(2), 517–552 (2014)
54. Padmanabha, G.A., Zabaras, N.: A Bayesian multiscale deep learning framework for flows in random media. Found. Data Sci. **3**(2), 251–303 (2021)
55. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., PyTorch, S.C.: An imperative style, high-performance deep learning library. In: Wallach, H., Larochelle, H., Beygelzimer, A., Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 32, pp. 8024–8035. Curran Associates, Red Hook (2019)
56. Peterseim, D.: Variational multiscale stabilization and the exponential decay of fine-scale correctors. In: Building Bridges: Connections and Challenges in Modern Approaches to Numerical Partial Differential Equations. Lect. Notes Comput. Sci. Eng., vol. 114, pp. 341–367. Springer, Cham (2016)
57. Peterseim, D.: Eliminating the pollution effect in Helmholtz problems by local subscale correction. Math. Comput. **86**(305), 1005–1036 (2017)
58. Peterseim, D., Sauter, S.A.: Finite elements for elliptic problems with highly varying, nonperiodic diffusion matrix. Multiscale Model. Simul. **10**(3), 665–695 (2012)
59. Peterseim, D., Schedensack, M.: Relaxing the CFL condition for the wave equation on adaptive meshes. J. Sci. Comput. **72**(3), 1196–1213 (2017)
60. Raissi, M., Perdikaris, P., Karniadakis, G.E.: Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J. Comput. Phys. **378**, 686–707 (2019)
61. Ren, X., Hannukainen, A., Belahcen, A.: Homogenization of multiscale Eddy current problem by localized orthogonal decomposition method. IEEE Trans. Magn. **55**(9), 1–4 (2019)
62. Schwab, C., Zech, J.: Deep learning in high dimension: neural network expression rates for generalized polynomial chaos expansions in UQ. Anal. Appl. **17**(01), 19–55 (2019)
63. Sirignano, J., Spiliopoulos, K.: DGM: a deep learning algorithm for solving partial differential equations. J. Comput. Phys. **375**, 1339–1364 (2018)
64. Spagnolo, S.: Sulla convergenza di soluzioni di equazioni paraboliche ed ellittiche. Ann. Sc. Norm. Super. Pisa, Cl. Sci. **3**(22), 571–597 (1968)
65. Wang, Y., Cheung, S.W., Chung, E.T., Efendiev, Y., Wang, M.: Deep multiscale model learning. J. Comput. Phys. **406**, 109071 (2020)