Preprint ISSN 2198-5855

Multilevel CNNs for parametric pdes based on adaptive finite elements

Janina Schütte, Martin Eigel

submitted: August 30, 2024

Weierstrass Institute
Mohrenstr. 39
10117 Berlin
Germany
E-Mail: janina.schuette@wias-berlin.de
martin.eigel@wias-berlin.de

No. 3124
Berlin 2024

2020 *Mathematics Subject Classification.* 35R60, 65N22, 65J10, 68T07, 68T05, 65N15, 65N30, 65N50, 65N55.

Key words and phrases. Parametric PDE, adaptive finite element method, multilevel decomposition, neural network approximation.

We acknowledge funding from the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) in the the priority programme SPP 2298 "Theoretical Foundations of Deep Learning".

Edited by
Weierstraß-Institut für Angewandte Analysis und Stochastik (WIAS)
Leibniz-Institut im Forschungsverbund Berlin e. V.
Mohrenstraße 39
10117 Berlin
Germany

Fax: +49 30 20372-303
E-Mail: `preprint@wias-berlin.de`
World Wide Web: `http://www.wias-berlin.de/`

Multilevel CNNs for parametric pdes based on adaptive finite elements

Janina Schütte, Martin Eigel

Abstract

A neural network architecture is presented that exploits the multilevel properties of high-dimensional parameter-dependent partial differential equations, enabling an efficient approximation of parameter-to-solution maps, rivaling best-in-class methods such as low-rank tensor regression in terms of accuracy and complexity. The neural network is trained with data on adaptively refined finite element meshes, thus reducing data complexity significantly. Error control is achieved by using a reliable finite element a posteriori error estimator, which is also provided as input to the neural network.

The proposed U-Net architecture with CNN layers mimics a classical finite element multigrid algorithm. It can be shown that the CNN efficiently approximates all operations required by the solver, including the evaluation of the residual-based error estimator. In the CNN, a culling mask set-up according to the local corrections due to refinement on each mesh level reduces the overall complexity, allowing the network optimization with localized fine-scale finite element data.

A complete convergence and complexity analysis is carried out for the adaptive multilevel scheme, which differs in several aspects from previous non-adaptive multilevel CNN. Moreover, numerical experiments with common benchmark problems from Uncertainty Quantification illustrate the practical performance of the architecture.

1 Introduction

In recent years, the intersection of partial differential equations (PDEs) and neural networks has emerged as a powerful and promising field of research. In a wider sense, this increasingly popular research area is called scientific machine learning (SciML), which strives to make use of modern deep learning methods for the solution of differential equations that are common to model physical processes in engineering and the natural sciences. Opposite to many tasks in classification or generation of images, videos, sounds or text, the data in SciML typically has specific properties that can be exploited, e.g. regularity or sparsity of functions. Moreover, data often can be generated synthetically by running (possibly computationally very costly) simulations with classical solvers such as finite elements (FE).

We consider parametric PDEs as a flexible mathematical model to describe real-world phenomena, allowing for the incorporation of variable, stochastic parameters that capture uncertainties and changing properties. Problems of this type have been examined extensively in Uncertainty Quantification (UQ) in recent years. They can be approached with sampling methods or by computing functional surrogates in different model classes such as low-rank tensors [17, 15], by which a larger part of or the entire statistics of the quantity of interest is approximated. Neural network surrogate models in an infinite-dimensional setting have been analyzed, e.g. the DeepONet (deep operator network) architecture in [9, 30, 33, 43, 35, 39], neural operators based on model reduction in [2], and the FNO (Fourier neural operator) in [32, 28] and references therein. In a discretized setting the problem is combined with reduced basis methods in [29, 21, 10]. In [5, 6], adaptively created meshes are used to train a fully connected neural network mapping, the parameter and the point in the physical domain to the evaluation of the corresponding solution. A multilevel collocation approach to the pPDE problem can be found in [41] and a neural network multilevel method for recovering a quantity of interest is presented in [34].

Many results on NN parameter complexity estimates for function approximation are based on the pivotal work [46], where it is shown that NNs with a ReLU activation function are able to efficiently represent polynomials. In this work (as in [27]) the analysis is based on an approximation of the multiplication operator with a fixed number of trainable parameters independent of the desired accuracy as shown in [25, Corollary C.3]. Here, it is assumed that the activation function is three times continuously differentiable in a neighborhood of some point with nonzero second derivative, see Assumption 5.1. Then parameter bounds for an architecture as described in [38] can be derived.

1.1 Adaptive neural network approach

In this paper we present an approach to solve parametric PDEs based on training data generated by an adaptive FE discretization. This is combined with a multilevel neural network (ML-Net) architecture, which mimics a classical multilevel solver and supports local corrections, corresponding to local mesh refinements.

In [27] the ML-Net architecture is derived to approximate the finite element coefficients of the solutions on uniformly refined grids. We generalize this approach by introducing local corrections with respect to the global discretization mesh with data being generated by an efficient adaptive solver. We show that CNNs are able to efficiently approximate a posteriori finite element error estimators and construct a culling mask based on the estimations, which only adds parts of the domain where fine scale corrections are needed. By this, only small parts on each level are considered in the multigrid scheme. As a consequence of this data reduction, in principle much finer meshes (and hence a higher approximation accuracy) can be used for the training.

We consider the *parametric stationary diffusion PDE* (also known as "parametric Darcy problem") with a possibly countably infinite dimensional parameter space $\Gamma \subset \mathbb{R}^{\mathbb{N}}$ and a physical domain $D \subset \mathbb{R}^d, d \in \{1, 2\}$. The objective is to find $u : D \times \Gamma \to \mathbb{R}$ such that

$$\begin{aligned} -\nabla \cdot (\kappa(\cdot, \mathbf{y})\nabla u(\cdot, \mathbf{y})) &= f \qquad \text{on } D,\\ u(\cdot, \mathbf{y}) &= 0 \qquad \text{on } \partial D \end{aligned}$$

for every $\mathbf{y} \in \Gamma$ with a parameter field $\kappa : D \times \Gamma \to \mathbb{R}$ and a right-hand side $f \in H^{-1}(D)$.

We propose an adaptive finite element solver for this task. Starting with a coarse uniform triangulation as an initial discretization, the following well-known steps are executed iteratively:

$$\text{Solve} \ \to \ \text{Estimate} \ \to \ \text{Mark} \ \to \ \text{Refine}. \tag{1.1}$$

To solve the PDE in each iteration, we derive a successive subspace correction algorithm (SSC), for which we refer to [44, 8]. The algorithm is based on a multilevel discretization of the domain.

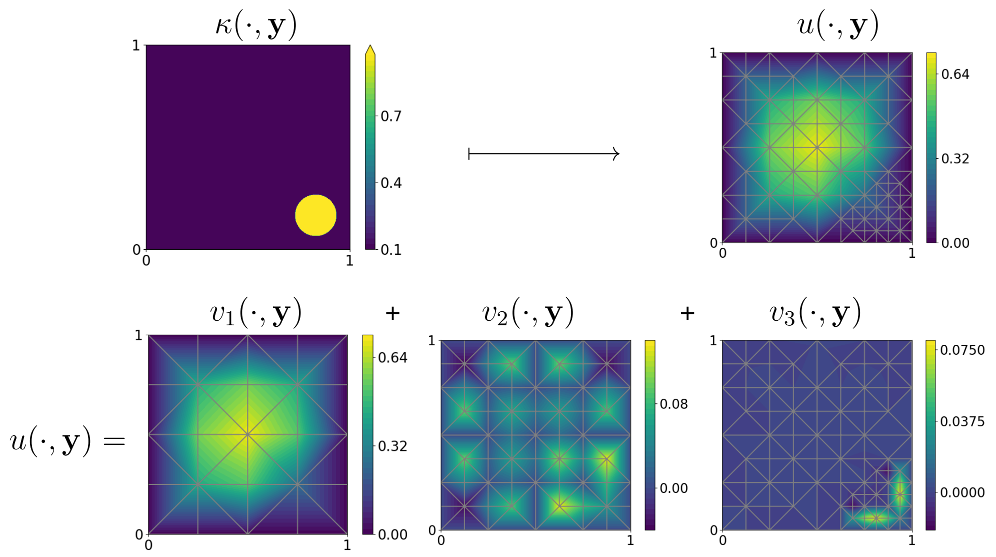


Figure 1.1: The first row depicts the parameter κ to solution u map for a realization of the parameter vector $\mathbf{y} \in \Gamma$ for (2.2). In the second row, the multigrid decomposition of the solution into a coarse grid function v_1 and finer grid corrections v_2, v_3 is visualized.

On coarser grids, the amplitude of the functions is larger. Given sufficient regularity, it decreases quickly on finer grids, where higher frequencies of the solution have to be represented. The principle is illustrated in Figure 1.1. To estimate the approximation error in the energy norm, a classical residual based finite element error estimator is implemented. The local error on each triangle T with side length h_T is bounded by

$$\eta_T^2 := h_T^2 \left\| f + \nabla \cdot (\kappa(\cdot,\mathbf{y})\nabla u) \right\|_{L_2(T)}^2 + h_T \left\| [\![\kappa(\cdot,\mathbf{y})\nabla u]\!] \right\|_{L_2(\partial T)}^2 .$$

Given an estimation of the local error contributions, triangles are selected for refinement e.g. with a Dörfler or a threshold marking strategy. The approximation space is enriched by adding nodal basis functions from a uniformly refined grid to the current basis. The structure of such a non-standard approximation space is illustrated in Figure 1.2. It can be seen that the constructed space consists of a selection of FE basis functions on different levels, which does not resemble a typical FE space.

We derive a suitable CNN architecture based on U-Nets for the problem and show that the architecture is expressive enough to accurately approximate each step in the adaptive solver in Section 2.5. The local refinements are incorporated in a the network by $0/1$-masks. Our main result is summarized as follows.

Theorem 1.1 (CNNs can approximate adaptive finite element solvers)**.** *Assume that κ is uniformly bounded from below and above. Let $\varepsilon > 0$ and $K, L \in \mathbb{N}$ be the number of iterations of the derived AFEM and the maximal refinements of each triangle, respectively. Consider a threshold marking strategy. Then there exists a CNN Ψ such that the number of parameters of the network is in $\mathcal{O}(LK \log(\varepsilon^{-1})/\log(c_L^{-1}))$ with $c_L := \frac{cL}{1+cL}, c > 0$. Moreover, for any $\mathbf{y} \in \Gamma$ the network maps finite element coefficients of the parameter field κ to coefficients of a finite element approximation of the solution u of the pPDE* (2.2) *such that*

$$\|u(\cdot,\mathbf{y}) - \mathcal{C}(\Psi(\boldsymbol{\kappa}_{\mathbf{y}}, \mathbf{f}))\|_{H^1(D)} \leq \|u(\cdot,\mathbf{y}) - \mathcal{C}(\mathrm{AFEM}(V_1, K))\|_{H^1(D)} + \varepsilon,$$

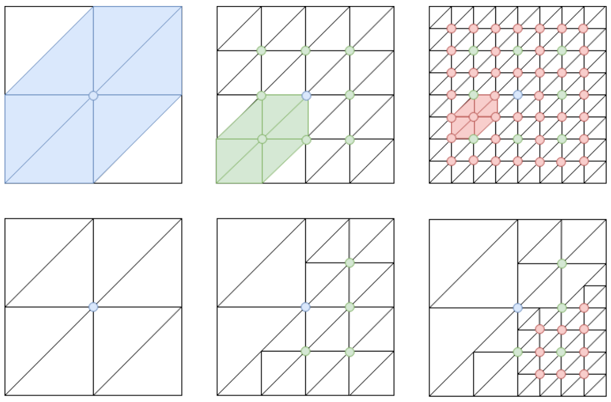


Figure 1.2: In the top row the support of the considered nodal basis functions on different levels is visualized. Uniformly refined meshes as used in [27] are shown in the top row, locally refined meshes as used in this work in the bottom row. The local refinement is realized by using a subset of the nodes in the uniformly refined meshes.

where $\mathcal{C}$ maps the finite element coefficients to the corresponding function.

The proof is based on the observation that U-Nets are able to approximate a successive subspace algorithm based on a multigrid decomposition. The required interpolation between grids can be achieved efficiently with strided and transpose strided convolutions. Furthermore, we show that the estimator can be approximated and refinement in each step of the adaptive algorithm. This is realized by $0/1$-masks multiplied by large images making it possible to work with sparse images for fine grids.

Since the solution representation is based on local contributions (corresponding to small regions of images with higher resolution), the proposed architecture can significantly improve computational efficiency by exploiting representation sparsity.

1.2 Main contributions

A multigrid solver, error estimator and refinement strategy are chosen, such that the corresponding AFEM can provably be approximated by an introduced CNN architecture. Complexity bounds for the approximation of the AFEM are shown for the architecture. In the course of the proof it is shown that CNNs can approximate a multigrid solver on locally refined grids. Moreover, CNNs can approximate the error estimator and the refinement can be incorporated by a novel error estimator based masking. This leads to an implicitly adaptive CNN tracking the error of individual outputs by error estimator prediction. The sparsity introduced by the masks on high resolution grids leads to a smaller number of operations on each grid and a higher accuracy for the same number of nonzero entries.

In contrast to the work carried out in [27], here the proof relies on actions on only parts of each image in the CNNs computations. Therefore, on each level, the boundary of the subsets has to be considered carefully and different index sets need to be considered in the analysis to always be able to represent all necessary information on each grid. In practice, the locality translates to manifold sparse convolutions, where kernels are only applied to nonzero entries of each image.

1.3 Structure of the paper

After the problem statement and a short finite element introduction, the individual steps of the AFEM and the algorithm itself are introduced in Section 2. The multigrid solver in the algorithm is explained in detail and its convergence is shown in Section 3. Section 4 is concerned with the the used data decomposition of continuous and discontinuous finite element functions to represent the solutions and estimators as images in the CNN. Furthermore, types of convolutions are discussed shortly. The main results, i.e. the expressivity theorems for the solver, the estimator and the whole AFEM algorithm can be found in Section 5. In Section 6 a numerical test is presented. Summary and outlook are given in Section 7.

2 Finite element discretization and notation

To generate data-efficient data, the finite element method (FEM) with an adaptive algorithm (coined AFEM) is used. This section is concerned with the introduction of this AFEM, which is steered by an a posteriori error estimator. This forms the basis of the subsequently derived neural network architecture. A more detailed introduction to finite elements can e.g. be found in [3].

2.1 Problem setting

We assume a regular conforming triangulation $\mathcal{T}$ of the (smoothly bounded) domain D, e.g. as depicted in Figure 1.1. Let $V_h = \operatorname{span}\{\varphi_j\}_{j=1}^{\dim V_h} \subset H_0^1(D)$ be a finite-dimensional subspace spanned by conforming first-order (Lagrange) basis functions (a FE function space). Any function $v_h \in V_h$ has a representation

$$v_h = \sum_{i=1}^{\dim V_h} \mathbf{v}_i \varphi_i,$$

where coefficient vectors with respect to the basis of V_h are written in bold face. Throughout this paper it is assumed that the parameter field fulfills a uniform boundedness assumption $\mathfrak{c} \leq \kappa(x, \mathbf{y}) \leq \mathfrak{C}$ for all $x \in D, \mathbf{y} \in \Gamma$ and some constants $\mathfrak{c}, \mathfrak{C} > 0$ independent of $\mathbf{y}$. We are concerned with finding a discrete solution $u_h \in V_h$ of the variational formulation for any $\mathbf{y} \in \Gamma$ such that for all test functions $w_h \in V_h$ it holds that

$$a_{\mathbf{y},h}(u_h, w_h) := \int_D \kappa_h(\cdot, \mathbf{y}) \langle \nabla u_h, \nabla w_h \rangle \mathrm{d}x = \int_D f w_h \mathrm{d}x =: \mathfrak{f}(w_h). \tag{2.1}$$

This is equivalent to determining $\mathbf{u} \in \mathbb{R}^{\dim V_h}$ by solving the algebraic system

$$A_{\mathbf{y}} \mathbf{u} = \mathbf{f} \tag{2.2}$$

with the right-hand side $\mathbf{f} := (\mathfrak{f}(\varphi_j))_{j=1}^{\dim V_h}$ and discretized operator

$$A_{\mathbf{y}} := (a_{\mathbf{y},h}(\varphi_i, \varphi_j))_{i,j=1}^{\dim V_h}. \tag{2.3}$$

We consider the following norms for $T \subset \mathbb{R}^d$ and $u : \mathbb{R}^d \to \mathbb{R}$

$$\|u\|^2_{L^2(T)} := \int_T u^2 \mathrm{d}x, \qquad \|u\|^2_{H^1(T)} := \int_T u^2 \mathrm{d}x + \int_T \langle \nabla u, \nabla u \rangle \,\mathrm{d}x, \qquad \|u\|^2_{a_{\mathbf{y},h}} := a_{\mathbf{y},h}(u,u).$$

For $\mathbf{u} \in \mathbb{R}^d$, we define the discrete norm

$$\|\mathbf{u}\|^2_{A_\mathbf{y}} := \mathbf{u}^\intercal A_\mathbf{y} \mathbf{u}.$$

Note that

$$\|\mathbf{u}\|^2_{A_\mathbf{y}} = \sum_{i,j=1}^{\dim V_h} \mathbf{u}_i \mathbf{u}_j a_{\mathbf{y},h}(\varphi_i, \varphi_j) = a_{\mathbf{y},h}\left(\sum_i^{\dim V_h} \mathbf{u}_i \varphi_i, \sum_{j=1}^{\dim V_h} \mathbf{u}_j \varphi_j \right) = a_{\mathbf{y},h}(u_h, u_h) = \|u_h\|_{a_{\mathbf{y},h}}.$$

Furthermore, we make use of the essential supremum norm L^∞ and the discrete supremum norm ℓ^∞.

2.2 Error estimation

We recollect the common residual based a posteriori error estimator for the Galerkin solution u_h of the (parametric) Darcy problem in V_h solving (2.1), cf. [42, 7] and [16] for the parametric setting.

Definition 2.1 (Jump & error estimator)**.** *The* jump *along the edge γ between the triangles $T^1, T^2 \in \mathcal{T}$ with $\nabla u_h^{(1)}$ and $\nabla u_h^{(2)}$ the gradients on the triangles, respectively, is defined by*

$$[\![\kappa_h(\cdot,\mathbf{y})\nabla u_h]\!] := \kappa_h(\cdot,\mathbf{y}) \left(\left\langle \nabla u_h^{(1)}, n_\gamma^{(1)} \right\rangle + \left\langle \nabla u_h^{(2)}, n_\gamma^{(2)} \right\rangle \right), \tag{2.4}$$

where $n_\gamma^{(1)}, n_\gamma^{(2)}$ are the normal vectors of γ pointing out of the triangles T^1, T^2, respectively. We define the local error contribution on each triangle T by

$$\eta_T^2 := h_T^2 \left\| f + \nabla \cdot (\kappa_h(\cdot,\mathbf{y})\nabla u_h) \right\|^2_{L_2(T)} + h_T \left\| [\![\kappa_h(\cdot,\mathbf{y})\nabla u_h]\!] \right\|^2_{L_2(\partial T)}. \tag{2.5}$$

We henceforth assume that the data error $\|\kappa - \kappa_h\|$ is negligible in the used norms. Then, the estimator is reliable and efficient, i.e. there exist constants $c_\mathbf{y}, C$ such that

$$\|u - u_h\|^2_{a_{\mathbf{y},h}} \le C \sum_{T\in\mathcal{T}} \eta_T^2 \qquad \text{and}$$
$$\eta_T \le c_\mathbf{y} \|u - u_h\|_{a_{\mathbf{y},h}} \quad \text{for any } T \in \mathcal{T}.$$

For the sake of a self-contained presentation, the derivation for the upper bound is recalled in Appendix A while a full analysis of this and other error estimators is carried out in standard references such as [42, 3].

2.3 Marking

For a complete adaptive finite element scheme as in (1.1) and as discussed in the next subsection, different marking strategies can be considered. A popular marking for which a fixed error convergence of the AFEM over the degrees of freedom can be shown is the Dörfler marking strategy [12, 36].

Definition 2.2 (Dörfler marking). *Let $\theta \in (0, 1)$. Define $\mathcal{M}$ such that*

$$\sum_{T\in\mathcal{M}} \eta_T^2 \geq \theta \sum_{T\in\mathcal{T}} \eta_T^2.$$

Alternatively, a maximum strategy can be considered [11]. When performing a marking decision for each element, access to the estimator for all other elements has to be available. Since the examined CNN architecture acts only locally on neighbouring elements, these marking strategies hence cannot be implemented and we resort to a threshold marking.

Definition 2.3 (Threshold marking). *For $k \in [L]$ let $\delta_h > 0$ be thresholds depending on the size of the triangles h, e.g. the maximal side length. Mark all elements $T \in \mathcal{T}$ with size h for refinement if $\eta_T^2 > \delta_h$.*

2.4 Mesh refinement

The next step of the AFEM consists of refining the current mesh in marked areas. In this work, in the Lth step of the AFEM the current space consists of the sum of subspaces of FE spaces on uniformly refined meshes with nodes $\mathcal{N}_k$ for $k \in [L]$ and corresponding basis functions $\{\varphi_i^k\}_{i\in\mathcal{N}_k}$. To refine a mesh element, all basis functions on a uniformly refined mesh (one level finer than the marked element) with overlapping support to the marked elements are included in V_h. Let $\mathcal{M} = \bigcup_{k=1}^L \mathcal{M}_k$ be the decomposition of the marked elements into sets of elements in the same uniformly refined mesh. Then the local mesh refinement is given by

$$V_h = V_h + \sum_{k=1}^{L} \text{span}\{\varphi_i^{k+1} : \exists T \in \mathcal{M}_k \text{ in level } k \text{ with } \operatorname{supp}\varphi_i^{k+1} \cap T \neq \emptyset\}.$$

2.5 Adaptive finite element method (AFEM)

Adaptive finite element methods are applied to find quasi optimal representations of PDE solutions by resolving local properties. Classical introductions can e.g. be found in [3, 42]. A version of AFEM used in the present work is depicted in Algorithm 1 and visualized in Figure 2.1. A multigrid solver introduced in Section 3 and the estimator in (2.5) are employed to approximate the solution of the Darcy problem (2.2) adaptively.

Starting with an initial FE function space V and an initial approximation $\mathbf{u} = 0$, the following steps are executed iteratively in the solver. The current solution corresponding to $\mathbf{u}$ is interpolated onto the current space V and $\mathbf{u}$ is set to the coefficients of the interpolated solutions. Then, the correction $\mathbf{v}$ to the best approximation in the current space is calculated by solving the system of linear equations

$$A_{\mathbf{y}}\mathbf{v} = \mathbf{f} - A_{\mathbf{y}}\mathbf{u}.$$

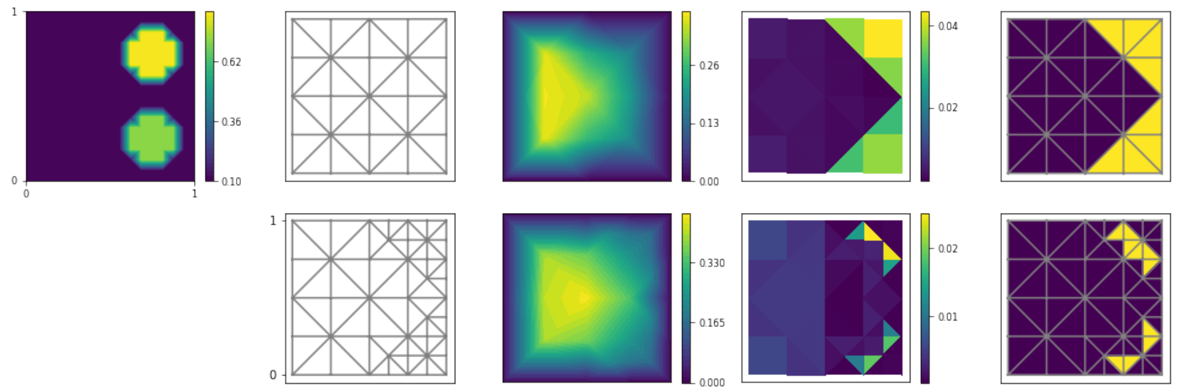


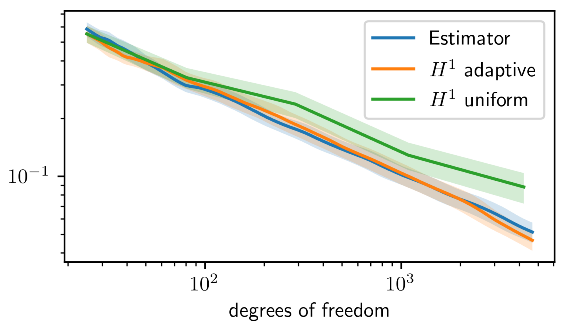
Figure 2.1: Two iterations of the adaptive finite element method on a unit square are depicted, where the first image on the left is a visualization of a possible parameter field $\kappa(\cdot, \mathbf{y})$. In the rest of the first row, the first mesh, solution, local error estimator and marker are depicted. The second row shows these steps for a locally refined mesh.

The solver is described in Section 3. The current solution is updated through $\mathbf{u} = \mathbf{u} + \mathbf{v}$. Local errors are estimated based on the a posteriori error estimator discussed in (2.5). Large errors are marked and the the space V is refined by including all basis elements of the next uniformly refined mesh in the current basis, which have an overlapping support with the marked regions.

To illustrate the practical performance, an example AFEM error convergence for the benchmark problem described in Section 6 can be compared to solutions on uniformly refined meshes in Figure 2.2. In addition to the relative errors in the H^1 and the L^2 norms, the error estimator is plotted. It can be observed, that the error estimator has the same decay as the true error in the H^1 norm as expected.

Algorithm 1: Adaptive finite element method $\mathrm{AFEM}(\kappa, f, V, K)$

```
1  Set $\mathbf{u} = 0$.
2  for $K$ iterations do
3      Interpolate the current solution in $V$ and set $\mathbf{u}$ to its coefficients.
4      Find $\mathbf{v}$ such that $A_{\mathbf{y}}\mathbf{v} = \mathbf{f} - A_{\mathbf{y}}\mathbf{u}$. (Section 3)
5      Update $\mathbf{u} = \mathbf{u} + \mathbf{v}$.
6      Estimate the error $\eta^2$. (Section 2.2)
7      Set $L$ to the number of levels of $V$.
8      Mark elements $\mathcal{M}_k$ on each level $k \in [L]$. (Section 2.3)
9      Refine the space
         $V = V + \sum_{k=1}^{L} \operatorname{span}\{\varphi_i^{k+1} : \exists T \in \mathcal{M}_k \text{ in level } k \text{ with } \operatorname{supp} \varphi_i^{k+1} \cap T \neq \emptyset\}$.
         (Section 2.4)
10 end
```



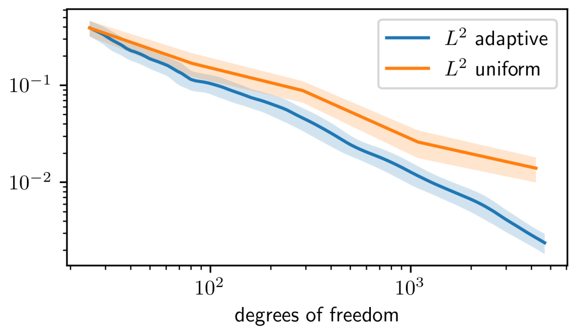


Figure 2.2: The two plots show the advantage of the AFEM in terms of degrees of freedom (FE coefficients) compared to solutions on uniformly refined meshes. Here, the mean and variance of the relative H^1 (left) and L^2 (right) errors of 100 samples of the problem described in Section 6 are plotted for the Dörfler marking with $\theta = 0.1$.

3 Solving on multiple grids

In this section, we derive a multigrid algorithm on the sum FE subspaces for uniformly refined grids to compute the corrections $\mathbf{v}$ in each step of Algorithm 1. It is closely related to classical FEM multigrid solvers, see e.g. [4, 26, 3]. Similarly, the convergence analysis is based on the more general framework of *successive subspace correction* (SSC) algorithms, see [44, 8]. It is exactly this algorithm that our CNN multilevel architecture is able to mimic. As shown in [27], an accurate and efficient NN representation of a multigrid solver exists for regular and uniform grids. In this work, the previous result is extended to locally refined grids and consequently to local multigrid corrections. While this should lead to a significant complexity improvement of the architecture and the training process, several technical difficulties are inevitably introduced by the locality of the subspace corrections.

We start our considerations with a number of levels $L \in \mathbb{N}$, which corresponds to the current maximal refinement in the step of the AFEM, where the solver is employed. Furthermore, a sequence of uniformly refined unit square grids with the set of nodes $(\mathcal{N}_\ell)_{\ell=1}^L$ indexed by[1] $i \in [n_\ell] \times [n_\ell] =: \mathcal{I}_U^\ell$ and set of triangles $(\mathcal{T}_\ell)_{\ell=1}^L$ is considered. The corresponding spaces spanned by the piecewise linear nodal hat functions $\varphi_i^\ell : \mathbb{R}^2 \to \mathbb{R}$ at nodes $i \in \mathcal{I}_U^\ell$ are denoted by $U^\ell :=$ span $\{\varphi_i^\ell :\ i \in \mathcal{I}_U^\ell\}$.

Since we intend to work on locally refined grids, on each level $\ell = 1, \dots, L$ only a subset of the index set $\mathcal{I}_V^\ell \subset \mathcal{I}_U^\ell$ and the corresponding triangles $\mathcal{T}_V^\ell$ are considered. These indices correspond to the nodal basis functions used in the local mesh refinement in Section 2.4 on each level. The discrete problem is then formulated with respect to $V_h := \sum_{\ell=1}^L V^\ell$ for $V^\ell :=$ span$\{\varphi_i^\ell : i \in \mathcal{I}_V^\ell\}$ with level $\ell \in [L]$ and $v_h \in V_h$. It can be represented by

$$v_h = \sum_{\ell=1}^{L} v^\ell = \sum_{\ell=1}^{L} \sum_{i \in \mathcal{I}_V^\ell} \mathbf{v}_i^\ell \varphi_i^\ell \tag{3.1}$$

with coefficients $\mathbf{v}^\ell \in \mathbb{R}^{\mathcal{I}_V^\ell}$ for $\ell \in [L]$ as visualized in Figure 3.1. We set the closure of $\mathcal{I}_V^\ell$ to $\overline{\mathcal{I}_V^\ell} := \mathcal{I}_V^k \cup \{i \in \mathcal{I}_U^\ell\ :\ \operatorname{supp}\varphi^\ell \cap \operatorname{supp} V^\ell \neq \emptyset\}$ to include all indices in $\mathcal{I}_U^\ell$, for which the

[1]we use the convention $[n] := \{1, \dots, n\}$

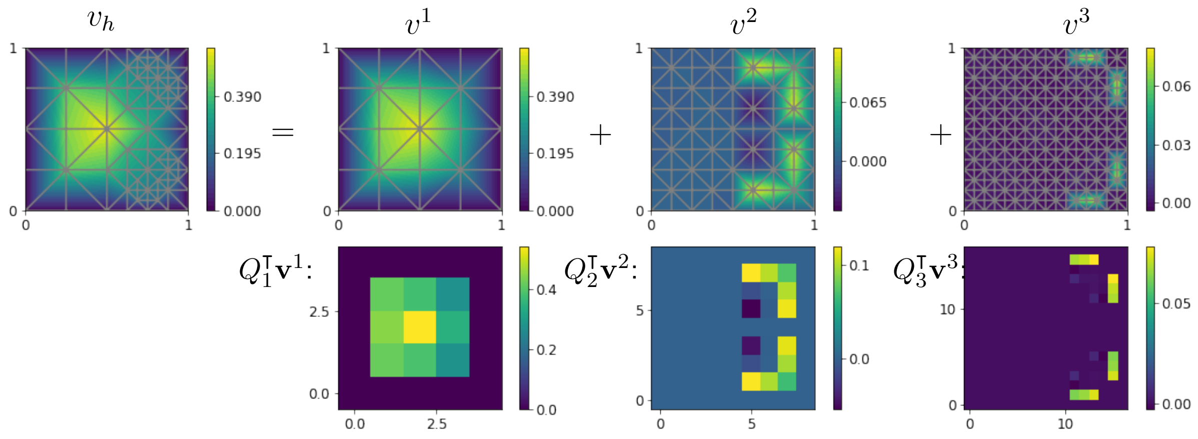


Figure 3.1: Depicted is the decomposition of a continuous function $v \in V_h$ into coarse grid parts and fine grid corrections on uniformly refined grids. Each function on a uniformly refined grid can be represented by an image, where one pixel corresponds to the value of one node. For local corrections the images are sparse.

corresponding functions have overlapping support with V^ℓ. Additionally, set the closure of V^ℓ to $\overline{V^\ell} :=$ span$\{\varphi_i^\ell : \operatorname{supp} \varphi_i^\ell \cap \operatorname{supp} V^\ell \neq \emptyset\}$. The closure is visualized in Figure 3.2. This set is of importance in the SSC algorithm with CNN, when projecting information of all subsets to one refinement level. It is a formal technicality that is due to the introduction of local contributions to the solution that were not present in [27].

3.1 Levelwise discretization

Since CNNs can only act on one discretization level (corresponding to one image size) at a time, we derive a SSC acting on the different levels separately. For this, define the ℓ^2–projection, restricting an element in the whole space to one subspace by

$$Q_k : \mathbb{R}^{\cup_{\ell\in[L]}\mathcal{I}_V^\ell} \to \mathbb{R}^{\mathcal{I}_V^k} \quad \text{with} \quad \mathbf{v} = (\mathbf{v}_{\ell,j})_{\ell\in[L],j\in\mathcal{I}_V^\ell} \mapsto (\mathbf{v}_{k,i})_{i\in\mathcal{I}_V^k} =: \mathbf{v}^k.$$

The transpose $Q_k^\intercal$ then trivially embeds an element of the subspace in the larger space. The above decomposition is visualized in Figure 3.1, depicting $Q_k^\intercal \mathbf{v}^k$ for each level $k = 1, \dots, L$. Furthermore, we set $\mathbf{v}^{<k} \in \mathbb{R}^{\sum_{\ell=1}^L \mathcal{I}_U^\ell}$ to be the contribution of $\mathbf{v}$ corresponding to levels smaller than k and $\mathbf{v}^{>k} \in \mathbb{R}^{\sum_{\ell=1}^L \mathcal{I}_U^\ell}$ to correspond to levels larger than k. Formally, this is defined by

$$\mathbf{v}^{<k} := \sum_{\ell=1}^{k-1} Q_k^\intercal \mathbf{v}^k \quad \text{and} \quad \mathbf{v}^{>k} := \sum_{\ell=k+1}^{L} Q_k^\intercal \mathbf{v}^k$$

with $\mathbf{v}^{<1} = \mathbf{v}^{>L} = 0 \in \mathbb{R}^{\cup_{\ell\in[L]}\mathcal{I}_V^\ell}$ such that

$$\mathbf{v} = \mathbf{v}^{<k} + Q_k^\intercal \mathbf{v}^k + \mathbf{v}^{>k}. \tag{3.2}$$

An SSC solving $A_\mathbf{y}\mathbf{u} = \mathbf{f}$ for some $\mathbf{f} \in \mathbb{R}^{\cup_{\ell\in[L]}\mathcal{I}_V^\ell}$ consists of *smoothing updates* on each level, carried out in a successive manner. One such update on level $k \in [L]$ has the form

$$\mathbf{u}^k \leftarrow \mathbf{u}^k + \omega_\mathbf{y}^k(\mathbf{f}^k - Q_k A_\mathbf{y} \mathbf{u}),$$

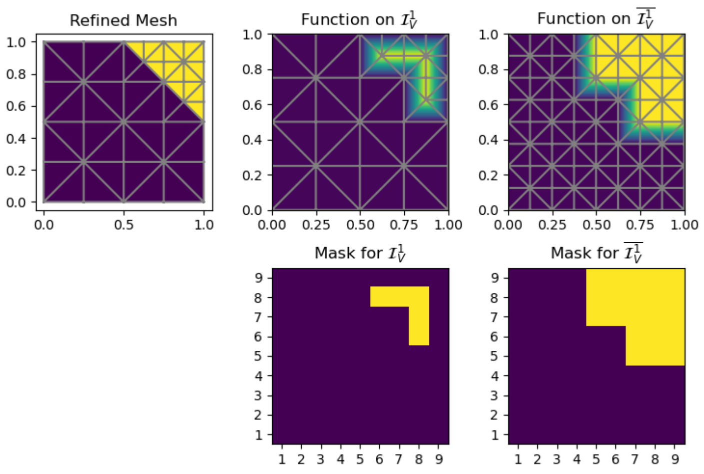


Figure 3.2: Refining a coarse mesh (compare first mesh in Figure 2.1) in the marked corner leads to the mesh depicted in the first row on the left-hand side. New degrees of freedom with indices in $\mathcal{I}_V^1$ stemming from this refinement (new nodes without the boundary nodes to incorporate the Dirichlet boundary condition) are depicted in the second image in the first row by a function, which is 1 on $\mathcal{I}_V^1$ and 0 otherwise. To visualize $\overline{\mathcal{I}_V^1}$, a function, which is 1 on indices in $\overline{\mathcal{I}_V^1}$ and 0 otherwise, is plotted in the last image in the first row. The corresponding masks on $\mathcal{I}_U^1$ are plotted in the second row.

where for each index $i \in \mathcal{I}_V^k$ on level k and each index on any level $(k_2, j) \in \cup_{\ell\in[L]}\{\ell\} \times \mathcal{I}_V^\ell$ the operator is set to $(Q_k A_{\mathbf{y}})_{i,(k_2,j)} := a_{\mathbf{y},h}(\varphi_i^k, \varphi_j^{k_2})$. Since the operator needs information of $\mathbf{u}$ on all levels, the operation has to be decomposed into contributions for each level individually. Therefore, in order to calculate $Q_k A_{\mathbf{y}} \mathbf{u}$, we consider the different contributions of the decomposition separately using the following prolongation and weighted restriction operations, which transfer discrete functions from one level to a consecutive level.

Definition 3.1 (Prolongation & weighted restriction)**.** *For $L \in \mathbb{N}$ and $k = 1, \dots, L-1$ define the* prolongation *$P_k : \mathbb{R}^{\overline{\mathcal{I}_V^k}} \to \mathbb{R}^{\overline{\mathcal{I}_V^{k+1}}}$ as the nodal interpolation of $\overline{V^k}$ onto $\overline{V^{k+1}}$. We call $P_k^\intercal$ the* weighted restriction *with $\varphi_i^k(x) = \sum_{j\in\overline{\mathcal{I}_V^{k+1}}} (P_k^\intercal)_{i,j}\varphi_j^{k+1}(x)$ for $x \in \operatorname{supp} V_{k+1}$.*

These operators can be used to connect different levels and calculate the application of the operator $A_{\mathbf{y}}$ to the whole vector $\mathbf{u}$ (with contributions from all levels) for a smoothing step on each level.

Theorem 3.1 (Levelwise calculation of $Q_k A_{\mathbf{y}} \mathbf{u}$)**.** *Let $\overline{A_{\mathbf{y}}^k}$ be defined as in* (2.3) *for indices in $\overline{\mathcal{I}_V^k} \times \overline{\mathcal{I}_V^k}$ and functions $\varphi_i^k \in \mathcal{I}_U^k$. Furthermore, set $\overline{\mathbf{u}_i^k} \in \mathbb{R}^{\overline{\mathcal{I}_V^k}}$ equal to $\mathbf{u}_i^k$ for $i \in \mathcal{I}_V^k$ and zero otherwise. To calculate $Q_k A_{\mathbf{y}} \mathbf{u}^{<k}$ and $Q_k A_{\mathbf{y}} \mathbf{u}^{>k}$, for $k \in [L]$ we define the auxiliary vectors*

$$\tilde{\mathbf{u}}^1 := 0, \quad \tilde{\mathbf{u}}^k := P_{k-1}\left(\tilde{\mathbf{u}}^{k-1} + \overline{\mathbf{u}^{k-1}}\right) \qquad \text{and} \tag{3.3}$$

$$\bar{\mathbf{u}}^L := 0, \quad \bar{\mathbf{u}}^k := P_k^\intercal\left(\bar{\mathbf{u}}^{k+1} + \overline{A_{\mathbf{y}}^{k+1}}^\intercal \mathbf{u}^{k+1}\right), \tag{3.4}$$

where $\tilde{\mathbf{u}}^k$ denotes the interpolation of $\mathbf{u}^{<k}$ into the current space and $\bar{\mathbf{u}}^k$ denotes the projection of $\mathbf{u}^{>k}$ onto the current space. This makes it possible to represent the multiplication with $A_{\mathbf{y}}$ on each level only using levelwise calculations, prolongations and weighted restrictions by

$$Q_k A_{\mathbf{y}} \mathbf{u} = \overline{A_{\mathbf{y}}^k}\left(\overline{\mathbf{u}^k} + \tilde{\mathbf{u}}^k\right) + \bar{\mathbf{u}}^k|_{\mathcal{I}_V^k}.$$

Proof. Since $A_\mathbf{y}$ is a linear operator, we considering the multiplication with the different parts of $\mathbf{u}$ separately. For $j \in \mathcal{I}_V^k$ it holds that

$$(Q_k A_\mathbf{y} Q_k^\intercal \mathbf{u}^k)_j = \sum_{i \in \mathcal{I}_V^k} \mathbf{u}_i^k \int \kappa_h \left\langle \nabla\varphi_i^k, \nabla\varphi_j^k \right\rangle \mathrm{d}x = \sum_{i \in \overline{\mathcal{I}_V^k}} \overline{\mathbf{u}_i^k} \int \kappa_h \left\langle \nabla\varphi_i^k, \nabla\varphi_j^k \right\rangle \mathrm{d}x = \left(\overline{A_\mathbf{y}^k}\,\overline{\mathbf{u}_i^k}\right)_j .$$

Lemma B.3 and Lemma B.4 yield

$$\begin{aligned} Q_k A_\mathbf{y} \mathbf{u}^{<k} &= \overline{A_\mathbf{y}^k}\tilde{\mathbf{u}}^k \quad \text{and} \\ Q_k A_\mathbf{y} \mathbf{u}^{>k} &= \bar{\mathbf{u}}^k|_{\mathcal{I}_V^k}, \end{aligned}$$

respectively. The claim follows with (3.2). □

With this, we can define an SSC using only levelwise actions and with restrictions and prolongations between two consecutive levels as depicted in the *Levelwise Local Multigrid Algorithm* (LLMG) in Algorithm 2. It consists of residual corrections and smoothing steps in each subspace $\mathbb{R}^{\mathcal{I}_V^k}$ separately, starting with the finest level L, successively including coarser levels, and subsequently updating finer levels until each level has been updated twice. After each update, the auxiliary variables $\bar{\mathbf{u}}, \tilde{\mathbf{u}}$ are updated. Since the definition in (3.4) of $\bar{\mathbf{u}}^k$ contains information only of $\mathbf{u}^\ell$ for $\ell > k$, i.e. information of finer levels, and $\tilde{\mathbf{u}}^k$ in (3.3) only depends on $\mathbf{u}^\ell$ for $\ell < k$, i.e. information of coarser levels, the update of one $\mathbf{u}^k$ leads to a change of $\bar{\mathbf{u}}^\ell$ for $\ell < k$ and a change of $\tilde{\mathbf{u}}^\ell$ for $\ell > k$. Therefore, when smoothing on a coarser level in the subsequent step, $\bar{\mathbf{u}}^{k-1}$ needs to be updated. When smoothing on a finer level in the subsequent step, the update has to be done for $\tilde{\mathbf{u}}^{k+1}$.

Algorithm 2: Levelwise Local Multigrid Algorithm $\mathrm{LLMG}(\mathbf{u}, \mathbf{f}, \mathbf{y})$

```
1  Calculate $\bar{\mathbf{u}}, \tilde{\mathbf{u}}$ as in (3.4), (3.3)    ▷ calculate auxiliary vectors
2  for $k = L, \dots 1$ do
3      $\mathbf{u}^k \leftarrow \mathbf{u}^k + \omega_\mathbf{y}^k(\mathbf{f}^k - [\overline{A_\mathbf{y}^k}(\mathbf{u}^k + \tilde{\mathbf{u}}^k) + \bar{\mathbf{u}}^k|_{\mathcal{I}_V^k}])$    ▷ smoothing on one level
4      if $k > 1$ then
5          $\bar{\mathbf{u}}^{k-1} \leftarrow P_{k-1}^\intercal(\bar{\mathbf{u}}^k + \overline{A_\mathbf{y}^k}^\intercal \mathbf{u}^k)$    ▷ update auxiliary vector $\bar{\mathbf{u}}$ fine to coarse
6      end
7  end
8  for $k = 1, \dots, L$ do
9      $\mathbf{u}^k \leftarrow \mathbf{u}^k + \omega_\mathbf{y}^k(\mathbf{f}^k - [\overline{A_\mathbf{y}^k}(\mathbf{u}^k + \tilde{\mathbf{u}}^k) + \bar{\mathbf{u}}^k|_{\mathcal{I}_V^k}])$    ▷ smoothing on one level
10     if $k < L$ then
11         $\tilde{\mathbf{u}}^{k+1} \leftarrow P_k\left(\tilde{\mathbf{u}}^k + \overline{\mathbf{u}^k}\right)$    ▷ update auxiliary vector $\tilde{\mathbf{u}}$ coarse to fine
12     end
13 end
```

With Theorem 3.1, the LLMG is a standard SSC algorithm (see Algorithm 3) and convergence can be derived from known results.

Theorem 3.2 (Convergence of the LLMG)**.** *Assume that there exist constants* $\mathfrak{c}, \mathfrak{C} > 0$ *such that* $\mathfrak{c} \le \min_{\mathbf{y}\in\Gamma} \lambda_{\min}(A_\mathbf{y})$ *and* $\mathfrak{C} \ge \max_{\mathbf{y}\in\Gamma} \lambda_{\max}(A_\mathbf{y})$. *Let* $\mathbf{u}$ *be the solution of* $A_\mathbf{y}\mathbf{u} = \mathbf{f}$ *and* $0 <$

$\omega_{\mathbf{y}}^k = \omega \le C^{-1}$. There exists a constant $c > 0$ such that for $c_L := \frac{cL}{1+cL}$, $\varepsilon > 0$ and $m \in \mathbb{N}$ with $m \ge \log(\varepsilon^{-1})/\log(c_L^{-1})$ it holds true that

$$\|\mathbf{u} - \mathrm{LLMG}^m(0, \mathbf{f}, \mathbf{y})\|_{A_{\mathbf{y}}} \le \varepsilon \|\mathbf{u}\|_{A_{\mathbf{y}}} ,$$

where LLMG^m denotes the application of the algorithm m times.

Proof. For a fixed $\mathbf{y} \in \Gamma$ the LLMG is equivalent to the local multigrid algorithm (LMG Algorithm 4) with Theorem 3.1 and therefore has the same convergence rate. We use the XZ-identity in Lemma B.1 shown in [8, Theorem 4] and a result similar to the Richardson smoothing contraction shown in [4, Lemma 4.3] in Lemma B.2 to deduce the convergence of the LMG in the appendix (see Theorem B.1 and Appendix B.2), showing that there exists a constant $C > 0$ such that for $c_L = \frac{CL}{1+CL}$ it holds

$$\left\|\mathbf{u} - \mathrm{LLMG}^m(\mathbf{u}^0, \mathbf{f}, \mathbf{y})\right\|_{A_{\mathbf{y}}} \le c_L^m \left\|\mathbf{u} - \mathbf{u}^0\right\|_{A_{\mathbf{y}}} .$$

Therefore, initializing with $\mathbf{u}^0 = 0$ and choosing $m \ge \log(\varepsilon^{-1})/\log(c_L^{-1})$ confirms the claim. $\square$

Remark 3.1. *Note that $\kappa(\cdot, \mathbf{y}) > c$ for some $c > 0$ and all $\mathbf{y} \in \Gamma$ implies with the Poincaré inequality that $\lambda_{\min}(A_{\mathbf{y}}) > C$ for some $C > 0$ and all $\mathbf{y} \in \Gamma$. Furthermore, $\kappa(\cdot, \mathbf{y}) < c$ for some $c > 0$ and all $\mathbf{y} \in \Gamma$ implies that $\lambda_{\max}(A_{\mathbf{y}}) < C$ for some $C > 0$ and all $\mathbf{y} \in \Gamma$.*

4 Convolutional neural networks (CNN) for finite element discretizations

CNNs are a specific neural network architecture tailored to tasks involving image data such as image classification and segmentation. Inspired by [20], they were first implemented with a backpropagation algorithm in [31] for image recognition. Applying the action of a CNN to an image involves the application of local kernels to a hierarchy of scaled representations of the input. This locality makes the architecture particularly suitable with partial differential equations, where local properties and interactions have to be resolved to obtain highly accurate representations. To incorporate interactions on a larger scale with respect to the image domain, compression and decompression of the input images can be implemented with CNNs through strided and transpose strided convolutions, leading the popular CNN architecture U-Nets [37]. This architecture is heavily exploited in this work.

4.1 Data decomposition

In the analysis of the implemented CNN architecture, images with different resolutions are used as the representation of the solutions of the parametric PDE. This is possible since FEM discretizations of functions determined by coefficient vectors are used on different grid levels, similar to the decomposition in [27]. Additionally, the discontinuous error estimator[2] defined on the triangles of the considered meshes are represented with images of different scales.

[2]Note that the solution is assumed as a conforming P1 function and the estimator is a DG0 function, i.e. defined by a scalar value per mesh element.

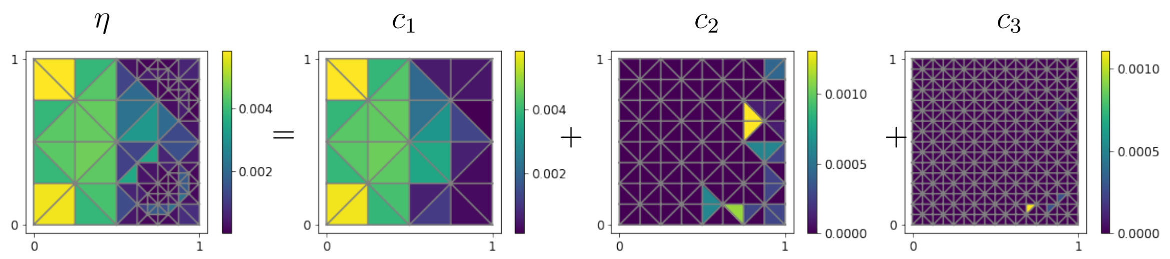


Figure 4.1: A piecewise constant discontinuous functions $\eta = \sum_{T\in\mathcal{T}} \eta_T \chi_T$ can be decomposed into a coarse grid piecewise constant function and a fine grid piecewise constant corrections.

Continuous functions Any $v_h \in V_h$ can be decomposed into its components on $v^\ell \in V^\ell$ by $v_h = \sum_{\ell=1}^{L} v^\ell$ as described in Section 3. The functions v^ℓ on each level can be represented by *coefficient images* on the whole uniformly refined grids. If $\mathcal{I}_V^k$ is a small subset of $\mathcal{I}_U^k$, i.e. in case of very local refinements in the AFEM, these images are sparse. The complete decomposition of $v_h \in V_h$ as depicted in Figure 3.1 reads

$$v_h = \sum_{\ell=1}^{L} v^\ell = \sum_{\ell=1}^{L} \sum_{i\in\mathcal{I}_V^\ell} \mathbf{v}_i^\ell \varphi_i^\ell .$$

Definition 4.1 (Coefficient images)**.** *For* $\mathbf{w} \in \mathbb{R}^{\mathcal{I}_V^k}$ *the coefficient image* $\mathbf{w}_{img} \in \mathbb{R}^{\mathcal{I}_U^k}$ *is defined for* $i \in \mathcal{I}_U^k$ *by*

$$(\mathbf{w}_{img})_i := \begin{cases} \mathbf{w}_i, & \text{if } i \in \mathcal{I}_V^k \\ 0, & \text{otherwise} \end{cases},$$

where the indices are two dimensional image indices $i = (i_1, i_2)$*.*

In the same manner as the continuous functions, the *piecewise constant functions* can be decomposed into corrections on uniformly refined meshes as depicted in Figure 4.1, namely

$$\eta = \sum_{\ell=1}^{L} \eta^\ell = \sum_{\ell=1}^{L} \sum_{i=1}^{n} \sum_{j=1}^{m^\ell} \boldsymbol{\eta}_{(i,j)}^\ell \tilde{\varphi}_{i,j}^\ell , \tag{4.1}$$

where n is the number of images needed for the representation. It depends on the structure of the mesh ($n = 1$ in the continuous case due to nodal representation) and $m^\ell, \ell = 1, \dots, L$ the number of pixels of each image. Here, $\tilde{\varphi}_{(i,j)}^\ell$ denotes the characteristic function on the triangle $(i, j) \in [n] \times [m^\ell]$ on discretization level $\ell \in [L]$ for the indexation of the triangles according to (4.1). In Figure 4.2 each (continuous) correction is represented with $n = 8$ *sparse images*. Piecewise constant (discontinuous) functions on meshes as depicted in Figure 1.2 can be represented with $n = 2$ images, e.g. by one image containing the values of the upper triangle in the upper right square of each node and the other image containing the lower triangle in the same square as illustrated in Figure 5.1.

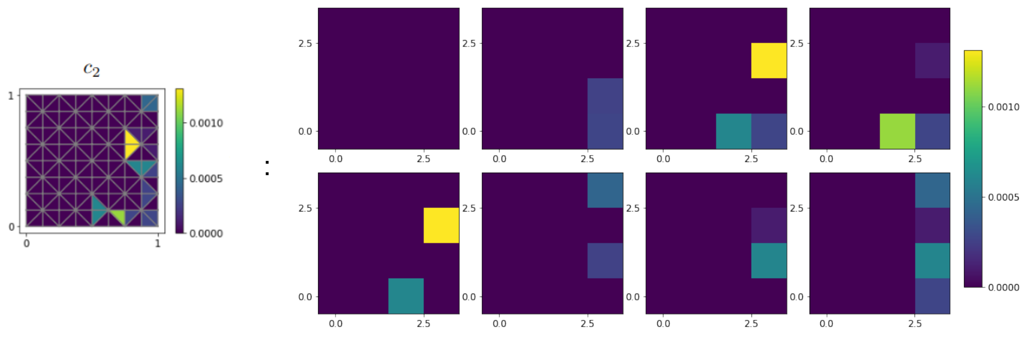


Figure 4.2: Each piecewise constant function for the meshes depicted in Figure 4.1 can be represented with 8 images. Every other node in each direction is surrounded by 8 triangles. Each image corresponds to one of the triangles for every such node.

4.2 Submanifold sparse CNN

Different types of convolutions are considered in this work. The *vanilla* convolution $*$ sweeps a kernel over the input, the 2-*strided* convolution $*^{2\text{s}}$ applies the kernel on every other pixel of the input images, approximately halving the size of the input images. Moreover, the 2-*transpose strided* convolution $*^{2\text{st}}$ sweeps the kernel over a dilated image with added zeros between every two pixels, doubling the input image size and the *submanifold sparse* convolution $*^{\text{sp}}$ as used in [24, 23] applies the kernel only to nonzero pixels of the input image and sets the remaining entries to zero. The different convolutions are visualized in Figure C.1, see also [13, 14].

Every step of the successive subspace correction algorithm Algorithm 2 and the error estimator (2.5) can be represented with a CNN by incorporating these different convolutions. When additionally including a marking function as culling mask, the whole adaptive scheme Algorithm 1 can be approximated by the derived CNNs on sparse images.

5 Expressivity results

For the analysis in this work, the used activation function has to satisfy the following assumption.

Assumption 5.1 (Activation function)**.** *Let $\sigma \in L^\infty_{\mathit{loc}}$ such that there exists $x_0 \in \mathbb{R}$, where σ is three times continuously differentiable in a neighborhood and $\sigma''(x_0) \neq 0$.*

These properties are fulfilled for a number of classical activation functions such as softplus, sigmoids and the exponential linear unit. In the following subsections, the individual parts of the AFEM (Algorithm 1) are approximated individually. The estimations are then collected for the overall convergence result. To illustrate our constructions, the meshes depicted in Figure 1.2 are considered.

5.1 NN approximation of the multigrid solver

To approximate the solution on a fixed grid in each step of the AFEM, the "Levelwise Local Multigrid Algorithm"LLMG (Algorithm 2) is employed. Its main ingredient is the smoothing on each (locally refined) subspace. For one smoothing step, the crucial part is the approximation of the action of the parametric operator $A_{\mathbf{y}}\mathbf{u}$ with a CNN. The analysis is similar to [27, Theorem 6]. However, the local corrections impose several technical additions, for which some auxiliary vectors defined in the algorithm are introduced. We are then able to show the following complexity bound.

Theorem 5.2. *Assume that there exist constants $\mathfrak{c}, \mathfrak{C} > 0$ such that $\mathfrak{c} \leq \min_{\mathbf{y}\in\Gamma} \lambda_{\min}(A_{\mathbf{y}})$ and $\mathfrak{C} \geq \max_{\mathbf{y}\in\Gamma} \lambda_{\max}(A_{\mathbf{y}})$. There exists a positive constant $C > 0$ such that for every $\varepsilon, M > 0$ there exists a CNN $\Psi : \mathbb{R}^{2\times\mathcal{I}_U^L} \to \mathbb{R}^{\times_{k=1}^L \mathcal{I}_V^k}$ such that*

1. $\left\|\Psi(\boldsymbol{\kappa}_{\mathbf{y}\,img}, \mathbf{f}_{img}) - \mathrm{LLMG}^m(0, \mathbf{f}, \mathbf{y})\right\|_{A_{\mathbf{y}}} \leq \varepsilon$ *for all $\kappa(\cdot, \mathbf{y}), f \in U^L$*
2. *number of weights bounded by $M(\Psi) \leq CLm$.*

The proof can be found in Appendix D.1. Combining this result with Theorem 3.2 leads to the following corollary, stating that the solution of the Darcy problem (2.2) on a an adaptively refined mesh can be approximated arbitrarily well by a CNN with a prescribed bound for the number of parameters.

Corollary 5.1. *Let $\mathbf{u}$ be the solution of $A_{\mathbf{y}}\mathbf{u} = \mathbf{f}$. Choose $m \geq \log(\varepsilon^{-1})/\log(c_L^{-1})$ with chosen as in Theorem 3.2. Then there exists a constant $C > 0$ such that for any $\varepsilon > 0$ there exists a CNN $\Psi : \mathbb{R}^{2\times\mathcal{I}_U^L} \to \mathbb{R}^{\times_{k=1}^L \mathcal{I}_V^k}$ with the number of parameters bounded by $M(\Psi) \leq CL\log(\varepsilon^{-1})/\log(c_L^{-1})$ such that*

$$\begin{aligned}\left\|\Psi(\boldsymbol{\kappa}_{\mathbf{y}\,img}, \mathbf{f}_{img}) - \mathbf{u}\right\|_{A_{\mathbf{y}}} &\leq \left\|\Psi(\boldsymbol{\kappa}_{\mathbf{y}\,img}, \mathbf{f}_{img}) - \mathrm{LLMG}^m(0, \mathbf{f}, \mathbf{y})\right\|_{A_{\mathbf{y}}} \\ &\quad + \|\mathrm{LLMG}^m(0, \mathbf{f}, \mathbf{y}) - \mathbf{u}\|_{A_{\mathbf{y}}} \\ &\leq \varepsilon(1 + \|\mathbf{u}\|_{A_{\mathbf{y}}}).\end{aligned}$$

5.2 Estimator approximation

A central novelty of this work is the CNN representation of the a posteriori error estimator η subject to u_h as used in the AFEM, see Section 2.2. For the analysis of the approximation, the two parts of the estimator (the jump term and strong residual) are considered separately. The analysis is carried out for a reference triangle in $\mathcal{T}_{V,k}$ from a triangulation V^k.

Definition 5.1 (Strong residual & jump images)**.** *Let $T^1_{k,i}$ and $T^2_{k,i}$ be the triangles in the top right quadrant of node $i \in \mathcal{I}_U^k$ as depicted in Figure 5.1 and let $k \in [L]$. Define $\mathrm{r}^2_{k,T^q}, \mathrm{j}^2_{k,T^q} \in \mathbb{R}^{\mathcal{I}_U^k}$ for $q = 1, 2$ by*

$$\begin{aligned}(\mathrm{r}^2_{k,T^q})_i &:= h^2_{T^q_{k,i}} \|f + \nabla\cdot(\kappa_h(\cdot,\mathbf{y})\nabla u_h)\|^2_{L_2(T^q_{k,i})} && \text{as the strong residual image and}\\ (\mathrm{j}^2_{k,T^q})_i &:= h_{T^q_{k,i}} \|[\![\kappa_h(\cdot,\mathbf{y})\nabla u_h]\!]\|^2_{L_2(\partial T^q_{k,i})} && \text{as the jump image.}\end{aligned}$$

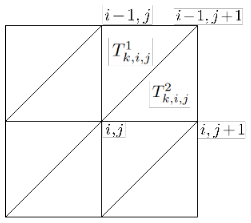


Figure 5.1: On each level $k \in [L]$ at each node $i \in \mathcal{I}_U^k$ we define the mesh elements $T^1_{k,i}$ and $T^2_{k,i}$ as the triangles in the upper right quadrant.

Then the a posteriori error estimator in a triangle $T^q_{k,i} \in \mathcal{T}_{V,k}$ can be written as

$$\eta^2_{T^q_{k,i}} = \left(\mathrm{r}^2_{k,T^q}\right)_i + \left(\mathrm{j}^2_{k,T^q}\right)_i .$$

Definition 5.2 (Uniform prolongation and weighted restriction)**.** *Extending the definition of the prolongation and weighted restriction from the subspaces V_k to U_k, the* uniform prolongation *is defined as $P_{U,k} : \mathbb{R}^{\mathcal{I}_U^k} \to \mathbb{R}^{\mathcal{I}_U^{k+1}}$ such that $\varphi_i^k = \sum_{j \in \mathcal{I}_U^{k+1}} (P_{U,k})_{j,i} \varphi_j^{k+1}$ for all $i \in \mathcal{I}_U^k$. The transpose $P_{U,k}^\intercal$ is called* uniform weighted restriction*.*

Note that this is the prolongation as defined in [27, Defnition 2].

Theorem 5.3 (Estimator approximation)**.** *Let $\eta_k^2, M^k \in \mathbb{R}^{2 \times \mathcal{I}_U^k}$ be defined for $k = 1, \ldots, L$, $q \in \{1,2\}$ and $i \in \mathcal{I}_U^k$ by $(\eta_k^2)[q]_i := \eta^2_{T^q_{k,i}}$. Moreover, let $M^k[q]_i := 1$, if $T^q_{k,i} \in \mathcal{T}_V^k$ and zero otherwise. There exists a constant $C > 0$ such that for every $\varepsilon, M > 0$ there exists a CNN $\Psi : \mathbb{R}^{\times_{\ell=1}^L 3 \times \mathcal{I}_U^\ell} \to \mathbb{R}^{\times_{\ell=1}^L \mathcal{I}_U^\ell}$ such that*

1 $\left\| M^\ell \odot \Psi(\bigtimes_{k=1}^L \mathbf{u}^k_{img} \times \boldsymbol{\kappa}^k_{\mathbf{y}\,img} \times \mathbf{f}^k_{img})[\ell] - \eta^2_\ell \right\|_{\ell^\infty} \leq \varepsilon$ holds for all $\ell = 1, \ldots, L$ and

2 the number of parameters is bounded by $M(\Psi) \leq CL$.

Proof. For the finest level L and $q = 1,2$ let the estimator images $\mathrm{r}^2_{L,T^q}, \mathrm{j}^2_{L,T^q} \in \mathbb{R}^{\mathcal{I}_U^L}$ be defined as in Definition 5.1. To represent the solution on the finest level let $\mathbf{u}^{P_1}_{\text{img}} := \mathbf{u}^1_{\text{img}}$ and $\mathbf{u}^{P_k}_{\text{img}} := P_{U,k} \mathbf{u}^{P_{k-1}}_{\text{img}} + \mathbf{u}^k_{\text{img}}$ according to Definition 3.1 for $k = 1, \ldots, L$. Then, with [27, Remark 19] $\mathbf{u}^{P_k}_{\text{img}}$ can be calculated with a CNN and $\mathbf{u}^{P_L}$ contains the coefficients of the nodal interpolation of the function $u_h \in V$ defined by $\bigtimes_{k=1}^L \mathbf{u}^k$ in U^L.

Now the estimator can be approximated on every level by approximating the residual and jump images in the fines level and combining them correctly. In Theorem D.4 we show that there exists a CNN architecture such that for every $\varepsilon, M > 0$ and $q = 1,2$ there exists a CNN Ψ with

$$\left\| \Psi(\mathbf{u}^{P_L}, \mathbf{f}^L, \boldsymbol{\kappa}^L_{\mathbf{y}})[q] - (\mathrm{r}^2_{L,T^q}, \mathrm{j}^2_{L,T^q}) \right\|_\infty \leq \varepsilon.$$

Observe that for triangles $T_k \in \mathcal{T}_k$ it holds that $h_{T_k} = h_0/2^k = 2h_0/2^{k+1} = 2h_{T_{k+1}}$. Furthermore, each triangle on level k is equal to the union of four triangles on level $k+1$

$$T^q_{k,i} = \bigcup_{\tilde q \in \{1,2\}} \bigcup_{\substack{j \in \mathcal{I}_U^{k+1} \text{s.t.} \\ T^{\tilde q}_{k+1,j} \subset T^q_{k,i}}} T^{\tilde q}_{k+1,j}.$$

This yields for $i \in \mathcal{I}_U^k$ and for triangles as in Figure 5.1

$$\begin{aligned}(\mathrm{r}^2_{k,T^q})_i &= h^2_{T^q} \left\| f + \nabla \cdot (\kappa_h(\cdot, \mathbf{y}) \nabla u_h) \right\|^2_{L_2(T^q_{k,i})} \\ &= 2^2 \sum_{\tilde{q} \in \{1,2\}} \sum_{\substack{j \in \mathcal{I}_U^{k+1} \text{s.t.} \\ T^{\tilde{q}}_{k+1,j} \subset T^q_{k,i}}} h^2_{T_{k+1}} \left\| f + \nabla \cdot (\kappa_h(\cdot, \mathbf{y}) \nabla u_h) \right\|^2_{L_2(T^{\tilde{q}}_{k+1,j})} \\ &= 4 \sum_{\tilde{q} \in \{1,2\}} \sum_{\substack{j \in \mathcal{I}_U^{k+1} \text{s.t.} \\ T^{\tilde{q}}_{k+1,j} \subset T^q_{k,i}}} (\mathrm{r}^2_{k+1,T^{\tilde{q}}})_j.\end{aligned}$$

This can be implemented with one CNN layer with a sparse kernel and stride 2 for each level. Since the jump term is zero on edges in the fine discretization, which have not been used to solve for u_h, jumps over edges of some triangle $T^{\tilde{q}}_{k+1,j}$ on level $k+1$ inside a triangle $T^q_{k,i}$ on level k can be added up to yield the jumps only over edges on the coarser level. This yields

$$\begin{aligned}(\mathrm{j}^2_{k,T^q})_i = h_{T_k} \left\| [\![\kappa_h(\cdot, \mathbf{y}) \nabla u_h]\!] \right\|^2_{L^2(\partial T^q_{k,i})} &= 2 \sum_{\tilde{q} \in \{1,2\}} \sum_{\substack{j \in \mathcal{I}_U^{k+1} \text{s.t.} \\ T^{\tilde{q}}_{k+1,j} \subset T^q_{k,i}}} h_{T_{k+1}} \left\| [\![\kappa_h(\cdot, \mathbf{y}) \nabla u_h]\!] \right\|^2_{L^2(\partial T^{\tilde{q}}_{k+1,j})} \\ &= 2 \sum_{\tilde{q} \in \{1,2\}} \sum_{\substack{j \in \mathcal{I}_U^{k+1} \text{s.t.} \\ T^{\tilde{q}}_{k+1,j} \subset T^q_{k,i}}} (\mathrm{j}^2_{k+1,T^{\tilde{q}}})_j.\end{aligned}$$

This can also be realized by one CNN layer with a sparse kernel and a stride of 2 for each level. Since adding the strong residual image and jump image yields the error estimator for triangles $T^q_{k,i} \in \mathcal{T}_{V,k}$, multiplying with a mask setting all other output entries to zero yields the claim. □

5.3 AFEM approximation

Combining the approximation of the multigrid solver and the error estimator with a marker based on the estimator leads to an approximation of the whole AFEM algorithm. For the formulation of our main theorem, the following coefficient-to-function map is needed.

Definition 5.3. *Let $\mathcal{C} : \mathbb{R}^{\mathcal{I}_U^1 \times \cdots \times \mathcal{I}_U^L} \to H_0^1$ be the function that maps the finite element coefficients to the corresponding function in H_0^1 by*

$$\mathcal{C}(\mathbf{u}) := \sum_{k=1}^{L} \sum_{i \in \mathcal{I}_U^k} \mathbf{u}_i^k \varphi_i^k.$$

Our main result then gives an upper bound on the number of parameters needed by the constructed networks architecture to approximate the solution of the Darcy problem as well as the result of the AFEM algorithm. In summary, the derived bound depends linearly on the number of refinement levels as well as linearly on the number of steps of the AFEM and logarithmically on the inverse of the desired accuracy.

Theorem 5.4 (Approximate AFEM)**.** *Assume there exist $\mathfrak{c}, \mathfrak{C} > 0$ such that $\mathfrak{c} \leq \kappa(x, \mathbf{y}) \leq \mathfrak{C}$ for all $x \in D$ and $\mathbf{y} \in \Gamma$. Let $\varepsilon > 0$ and $K, L \in \mathbb{N}$ be the number of AFEM iterations and maximal refinements of each triangle, respectively. Consider a threshold marking strategy. Then there exists a CNN Ψ such that $M(\Psi) \lesssim LK \log(\varepsilon^{-1})/\log(c_L^{-1})$ with $c_L := \frac{cL}{1+cL}, c > 0$ and for any $\mathbf{y} \in \Gamma$*

$$\|u(\cdot, \mathbf{y}) - \mathcal{C}(\Psi(\boldsymbol{\kappa}_{\mathbf{y}}, \mathbf{f}))\|_{H^1(D)} \leq \|u(\cdot, \mathbf{y}) - \mathcal{C}(\mathrm{AFEM}(V_1, K))\|_{H^1(D)} + \varepsilon.$$

The proof of this theorem can be found in Appendix D.3. The complete architecture is depicted in Figure 5.2. Here, the solver in each step is approximated by U-Nets encoded by green arrows outputting approximations of the solution in a multigrid discretization (green boxes) as described in Theorem 5.2. The estimator (orange) is approximated based on the approximate solutions with networks as constructed in Theorem 5.3 and the refinement masks (purple boxes) are derived from the estimator and used in the next solver. Here, the space was refined uniformly in the first step leading to masks, which are 1 everywhere. In the second step the space was refined locally leading to a $0/1$-mask on the finest level. Adding all continuous functions corresponding to the images in the green boxes as in (3.1) leads to the full approximate solution.

Remark 5.1. *Global marking strategies such as Dörfler marking cannot be implemented directly in a CNN due to its local action in a neighbourhood. However, such a marking can in principle be implemented outside the CNN based on the estimator prediction of the CNN. As an alternative, the marking strategy could be learned by a separate NN based on the locally adapted training data. This marking NN could then be combined with the proposed CNN. Recent research in this direction can e.g. be found in [22, 40, 45, 19, 18].*

6 Numerical experiments

This section is concerned with the practical performance of the proposed architecture. Here, we present preliminary proof of concept results. The architecture should be tested for more steps of the adaptive solver and different expansions of the parameter field. The numerical tests are implemented for a parametric stationary diffusion problem with parametric coefficient defined by

$$\kappa(\cdot, \mathbf{y}) := 0.1 + \mathbf{y}_1 \chi_{D_1} + \mathbf{y}_2 \chi_{D_2}.$$

For this “cookie problem”, we assume that $\mathbf{y} \sim U([0,1]^2)$, D_1, D_2 are disks of radius $r = 0.15$ and centers at $(0.75, 0.25)$ and $(0.75, 0.75)$, respectively. The architecture is implemented for $K = 3$ steps of the AFEM. For each step a solver was approximated with $3, 2, 1$ U-Nets. The overall number of trainable parameters is $2\,441\,516$. In Figure 6.1, the final network outputs are compared to a reference solution (obtained by solving on twice uniformly refined meshes) as well as error estimators and markers from the training data. It can be observed that solution and estimator are approximated well with local errors magnitudes smaller than the actual values. Note that the marker based on the error estimator as derived in the network differs from the marker used to generate the data. This inaccuracy in the prediction of the marked elements leads to nonzero elements in the network output in areas, which ideally should not be refined. In Figure 6.1 this leads to the local error in the upper right corner in the solution approximation and to the difference in H^1-error decays in the first row

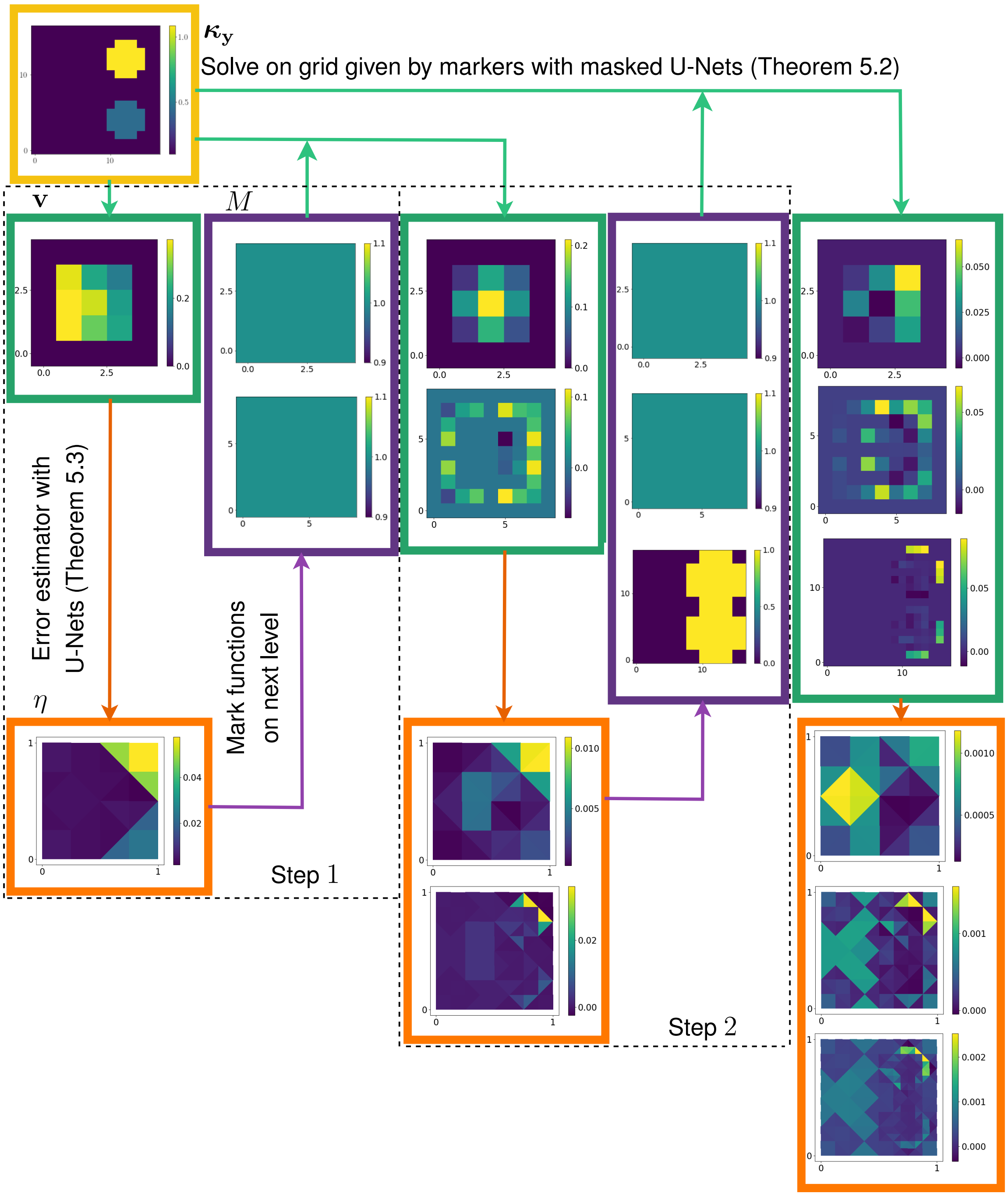

$\boldsymbol{\kappa}_{\mathbf{y}}$
Solve on grid given by markers with masked U-Nets (Theorem 5.2)
$\mathbf{v}$
M
Error estimator with U-Nets (Theorem 5.3)
Mark functions on next level
η
Step 1
Step 2

Figure 5.2: The derived CNN architecture is depicted for an approximation of three steps of the AFEM. The CNN mapping starts with the nodal interpolation of the parameter field $\boldsymbol{\kappa}_{\mathbf{y}}$ on the finest level given as an input image. As in Algorithm 1, in every step the solution $\mathbf{v}$ of the system of linear equations (Line 3, Line 9) is calculated (green arrows) and the solution is given in a multigrid decomposition (3.1) (green boxes), compare Theorem 5.2. The approximation of the error estimator η represented as in Theorem 5.3 is depicted in orange errors and its multigrid decomposition as in (4.1) in orange boxes. The derived markers are encoded by $0/1$-masks M visualized in the purple boxes. The masks are then used in the network of the next iteration to enforce an action only on local parts of the larger images to imitate a local mesh refinement. Note that one AFEM iteration corresponds to one black dashed box.

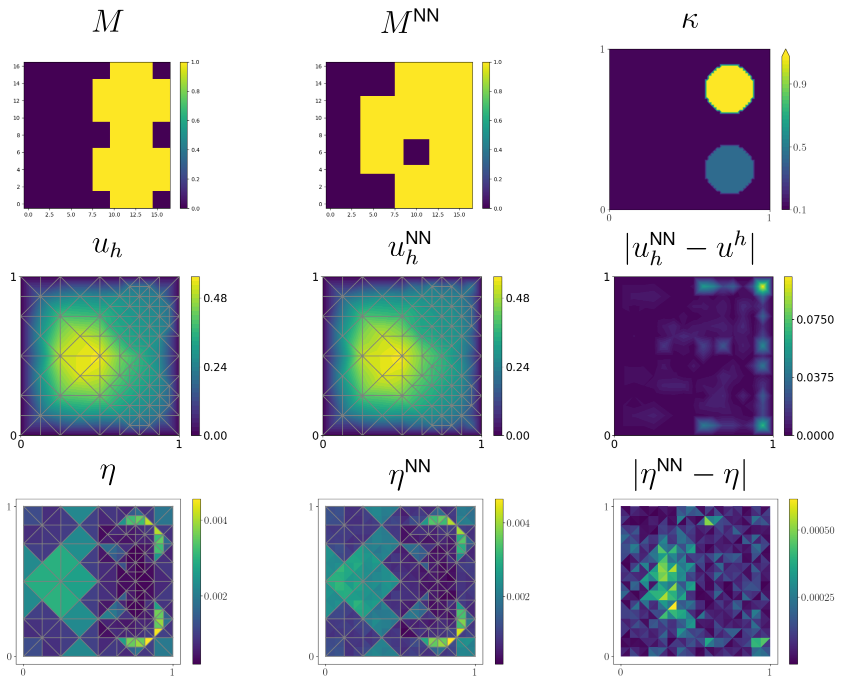


Figure 6.1: True and network prediction solutions, estimators and markers are plotted for the third step of the AFEM. From left to right, the first row shows the marking image on the third level, which was used for training, the marking image, which the network deduced from the estimator of the the second solution and the parameter field sample. The second row shows the Galerkin solution on the mesh used for training, the second plot shows the network output, and the last images shows the difference between the two. The last row shows the first the estimator of the third solution in the AFEM iteration, the network approximation of the estimator and the difference between the two. It can be seen that the pointwise distances are a magnitude smaller than the true values.

in Figures 6.2 and 6.3. In these figures, the graphs depict the H^1 and L^2 relative errors of the neural network approximation and the solutions of the AFEM. In Figure 6.2, the errors are plotted over the steps of the AFEM and in Figure 6.3 they are plotted against the degrees of freedom used in the approximation and the AFEM. The graphs in the first rows show results for the fully adaptive CNNs, choosing the markers based on the approximated estimators without using the mesh refinement used in the `FEniCS`[1] FEM package, which was used for the data generation. This element is still inexact and needs to be adjusted. The second rows show the results based on a CNN using masks known from the data generation. Since the decays with known masks match the true error decay of the test data, the main step to optimize is the mask generation.

In summary, for a local refinement with known masks, the relative H^1 and L^2 errors of the network show the same decay as the true AFEM for three steps. For a fully adaptive CNN, the L^2 errors match the true errors while the H^1 errors are larger, probably due to inexact masks, which will be a topic of future investigations.

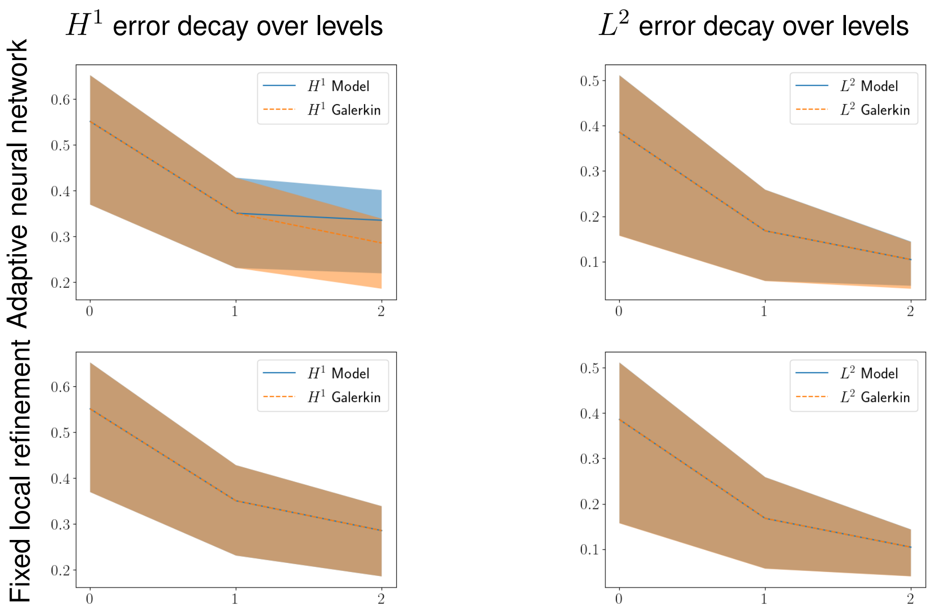


Figure 6.2: The average relative H^1 and L^2 errors are plotted against the number of steps of the AFEM $K = 1, 2, 3$ together with the error range from the minimal to the maximal error in every step.

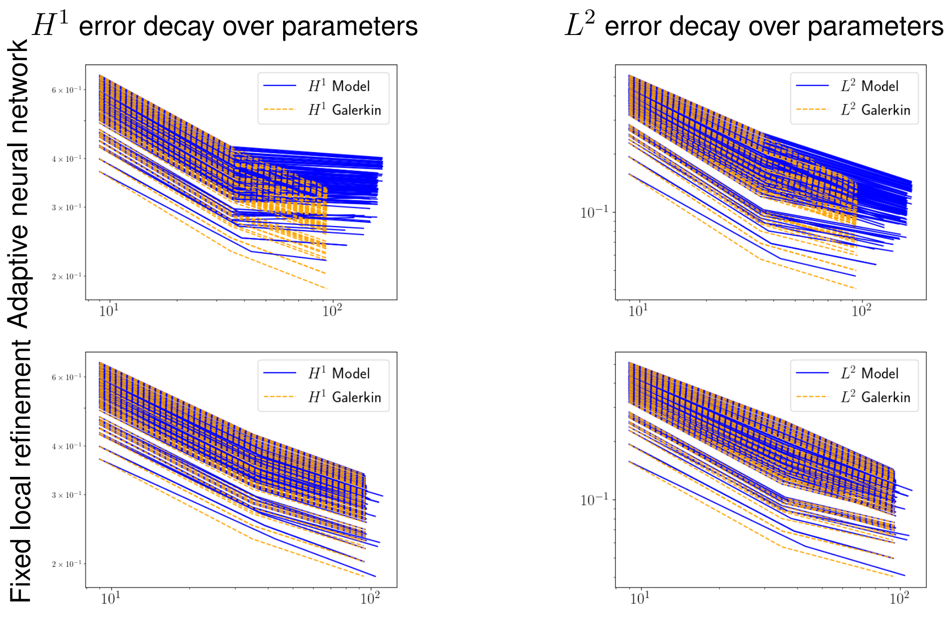


Figure 6.3: The average relative H^1 and L^2 errors are plotted against the number of parameters used by the AFEM and the neural network.

7 Outlook

In the paper, we derived an algorithm LLMG, which approximates the parametric diffusion PDE on a fixed grid based on a multigrid decomposition of the solution and a successive subspace correction algorithm. We showed that the derived algorithm can be approximated efficiently in the number of parameters by a derived CNN architecture. Furthermore, we showed that an efficient and reliable finite element error estimator can be approximated by a specific CNN construction. These results were combined to show upper bounds for the number of parameters of CNNs approximating a complete adaptive finite element scheme.

It is now interesting to put this architecture to use and explore the efficiency of the networks numerically, mainly with respect to two aspects. First, the number of calculations on each level should be reduced comparing to fully refined meshes [27] due to the submanifold sparse convolutions on sparse tensors. Note that the sparsity of the tensors stems from the used multigrid decomposition of the data. Second, the efficiency with respect to the number of samples needed for training should be explored, considering that in each step corrections with decreasing influence on the whole solutions need to be learned. This should lead to fewer training samples on fine grids and hence a more efficient training and data generation process.

In addition, one might be interested in deriving convergence results for the proposed adaptive scheme with the presented refinement and for different marking strategies. The direction of showing the CNN approximation results for other meshes without hanging nodes might also be of interest.

References

[1] M. S. Alnæ, J. Blechta, J. Hake, A. Johansson, B. Kehlet, A. Logg, C. Richardson, J. Ring, M. E. Rognes, and G. N. Wells. The fenics project version 1.5. *Archive of Numerical Software*, 3(100):9–23, 2015.

[2] K. Bhattacharya, B. Hosseini, N. Kovachki, and A. Stuart. Model reduction and neural networks for parametric pdes. *The SMAI journal of computational mathematics*, 7, 05 2020.

[3] D. Braess. *Finite elements: Theory, fast solvers, and applications in solid mechanics*. Cambridge University Press, 2007.

[4] D. Braess and W. Hackbusch. A new convergence proof for the multigrid method including the V-cycle. *Siam Journal on Numerical Analysis - SIAM J NUMER ANAL*, 20:967–975, 10 1983.

[5] A. Caboussat, M. Girardin, and M. Picasso. Error assessment of an adaptive finite elements—neural networks method for an elliptic parametric pde. *Computer Methods in Applied Mechanics and Engineering*, 421:116784, 2024.

[6] A. Caboussat, M. Girardin, and M. Picasso. Error assessment of an adaptive finite elements—neural networks method for an elliptic parametric pde. *Computer Methods in Applied Mechanics and Engineering*, 421:116784, 2024.

[7] C. Carstensen, M. Eigel, R. H. Hoppe, and C. Löbhard. A review of unified a posteriori finite element error control. *Numerical Mathematics: Theory, Methods and Applications*, 5(4):509–558, 2012.

[8] L. Chen. Deriving the x-z identity from auxiliary space method*. In Y. Huang, R. Kornhuber, O. Widlund, and J. Xu, editors, *Domain Decomposition Methods in Science and Engineering XIX*, pages 309–316, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[9] T. Chen and H. Chen. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE Transactions on Neural Networks*, 6(4):911–917, 1995.

[10] N. Dal Santo, S. Deparis, and L. Pegolotti. Data driven approximation of parametrized pdes by reduced basis and neural networks. *Journal of Computational Physics*, 416:109550, 2020.

[11] L. Diening, C. Kreuzer, and R. Stevenson. Instance optimality of the adaptive maximum strategy. *Foundations of Computational Mathematics*, 16(1):33–68, 2015.

[12] W. Dörfler. A convergent adaptive algorithm for Poisson's equation. *SIAM Journal on Numerical Analysis*, 33(3):1106–1124, 1996.

[13] V. Dumoulin and F. Visin. A guide to convolution arithmetic for deep learning. *ArXiv e-prints*, mar 2016.

[14] V. Dumoulin and F. Visin. conv_arithmetic. `https://github.com/vdumoulin/conv_arithmetic`, 2016.

[15] M. Eigel, N. Farchmin, S. Heidenreich, and P. Trunschke. Adaptive nonintrusive reconstruction of solutions to high-dimensional parametric pdes. *SIAM Journal on Scientific Computing*, 45(2):A457–A479, 2023.

[16] M. Eigel, C. J. Gittelson, C. Schwab, and E. Zander. Adaptive stochastic galerkin FEM. *Computer Methods in Applied Mechanics and Engineering*, 270:247–269, Mar. 2014.

[17] M. Eigel, M. Marschall, M. Pfeffer, and R. Schneider. Adaptive stochastic galerkin fem for lognormal coefficients in hierarchical tensor representations, 2020.

[18] M. Feischl and J. Bohn. Recurrent neural networks as optimal mesh refinement strategies. *CRC Preprint 2020/33*, 2020.

[19] C. Foucart, A. Charous, and P. F. Lermusiaux. Deep reinforcement learning for adaptive mesh refinement. *Journal of Computational Physics*, 491:112381, 2023.

[20] K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36:193–202, 1980.

[21] M. Geist, P. Petersen, M. Raslan, R. Schneider, and G. Kutyniok. Numerical solution of the parametric diffusion equation by deep neural networks. *Journal of Scientific Computing*, 88:22–88, 2021.

[22] A. Gillette, B. Keith, and S. Petrides. Learning robust marking policies for adaptive mesh refinement. *SIAM Journal on Scientific Computing*, 46(1):A264–A289, 2024.

[23] B. Graham, M. Engelcke, and L. van der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. *CVPR*, 2018.

[24] B. Graham and L. van der Maaten. Submanifold sparse convolutional networks. *arXiv preprint arXiv:1706.01307*, 2017.

[25] I. Gühring and M. Raslan. Approximation rates for neural networks with encodable weights in smoothness spaces. *Neural Networks*, 134:107–130, 2021.

[26] W. Hackbusch. *Multi-grid methods and applications*, volume 4. Springer Science & Business Media, 2013.

[27] C. Heiß, I. Gühring, and M. Eigel. Multilevel cnns for parametric pdes. *Journal of Machine Learning Research*, 24(373):1–42, 2023.

[28] N. Kovachki, S. Lanthaler, and S. Mishra. On universal approximation and error bounds for fourier neural operators. *Journal of Machine Learning Research*, 22(290):1–76, 2021.

[29] G. Kytyniok, P. Petersen, M. Raslan, and R. Schneider. A theoretical analysis of deep neural networks and parametric pdes. *Constructive Approximation*, 55:73–125, 2022.

[30] S. Lanthaler, S. Mishra, and G. E. Karniadakis. Error estimates for DeepONets: a deep learning framework in infinite dimensions. *Transactions of Mathematics and Its Applications*, 6(1):tnac001, 03 2022.

[31] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel. Handwritten digit recognition with a back-propagation network. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann, 1989.

[32] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021.

[33] L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3:218–229, 2021.

[34] K. O. Lye, S. Mishra, and R. Molinaro. A multi-level procedure for enhancing accuracy of machine learning algorithms, 2020.

[35] C. Marcati and C. Schwab. Exponential convergence of deep operator networks for elliptic partial differential equations. *SIAM Journal on Numerical Analysis*, 61(3):1513–1545, 2023.

[36] R. H. Nochetto, K. G. Siebert, and A. Veeser. Theory of adaptive finite element methods: An introduction. In R. DeVore and A. Kunoth, editors, *Multiscale, Nonlinear and Adaptive Approximation*, pages 409–542, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[37] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing.

[38] J. E. Schütte and M. Eigel. Adaptive multilevel neural networks for parametric PDEs with error estimation. In *ICLR 2024 Workshop on AI4DifferentialEquations In Science*, 2024.

[39] C. Schwab and J. Zech. Deep learning in high dimension: Neural network expression rates for generalized polynomial chaos expansions in UQ. *Analysis and Applications*, 17(01):19–55, 2019.

[40] T. Służalec, R. Grzeszczuk, S. Rojas, W. Dzwinel, and M. Paszyński. Quasi-optimal hp-finite element refinements towards singularities via deep neural network prediction. *Computers & Mathematics with Applications*, 142:157–174, 2023.

[41] A. L. Teckentrup, P. Jantsch, C. G. Webster, and M. Gunzburger. A multilevel stochastic collocation method for partial differential equations with random input data. *SIAM/ASA Journal on Uncertainty Quantification*, 3(1):1046–1074, 2015.

[42] R. Verfürth. *A Posteriori Error Estimation Techniques for Finite Element Methods*. Oxford University Press, 04 2013.

[43] S. Wang, H. Wang, and P. Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed deeponets. *Science Advances*, 7(40):eabi8605, 2021.

[44] J. Xu. Iterative methods by space decomposition and subspace correction. *SIAM Review*, 34(4):581–613, 1992.

[45] J. Yang, T. Dzanic, B. Petersen, J. Kudo, K. Mittal, V. Tomov, J.-S. Camier, T. Zhao, H. Zha, T. Kolev, et al. Reinforcement learning for adaptive mesh refinement. In *International Conference on Artificial Intelligence and Statistics*, pages 5997–6014. PMLR, 2023.

[46] D. Yarotsky. Error bounds for approximations with deep relu networks. *Neural Networks*, 94:103–114, 2017.

A Error Estimator Derivation

We consider the residual in variational form for error $e := u - u_h$, where u_h is the Bubnov-Galerkin approximation of u on V_h and $\mathcal{T}$ is an (exact) triangulation of domain D. The residual based error estimator is common knowledge in the FEM literature, cf. [3, 7, 42]. For the sake of a self-contained presentation, we provide the derivation in what follows since it may help the comprehension of the CNN approximation in this paper. It holds that

$$
\begin{aligned}
a_{\mathbf{y},h}(e,v) &= a_{\mathbf{y},h}(u,v) - a_{\mathbf{y},h}(u_h,v) = f(v) - a_{\mathbf{y},h}(u_h,v) = \int_D fv - \kappa_h(\cdot,\mathbf{y}) \langle \nabla u_h, \nabla v\rangle \,\mathrm{d}x \\
&= \sum_{T\in\mathcal{T}} \int_T fv - \kappa_h(\cdot,\mathbf{y}) \langle \nabla u_h, \nabla v\rangle \,\mathrm{d}x.
\end{aligned}
$$

Furthermore, let n_T be the unit outward normal vector to ∂T for $T \in \mathcal{T}$. Then, it holds that

$$
\begin{aligned}
a_{\mathbf{y},h}(e,v) &= \sum_{T\in\mathcal{T}} \int_T fv\mathrm{d}x + \int_T v\nabla\cdot(\kappa_h(\cdot,\mathbf{y})\nabla u_h)\mathrm{d}x - \int_{\partial T} v\kappa_h(\cdot,\mathbf{y})\frac{\partial u_h}{\partial n_T}\mathrm{d}s \\
&= \sum_{T\in\mathcal{T}} \int_T (f + \nabla\cdot(\kappa_h(\cdot,\mathbf{y})\nabla u_h))v\mathrm{d}x + \sum_{\gamma\in\partial\mathcal{T}} \int_\gamma v\kappa_h(\cdot,\mathbf{y}) \left(\left\langle \nabla u_h^{(1)}, n_\gamma^{(1)}\right\rangle + \left\langle \nabla u_h^{(2)}, n_\gamma^{(2)}\right\rangle\right)\mathrm{d}s.
\end{aligned}
$$

Here, $n_\gamma^{(1)}$ and $n_\gamma^{(2)}$ are the unit outward normal vectors of the elements of the mesh containing γ and $\nabla u_h^{(1)}, \nabla u_h^{(2)}$ are the gradients of u_h on the elements. With the Galerkin projection v_h of v on V_h and the definition of the jump (2.4), with some $\tilde{C} > 0$ one gets the estimate

$$
\begin{aligned}
a_{\mathbf{y},h}(e,v) &= \sum_{T\in\mathcal{T}} \int_T (f + \nabla\cdot(\kappa_h(\cdot,\mathbf{y})\nabla u_h))(v - v_h)\mathrm{d}x + \sum_{\gamma\in\partial\mathcal{T}} \int_\gamma [\![\kappa(\cdot,\mathbf{y})\nabla u_h\cdot\hat{n}]\!](v - v_h)\mathrm{d}s \\
&\leq \sum_{T\in\mathcal{T}} \|f + \nabla\cdot(\kappa_h(\cdot,\mathbf{y})\nabla u_h\|_{L_2(T)} \|v - v_h\|_{L_2(T)} + \sum_{\gamma\in\partial\mathcal{T}} \|[\![\kappa_h(\cdot,\mathbf{y})\nabla u_h]\!]\|_{L_2(\gamma)} \|v - v_h\|_{L_2(\gamma)} \\
&\leq \tilde{C}\,\|v\|_{H^1(\Omega)} \left(\sum_{T\in\mathcal{T}} h_T^2 \|f + \nabla\cdot(\kappa_h(\cdot,\mathbf{y})\nabla u_h)\|_{L_2(T)}^2 + \sum_{\gamma\in\partial\mathcal{T}} h_E \|[\![\kappa_h(\cdot,\mathbf{y})\nabla u_h]\!]\|_{L_2(\gamma)}^2\right)^{1/2}.
\end{aligned}
$$

Setting $v = e$ and with $\|v\|_{H^1(\Omega)} \leq C\|v\|_{a_{\mathbf{y},h}}$ we arrive at

$$
\begin{aligned}
\|e\|_{a_{\mathbf{y},h}}^2 &= \frac{(\|e\|_{a_{\mathbf{y},h}}^2)^2}{\|e\|_{a_{\mathbf{y},h}}^2} = \frac{(a_{\mathbf{y},h}(e,e))^2}{\|e\|_{a_{\mathbf{y},h}}^2} \\
&\leq \frac{1}{\|e\|_{a_{\mathbf{y},h}}^2} \tilde{C}^2 C^2 \|e\|_{a_{\mathbf{y},h}}^2 \left(\sum_{T\in\mathcal{T}} h_T^2 \|f + \nabla\cdot(\kappa_h(\cdot,\mathbf{y})\nabla u_h)\|_{L_2(T)}^2 + \sum_{\gamma\in\partial\mathcal{T}} h_T \|[\![\kappa_h(\cdot,\mathbf{y})\nabla u_h]\!]\|_{L_2(\gamma)}^2\right) \\
&\leq \hat{C} \sum_{T\in\mathcal{T}} h_T^2 \|f + \nabla\cdot(\kappa_h(\cdot,\mathbf{y})\nabla u_h)\|_{L_2(T)}^2 + h_T \|[\![\kappa_h(\cdot,\mathbf{y})\nabla u_h]\!]\|_{L_2(\partial T)}^2,
\end{aligned}
$$

which proofs reliability of the estimator with some $\hat{C} > 0$.

B Proofs of convergence of the levelwise local multigrid algorithm

The sequence of uniform meshes $(\mathcal{T}_k)_{k=1}^L$, the piecewise linear finite element function spaces over the meshes U^k and the space $V_h = \sum_{k=1}^L V^k$ are introduced in Section 3 with $V^k \subseteq U^k$. Furthermore, recall that the operator Q_k is defined as the ℓ^2–projection of the coefficients of functions in V_h onto the coefficients of V^k. The action of $A_\mathbf{y}$ restricted to the coefficient spaces of V^k is defined by $A_\mathbf{y}^k$.

B.1 Successive Subspace Correction

The successive subspace algorithm (SSC) approximates the solution $\mathbf{u} \in \mathbb{R}^{\sum_{k=1}^L n_k^2}$ to $A_\mathbf{y}\mathbf{u} = \mathbf{f}$ by iteratively updating the solution on the individual subspaces $\mathbb{R}^{n_k^2}$ for $k = 1, \dots, L$ by the weighted residual, see Algorithm 3.

Algorithm 3: Successive Subspace Correction $\mathrm{SSC}(\mathbf{w})$

```
1 for k=1,..., L do
2     $\mathbf{w} \leftarrow \mathbf{w} + \omega_\mathbf{y}^k(\mathbf{f}^k - Q_k A_\mathbf{y}\mathbf{w})$
3 end
4 return $\mathbf{w}$
```

The error of the current approximation after each step of the algorithm denoting the update by $\mathbf{w}_{\text{update}}$ can be written as

$$\mathbf{w}_{\text{update}} - \mathbf{u} = \mathbf{w} + \omega_\mathbf{y}^k(\mathbf{f}^k - Q_k A_\mathbf{y}\mathbf{w}) - \mathbf{u} = (I - \omega_\mathbf{y}^k Q_k A_\mathbf{y})(\mathbf{w} - \mathbf{u}).$$

Define the operator $T_k : \mathbb{R}^{\sum_{\ell=1}^L n_\ell} \to \mathbb{R}^{n_k^2}$ by $x \mapsto \omega_\mathbf{y}^k Q_k A_\mathbf{y} x$, where $A_\mathbf{y}^k$ is the restriction of $A_\mathbf{y}$ to V^k. Let $\lambda_{\max}(B)$ denote the largest eigenvalue of matrix B and $\omega_\mathbf{y}^k$ be the smoothing factor such that

$$0 < \omega_\mathbf{y}^k \leq \lambda_{\text{max}}(A_\mathbf{y}^k)^{-1}. \tag{B.1}$$

The error of the successive subspace correction algorithm then has the recursive form

$$\mathbf{u} - \mathrm{SSC}(\mathbf{w}) = (I - T_L)(I - T_{L-1}) \dots (I - T_0)(\mathbf{u} - \mathbf{w}).$$

To bound the error, the X-Z identity from [8] can be used.

Lemma B.1 ([8, Theorem 4]). *Suppose that $\left\| I - \omega_\mathbf{y}^k A_\mathbf{y}^k \right\|_{A_\mathbf{y}^k} < 1$ for each $k = 0, \dots, L$. Then there exists a $c_0 \geq 0$ such that*

$$\| (I - T_L)(I - T_{L-1}) \dots (I - T_1) \|_{A_\mathbf{y}}^2 = \frac{c_0}{1 + c_0}$$

with

$$c_0 = \sup_{\|v\|_{A_{\mathbf{y}}}=1} \inf_{\sum_{k=1}^{L} v_i = v} \sum_{k=1}^{L} \left\| \omega_{\mathbf{y}}^{k} (Q_k A_{\mathbf{y}} \sum_{i=k}^{L} v_i - \omega_{\mathbf{y}}^{k^{-1}} v_k) \right\|_{\bar{R}_k^{-1}}^{2},$$

where $\bar{R}_k = (2I - \omega_{\mathbf{y}}^k A_{\mathbf{y}}^k)\omega_{\mathbf{y}}^k$.

To ensure the condition in Lemma B.1, we consider the following result.

Lemma B.2 (similar to [4, Lemma 4.3])**.** *Let* $k \in [L]$ *and* $\kappa(\cdot, \mathbf{y}) > 0$ *everywhere. Then for any* $\mathbf{w} \in \mathbb{R}^{\mathcal{I}_V^k}$ *it holds that*

$$\left\| (I - \omega_{\mathbf{y}}^k A_{\mathbf{y}}^k)\mathbf{w} \right\|_{A_{\mathbf{y}}^k} < \|\mathbf{w}\|_{A_{\mathbf{y}}^k},$$

where $0 < \omega_{\mathbf{y}}^k \le \lambda_{\max}(A_{\mathbf{y}}^k)^{-1}$.

Proof. Let $\Omega_k = \operatorname{supp} V_k$. First, assume some $\varphi : \Omega_k \to \mathbb{R}$ with $\varphi = 0$ on $\partial\Omega_k$. If φ is not constant zero this implies that there exists a point $x_0 \in \Omega_k$ and $\varepsilon > 0$ such that $\nabla\varphi \neq 0$ on an ε neighborhood of x_0 denoted b $U_\varepsilon(x_0)$. Then, due to $\kappa(\cdot, \mathbf{y}) > 0$ everywhere, we obtain that

$$a_{\mathbf{y},k}(\varphi, \varphi) = \int_{\Omega_k} \kappa(\cdot, \mathbf{y}) \langle \nabla\varphi, \nabla\varphi \rangle \,\mathrm{d}x \ge \int_{U_\varepsilon(x_0)} \kappa(\cdot, \mathbf{y}) \langle \nabla\varphi, \nabla\varphi \rangle \,\mathrm{d}x > 0,$$

where we set $a_{\mathbf{y},k} = a_{\mathbf{y},h}$ as in (2.1) for $V_h = V_k$. Therefore, for any $\mathbf{w} \in \mathbb{R}^{\mathcal{I}_V^k}$ we have that

$$\mathbf{w}^\intercal A_{\mathbf{y}}^k \mathbf{w} = a_{\mathbf{y},k}\left(\sum_{i \in \mathcal{I}_V^k} \mathbf{w}_i \phi_i, \sum_{i \in \mathcal{I}_V^k} \mathbf{w}_i \phi_i \right) > 0$$

and hence $A_{\mathbf{y}}^k$ is positive definite. Let $N_k := |\mathcal{I}_V^k|$ and denote the eigenvalues and eigenvectors of $A_{\mathbf{y}}^k$ by $\lambda_i, \mathbf{v}^i$ for $i = 1, \dots, N_k$ with

$$\begin{aligned} A_{\mathbf{y}}^k \mathbf{v}^i &= \lambda_i \mathbf{v}^i \qquad \text{such that} \\ \delta_{i,j} &= \left\langle \mathbf{v}^i, \mathbf{v}^j \right\rangle_{\ell^2} \text{ for all } i, j = 1, \dots, N_k. \end{aligned}$$

Furthermore, for $\mathbf{w} \in \mathbb{R}^{N_k}$, let

$$J_k \mathbf{w} := (I - \omega_{\mathbf{y}}^k A_{\mathbf{y}}^k)\mathbf{w}.$$

Then, with $\mathbf{w} = \sum_{i=1}^{N_k} c_i \mathbf{v}^i$

$$J_k \mathbf{w} = \sum_{i=1}^{N_k} c_i (I - \omega_{\mathbf{y}}^k A_{\mathbf{y}}^k)\mathbf{v}^i = \sum_{i=1}^{N_k} c_i (1 - \omega_{\mathbf{y}}^k \lambda_i)\mathbf{v}^i.$$

Second,

$$|\mathbf{w}|^2 := \sum_{i=1}^{N_k} \lambda_i (1 - \lambda_i \omega_{\mathbf{y}}^k) c_i^2$$

defines a semi-norm due to $0 < \omega_{\mathbf{y}}^k \leq \lambda_{\max}(A_{\mathbf{y}}^k)^{-1}$. Then, the following statements hold

$$|\mathbf{w}|^2 = \sum_{i,j=1}^{N_k} (1-\lambda_i \omega_{\mathbf{y}}^k)\lambda_i \left\langle \mathbf{v}^i, \mathbf{v}^j \right\rangle_{\ell^2} c_i c_j = \left\langle \sum_{i=1}^{N_k} c_i (1-\lambda_i \omega_{\mathbf{y}}^k)\mathbf{v}^i, \sum_{j=1}^{N_k} c_j \mathbf{v}^j \right\rangle_{A_{\mathbf{y}}^k} = \left\langle J_k \mathbf{w}, \mathbf{w} \right\rangle_{A_{\mathbf{y}}^k},$$

$$\|\mathbf{w}\|_{A_{\mathbf{y}}^k}^2 = \left\langle A_{\mathbf{y}}^k \mathbf{w}, \mathbf{w} \right\rangle_{\ell^2} = \sum_{i,j=1}^{N_k} \lambda_i c_i c_j \left\langle \mathbf{v}^i, \mathbf{v}^j \right\rangle_{\ell^2} = \sum_{i=1}^{N_k} c_i^2 \lambda_i,$$

$$|\mathbf{w}|^2 = \sum_{i=1}^{N_k} \lambda_i c_i^2 - \sum_{i=1}^{N_k} \lambda_i \omega_{\mathbf{y}}^k c_i^2 < \sum_{i=1}^{N_k} \lambda_i c_i^2 = \|\mathbf{w}\|_{A_{\mathbf{y}}^k}^2 .$$

The last inequality holds true for $\mathbf{w} \neq 0$ since $A_{\mathbf{y}}^k$ is positive definite and $\omega_{\mathbf{y}}^k > 0$. Then, with the Hölder inequality,

$$\begin{aligned}
\|J_k \mathbf{w}\|_{A_{\mathbf{y}}^k} &= \sum_{i=1}^{N_k} \lambda_i (c_i (1-\lambda_i \omega_{\mathbf{y}}^k))^2 = \sum_{i=1}^{N_k} (\lambda_i^{1/3} |c_i|^{2/3})(\lambda_i^{2/3} |c_i|^{4/3} (1-\lambda_i \omega_{\mathbf{y}}^k)^2) \\
&\leq \left(\sum_{i=1}^{N_k} (\lambda_i^{1/3} |c_i|^{2/3})^3 \right)^{1/3} \left(\sum_{i=1}^{N_k} (\lambda_i^{2/3} |c_i|^{4/3} (1-\lambda_i \omega_{\mathbf{y}}^k)^2)^{3/2} \right)^{2/3} \\
&= \left(\sum_{i=1}^{N_k} \lambda_i |c_i|^2 \right)^{1/3} \left(\sum_{i=1}^{N_k} \lambda_i |c_i|^2 (1-\lambda_i \omega_{\mathbf{y}}^k)^3 \right)^{2/3} .
\end{aligned}$$

This yields the result by estimating

$$\begin{aligned}
\|J_k \mathbf{w}\|_{A_{\mathbf{y}}^k}^3 &= \left(\|J_k \mathbf{w}\|_{A_{\mathbf{y}}^k}^2 \right)^{3/2} \\
&\leq \left(\sum_{i=1}^{N_k} \lambda_i |c_i|^2 \right)^{1/2} \left(\sum_{i=1}^{N_k} \lambda_i |c_i|^2 (1-\lambda_i \omega_{\mathbf{y}}^k)^3 \right) \\
&= \|\mathbf{w}\|_{A_{\mathbf{y}}^k}^2 |J_k \mathbf{w}| \\
&< \|\mathbf{w}\|_{A_{\mathbf{y}}^k}^2 \|J_k \mathbf{w}\|_{A_{\mathbf{y}}^k}
\end{aligned}$$

and dividing by $\|J_k \mathbf{w}\|_{A_{\mathbf{y}}^k}$. □

Theorem B.1. *Assume that there exists a constant $C > 0$ such that $\lambda_{\max}(A_{\mathbf{y}}^k) \leq \lambda_{\max}(A_{\mathbf{y}}) \leq C$ for all $\mathbf{y} \in \Gamma$ and choose $0 < \omega^k \leq C^{-1}$. Then, the error decays with*

$$\|(I - T_L)(I - T_{L-1}) \dots (I - T_1)\|_{A_{\mathbf{y}}}^2 \leq \frac{c_0}{1 + c_0}$$

for some $c_0 \leq \frac{\lambda_{\max}(A_{\mathbf{y}})}{\lambda_{\min}(A_{\mathbf{y}})} L$. Furthermore, if there exist constants $c_1, c_2 > 0$ such that $c_1 \leq \kappa(\cdot, \mathbf{y}) \leq c_2$ for all $x \in D$ and $\mathbf{y} \in \Gamma$ (uniform boundedness) leads to a bound of the convergence rate $c_0 \leq cL$ independent of $\mathbf{y}$ for some $c > 0$.

Proof. To apply Lemma B.1, we only need to verify that the smoothing on the subspaces yields a contraction for each $k = 1, \dots, L$, i.e.

$$\|I - T_k\|_{A_{\mathbf{y}}^k} < 1.$$

This is established in Lemma B.2. The constant in Lemma B.1

$$c_0 = \sup_{\|\mathbf{v}\|_{A_{\mathbf{y}}}=1} \inf_{\sum_{k=1}^L Q_i^\intercal \mathbf{v}_i = \mathbf{v}} \sum_{k=1}^L \left\| \omega_{\mathbf{y}}^k Q_k A_{\mathbf{y}} \sum_{i=k}^L Q_i^\intercal \mathbf{v}_i - \mathbf{v}_k \right\|_{\bar{R}_k^{-1}}^2 ,$$

for $\bar{R}_k = (2I - \omega_{\mathbf{y}}^k A_{\mathbf{y}}^k)\omega_{\mathbf{y}}^k$ has to be bounded by a constant independent of $\mathbf{y}$. If $\|T\| < 1$ then $\|(I-T)^{-1}\| \leq \frac{1}{1-\|T\|}$. Therefore, $T := \frac{\omega^k}{2} A_{\mathbf{y}}^k$, (B.1) and $\|T\| \leq \frac{1}{2}\lambda_{\max}(A_{\mathbf{y}}^k)^{-1} \left\|A_{\mathbf{y}}^k\right\| = \frac{1}{2} < 1$ implies that

$$\left\|\bar{R}_k^{-1}\right\| = \left\|(2\omega^k(I-T))^{-1}\right\| \leq \frac{\lambda_{\max}(A_{\mathbf{y}}^k)}{2} \left\|(I-T)^{-1}\right\| \leq \frac{\lambda_{\max}(A_{\mathbf{y}}^k)}{2} \frac{1}{1-\|T\|} = \lambda_{\max}(A_{\mathbf{y}}^k).$$

For $p_k : V \to V_k$ the $A_{\mathbf{y}}$ orthogonal projection onto V_k, it holds that $Q_k A_{\mathbf{y}} = A_{\mathbf{y}}^k p_k$. Therefore,

$$\begin{aligned}
c_0 &\leq \sup_{\|\mathbf{v}\|_{A_{\mathbf{y}}}=1} \inf_{\sum_{k=1}^L Q_i^\intercal \mathbf{v}_i = \mathbf{v}} \sum_{k=1}^L \lambda_{\max}(A_{\mathbf{y}}^k) \left\| \omega^k Q_k A_{\mathbf{y}} \sum_{i=k}^L Q_i^\intercal \mathbf{v}_i - \mathbf{v}_k \right\|^2 \\
&= \sup_{\|\mathbf{v}\|_{A_{\mathbf{y}}}=1} \inf_{\sum_{k=1}^L Q_i^\intercal \mathbf{v}_i = \mathbf{v}} \sum_{k=1}^L \lambda_{\max}(A_{\mathbf{y}}^k) \left(\left\| \omega^k Q_k A_{\mathbf{y}} \sum_{i=k+1}^L Q_i^\intercal \mathbf{v}_i + (\omega^k A_{\mathbf{y}}^k - I)\mathbf{v}_k \right\| \right)^2 \\
&= \sup_{\mathbf{v}\neq 0} \inf_{\sum_{k=1}^L Q_i^\intercal \mathbf{v}_i = \mathbf{v}} \sum_{k=1}^L \lambda_{\max}(A_{\mathbf{y}}^k) \left(\left\| \omega^k Q_k A_{\mathbf{y}} \sum_{i=k+1}^L Q_i^\intercal \frac{\mathbf{v}_i}{\|\mathbf{v}\|_{A_{\mathbf{y}}}} + (\omega^k A_{\mathbf{y}}^k - I)\frac{\mathbf{v}_k}{\|\mathbf{v}\|_{A_{\mathbf{y}}}} \right\| \right)^2 \\
&\leq \sup_{\mathbf{v}\neq 0} \inf_{\sum_{k=1}^L Q_i^\intercal \mathbf{v}_i = \mathbf{v}} \sum_{k=1}^L \lambda_{\max}(A_{\mathbf{y}}^k) \left(\left\| \omega^k Q_k A_{\mathbf{y}} \sum_{i=k+1}^L \frac{Q_i^\intercal \mathbf{v}_i}{\lambda_{\min}(A_{\mathbf{y}})^{\frac{1}{2}} \|\mathbf{v}\|} + \frac{(\omega^k A_{\mathbf{y}}^k - I)\mathbf{v}_k}{\lambda_{\min}(A_{\mathbf{y}})^{\frac{1}{2}} \|\mathbf{v}\|} \right\| \right)^2 \\
&= \lambda_{\min}(A_{\mathbf{y}})^{-1} \sup_{\|\mathbf{v}\|_2=1} \inf_{\sum_{k=1}^L Q_i^\intercal \mathbf{v}_i = \mathbf{v}} \sum_{k=1}^L \lambda_{\max}(A_{\mathbf{y}}^k) \left(\left\| \omega^k Q_k A_{\mathbf{y}} \sum_{i=k+1}^L Q_i^\intercal \mathbf{v}_i + (\omega^k A_{\mathbf{y}}^k - I)\mathbf{v}_k \right\| \right)^2 \\
&\leq \lambda_{\min}(A_{\mathbf{y}})^{-1} \sup_{\|\mathbf{v}\|_2=1} \inf_{\sum_{k=1}^L Q_i^\intercal \mathbf{v}_i = \mathbf{v}} \sum_{k=1}^L \lambda_{\max}(A_{\mathbf{y}}^k) \left(\omega^k \|Q_k A_{\mathbf{y}}\| \left\| \sum_{i=k+1}^L Q_i^\intercal \mathbf{v}_i \right\| + \left\|\omega^k A_{\mathbf{y}}^k - I\right\| \|\mathbf{v}_k\| \right)^2 \\
&\leq \lambda_{\min}(A_{\mathbf{y}})^{-1} \sup_{\|\mathbf{v}\|_2=1} \inf_{\sum_{k=1}^L Q_i^\intercal \mathbf{v}_i = \mathbf{v}} \sum_{k=1}^L \lambda_{\max}(A_{\mathbf{y}}^k) \left(\frac{\lambda_{\max}(A_{\mathbf{y}})}{\lambda_{\max}(A_{\mathbf{y}})} \left\| \sum_{i=k+1}^L Q_i^\intercal \mathbf{v}_i \right\| + \left(1 - \omega^k \lambda_{\min}(A_{\mathbf{y}}^k)\right) \|\mathbf{v}_k\| \right)^2 \\
&\leq \frac{\lambda_{\max}(A_{\mathbf{y}})}{\lambda_{\min}(A_{\mathbf{y}})} \sup_{\|\mathbf{v}\|_2=1} \inf_{\sum_{k=1}^L Q_i^\intercal \mathbf{v}_i = \mathbf{v}} \sum_{k=1}^L \left(\left\| \sum_{i=k+1}^L Q_i^\intercal \mathbf{v}_i \right\| + \|\mathbf{v}_k\| \right)^2 \\
&\leq \frac{\lambda_{\max}(A_{\mathbf{y}})}{\lambda_{\min}(A_{\mathbf{y}})} \sup_{\|\mathbf{v}\|_2=1} \inf_{\sum_{k=1}^L Q_i^\intercal \mathbf{v}_i = \mathbf{v}} \sum_{k=1}^L 1 \\
&\leq \frac{\lambda_{\max}(A_{\mathbf{y}})}{\lambda_{\min}(A_{\mathbf{y}})} L.
\end{aligned}$$

This is derived by using that $\|v\|_{A_{\mathbf{y}}}^2 \geq \lambda_{\min}(A_{\mathbf{y}})\,\|v\|^2$, $\max_k \lambda_{\max}(A_{\mathbf{y}}^k) \geq \lambda_{\min}(A_{\mathbf{y}})$ and $Q_i\mathbf{v}_i$ are orthogonal with respect to the ℓ^2 scalar product for $i = 1,\dots,L$. Therefore, the claim follows from the assumption that $\lambda_{\min}(A_{\mathbf{y}})$ and $\lambda_{\max}(A_{\mathbf{y}})$ are uniformly bounded from below and above for all $\mathbf{y}$, respectively. □

B.2 Local multigrid algorithm

Algorithm 4: Local Multigrid Algorithm LMG($\mathbf{u}_0$)

```
1 $\mathbf{u} = \mathbf{u}_0$
2 for k=L,..., 0 do
3     $\mathbf{u} = \mathbf{u} + \omega_{\mathbf{y}}^k(\mathbf{f}_k - Q_k A_{\mathbf{y}}\mathbf{u})$
4 end
5 for k=0,..., L do
6     $\mathbf{u} = \mathbf{u} + \omega_{\mathbf{y}}^k(\mathbf{f}_k - Q_k A_{\mathbf{y}}\mathbf{u})$
7 end
8 return $\mathbf{u}$
```

Building on the analysis of the successive subspace correction algorithm in Appendix B.1, the error of the Local Multigrid Algorithm 4 can be expressed similar to the successive subspace correction algorithm as

$$\mathbf{u} - LMG(\mathbf{u}_0) = (I - T_0)\dots(I - T_{L-1})(I - T_L)(I - T_L)(I - T_{L-1})\dots(I - T_0)(u - u_0).$$

Since the order of the subspaces in Theorem B.1 are not specified, the same constant smoothing factor $\omega_{\mathbf{y}}^k = \omega$ can be chosen such that the same constant $c_0 > 0$ satisfies

$$\|\mathbf{u} - \mathrm{LMG}(\mathbf{u}_0)\|_{A_{\mathbf{y}}}^2 \leq \left(\frac{c_0}{1+c_0}\right)^2 \|\mathbf{u} - \mathbf{u}^0\|_{A_{\mathbf{y}}}^2 .$$

B.3 Smoothing with multiple levels

For one smoothing step, the calculation of $Q_k A_{\mathbf{y}}\mathbf{u}$ is required. Since $\mathbf{u}$ is given as a levelwise discretization, we consider the evaluation of the multiplication on each level separately. Recall the auxiliary vectors $\tilde{\mathbf{u}}^k$ and $\bar{\mathbf{u}}^k$ in (3.3) and (3.4). Using the decomposition $\mathbf{u} = \mathbf{u}^{<k} + Q_k^\intercal \mathbf{u}^k + \mathbf{u}^{>k}$ facilitates a levelwise calculation as described below.

B.3.1 Coarse grid smoothing

For $Q_k A_{\mathbf{y}} \mathbf{u}^{<k}$ we get the following result.

Lemma B.3. *Let $\tilde{\mathbf{u}}^k$ be defined as in* (3.3) *by*

$$\tilde{\mathbf{u}}^1 := 0, \quad \tilde{\mathbf{u}}^k := P_{k-1}\left(\tilde{\mathbf{u}}^{k-1} + \overline{\mathbf{u}^{k-1}}\right)$$

with $\overline{\mathbf{u}_i^k} \in \mathbb{R}^{\overline{\mathcal{I}_V^k}}$ equal to $\mathbf{u}_i^k$ for $i \in \mathcal{I}_V^k$ and zero otherwise. Then

$$Q_k A_{\mathbf{y}} \mathbf{u}^{<k} = \overline{A_{\mathbf{y}}^k} \tilde{\mathbf{u}}^k.$$

Proof of Lemma B.3. Note that $Q_1 A_{\mathbf{y}} \mathbf{u}^{<1} = 0 = A_{\mathbf{y}}^k \tilde{\mathbf{u}}^1$. For $k = 2, \dots, L$ we show that for $x \in \operatorname{supp} V^k$

$$\sum_{\ell=1}^{k-1} \sum_{i \in \mathcal{I}_V^\ell} \mathbf{u}_i^\ell \varphi_i^\ell(x) = \sum_{i \in \overline{\mathcal{I}_V^k}} \tilde{\mathbf{u}}_i^k \varphi_i^k(x).$$

The proof is by induction. For $k = 1$, both sides are equal to 0. Assuming the statement holds for k, we get for $k+1$ and $x \in \operatorname{supp} V^{k+1} \subset \operatorname{supp} V^k$ that

$$\begin{aligned}
\sum_{\ell=1}^{k} \sum_{i \in \mathcal{I}_V^\ell} \mathbf{u}_i^\ell \varphi_i^\ell(x) &= \sum_{\ell=1}^{k-1} \sum_{i \in \mathcal{I}_V^\ell} \mathbf{u}_i^\ell \varphi_i^\ell(x) + \sum_{i \in \mathcal{I}_V^k} \mathbf{u}_i^k \varphi_i^k(x) = \sum_{i \in \overline{\mathcal{I}_V^k}} \tilde{\mathbf{u}}_i^k \varphi_i^k(x) + \sum_{i \in \overline{\mathcal{I}_V^k}} \overline{\mathbf{u}_i^k} \varphi_i^k(x) \\
&= \sum_{i \in \overline{\mathcal{I}_V^k}} (\tilde{\mathbf{u}}_i^k + \overline{\mathbf{u}_i^k}) \varphi_i^k(x) = \sum_{j \in \overline{\mathcal{I}_V^{k+1}}} \sum_{i \in \overline{\mathcal{I}_V^k}} (\tilde{\mathbf{u}}^k + \overline{\mathbf{u}^k})_i (P_k^\intercal)_{i,j} \varphi_j^{k+1}(x) \\
&= \sum_{j \in \overline{\mathcal{I}_V^{k+1}}} (P_k(\tilde{\mathbf{u}}^k + \overline{\mathbf{u}^k}))_j \varphi_j^{k+1}(x) = \sum_{i \in \overline{\mathcal{I}_V^{k+1}}} \tilde{\mathbf{u}}_i^{k+1} \varphi_i^{k+1}(x).
\end{aligned}$$

Then, for $j \in \mathcal{I}_V^k$ it holds that

$$\begin{aligned}
\left(Q_k A_{\mathbf{y}} \mathbf{u}^{<k}\right)_j &= \sum_{\ell=1}^{k-1} \sum_{i \in \mathcal{I}_V^\ell} \mathbf{u}_i^\ell \int \kappa_h(x, \mathbf{y}) \left\langle \nabla \varphi_i^\ell(x), \nabla \varphi_j^k(x) \right\rangle \mathrm{d}x \\
&= \int_{\operatorname{supp} V^k} \kappa_h(x, \mathbf{y}) \left\langle \nabla \sum_{\ell=1}^{k-1} \sum_{i \in \mathcal{I}_V^\ell} \mathbf{u}_i^\ell \varphi_i^\ell(x), \nabla \varphi_j^k(x) \right\rangle \mathrm{d}x \\
&= \int \kappa_h(x, \mathbf{y}) \left\langle \nabla \sum_{i \in \overline{\mathcal{I}_V^k}} \tilde{\mathbf{u}}_i^k \varphi_i^k(x), \nabla \varphi_j^k(x) \right\rangle \mathrm{d}x \\
&= \sum_{i \in \overline{\mathcal{I}_V^k}} \tilde{\mathbf{u}}_i^k \int \kappa_h(x, \mathbf{y}) \left\langle \nabla \varphi_i^k(x), \nabla \varphi_j^k(x) \right\rangle \mathrm{d}x \\
&= (\overline{A_{\mathbf{y}}^k} \tilde{\mathbf{u}}^k)_j.
\end{aligned}$$

□

B.3.2 Fine grid smoothing

We now consider $Q_k A_{\mathbf{y}} \mathbf{u}^{>k}$.

Lemma B.4. *Let* $\bar{\mathbf{u}}^k$ *be defined as in* (3.4) *by*

$$\bar{\mathbf{u}}^L := 0, \quad \bar{\mathbf{u}}^k := P_k^\intercal \left(\bar{\mathbf{u}}^{k+1} + \overline{A_{\mathbf{y}}^{k+1}}^\intercal \mathbf{u}^{k+1} \right).$$

Then,

$$Q_k A_{\mathbf{y}} \mathbf{u}^{>k} = \bar{\mathbf{u}}^k|_{\mathcal{I}_V^k}.$$

Proof. We prove the statement again by induction. Note that for $k = L$, it holds that $Q_k A_{\mathbf{y}} \mathbf{u}^{>L} = 0 = \bar{\mathbf{u}}^L$. Assuming that the statement holds for $k+1$, i.e. for $j \in \mathcal{I}_V^{k+1}$

$$(Q_{k+1} A_{\mathbf{y}} \mathbf{u}^{>k+1})_j = \sum_{\ell=k+2}^{L} \sum_{i \in \mathcal{I}_V^\ell} \mathbf{u}_i^\ell \int \kappa_h(\cdot, \mathbf{y}) \left\langle \nabla \varphi_i^\ell, \nabla \varphi_j^{k+1} \right\rangle \mathrm{d}x = \bar{\mathbf{u}}_j^{k+1},$$

we show that the statement also is true for k. In fact, for $j \in \mathcal{I}_V^k$ we deduce that

$$\begin{aligned}
(Q_k A_{\mathbf{y}} \mathbf{u}^{>k})_j &= \sum_{\ell=k+1}^{L} \sum_{i \in \mathcal{I}_V^\ell} \mathbf{u}_i^\ell \int \kappa_h(\cdot, \mathbf{y}) \left\langle \nabla \varphi_i^\ell, \nabla \varphi_j^k \right\rangle \mathrm{d}x \\
&= \sum_{m \in \overline{\mathcal{I}_V^{k+1}}} (P_k^\intercal)_{jm} \sum_{\ell=k+1}^{L} \sum_{i \in \mathcal{I}_V^\ell} \mathbf{u}_i^\ell \int \kappa_h(\cdot, \mathbf{y}) \left\langle \nabla \varphi_i^\ell, \nabla \varphi_m^{k+1} \right\rangle \mathrm{d}x \\
&= \sum_{m \in \overline{\mathcal{I}_V^{k+1}}} (P_k^\intercal)_{jm} \left(\sum_{i \in \overline{\mathcal{I}_V^{k+1}}} \mathbf{u}_i^{k+1} a_{\mathbf{y},h}(\varphi_i^{k+1}, \varphi_m^{k+1}) + \sum_{\ell=k+2}^{L} \sum_{i \in \mathcal{I}_V^\ell} \mathbf{u}_i^\ell a_{\mathbf{y},h}(\varphi_i^\ell, \varphi_m^{k+1}) \right) \\
&= \sum_{m \in \overline{\mathcal{I}_V^{k+1}}} (P_k^\intercal)_{jm} \left(\left(\overline{A_{\mathbf{y}}^{k+1}}^\intercal \mathbf{u}^{k+1} \right)_m + \bar{\mathbf{u}}_m^{k+1} \right) \\
&= \left(P_k^\intercal \left(\overline{A_{\mathbf{y}}^{k+1}}^\intercal \mathbf{u}^{k+1} + \bar{\mathbf{u}}^{k+1} \right) \right)_j = \bar{\mathbf{u}}_j^k.
\end{aligned}$$

□

C CNNs

Figure C.1 illustrates the different convolutions used in our architecture in Section 4.

D Proofs of CNN approximation theorems

Corollary D.1 (Multiplication approximation [27, Corollary 13]). *Let* σ *satisfy Assumption 5.1. Let* $W \in \mathbb{N}$ *be the input image size and* $B > 0$ *the range of the input values and* $\varepsilon \in (0, 1/2)$. *Then there exists a CNN* Ψ *with activation function* σ, *two-channel input and one channel output, spatial dimension of the kernels* 1, 2 *layers and number of parameters at most* 9 *such that*

$$\|\Psi(\mathbf{x}, \mathbf{y}) - \mathbf{x} \odot \mathbf{y}\|_{L^\infty([-B,B]^{2 \times W \times W})} \leq \varepsilon.$$

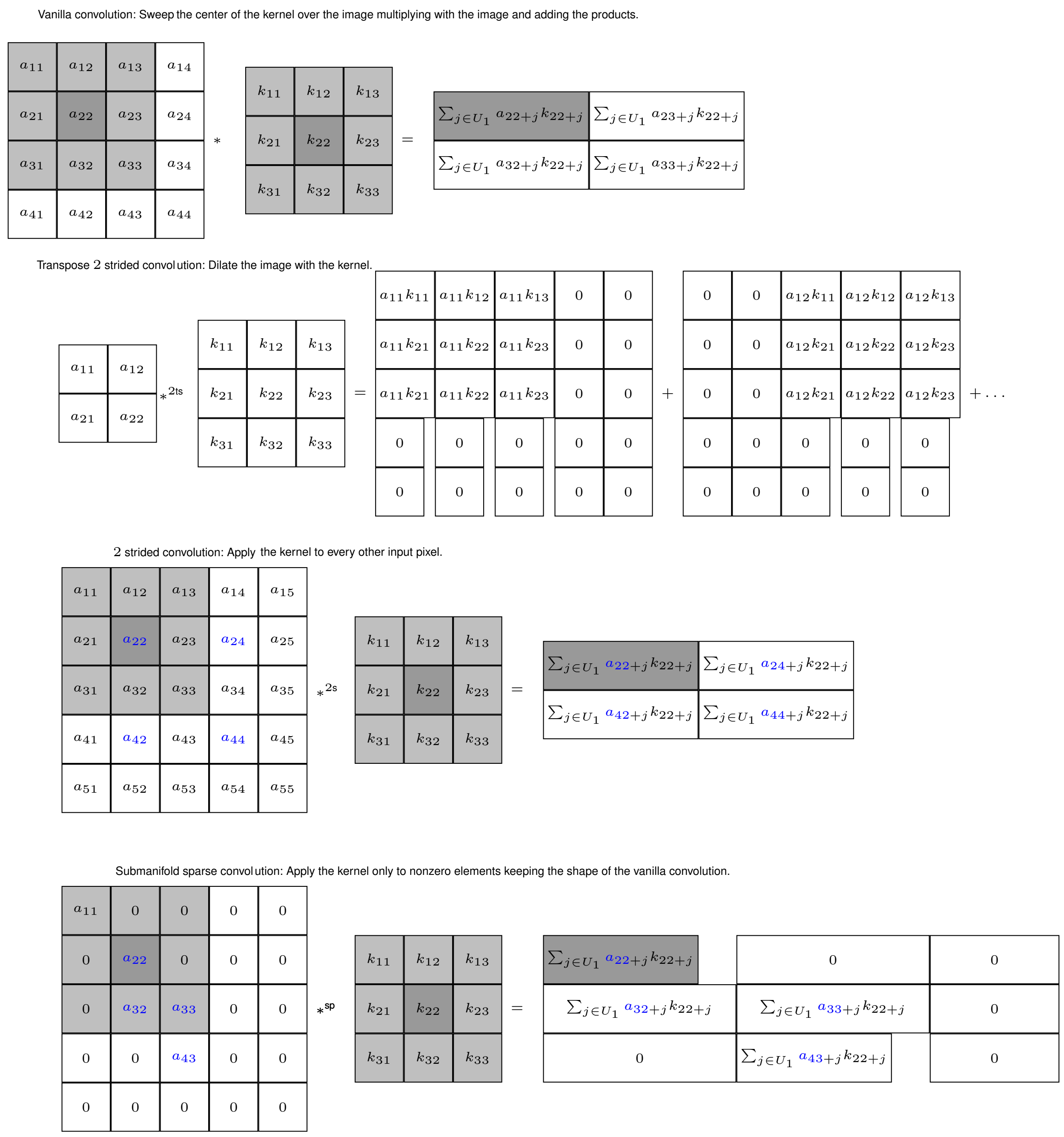

Vanilla convolution: Sweep the center of the kernel over the image multiplying with the image and adding the products.
Transpose 2 strided convolution: Dilate the image with the kernel.
2 strided convolution: Apply the kernel to every other input pixel.
Submanifold sparse convolution: Apply the kernel only to nonzero elements keeping the shape of the vanilla convolution.

Figure C.1: Visualization of the vanilla, the 2 strided, the 2 transpose strided and the submanifold sparse convolution, where $U := \{(i,j) : |i| \leq {}^{W-1}/_{2}, |j| \leq {}^{H-1}/_{2}\}$, where W and H denote the uneven width and height of the kernel respectively.

Lemma D.1 (Concatenation approximation [27, Lemma 20]). *Let $n, d_1, \dots, d_{n+1} \in \mathbb{N}$, $i \in [n]$ and $f_i : \mathbb{R}^{d_i} \to \mathbb{R}^{d_{i+1}}$ be continuous and let $F : \mathbb{R}^{d_1} \to \mathbb{R}^{d_{n+1}}$ be the concatenation $F := f_n \circ \dots \circ f_1$. Let $M, \varepsilon > 0$. Then there exists $\tilde{M}, \tilde{\varepsilon} > 0$ such that $\left\| f_i - \tilde{f}_i \right\|_{L^\infty([-\tilde{M},\tilde{M}]^{d_i})} \leq \tilde{\varepsilon}$ for each $i \in [n]$ and some $\tilde{f}_i : \mathbb{R}^{d_i} \to \mathbb{R}^{d_{i+1}}$ implies*

$$\left\| F - \tilde{f}_n \circ \dots \circ \tilde{f}_1 \right\|_{L^\infty([-M,M]^{d_1})} \leq \varepsilon.$$

D.1 Solver approximation

Using the notation from [27, Definition 14], for $\kappa \in H_0^1(D)$ $k = 1, \dots, L$, $\ell = 1, \dots, 6$ and $i \in \mathcal{I}_U^k$ define

$$\begin{aligned} \Upsilon(\kappa, \mathcal{T}_k, \ell, i) &:= \int_{T_i^\ell} \kappa \mathrm{d}x, \\ \Upsilon(\kappa, \mathcal{T}_k, \ell) &:= (\Upsilon(\kappa, \mathcal{T}_k, \ell, i))_{i \in \mathcal{I}_U^k}, \\ \Upsilon(\kappa, \mathcal{T}_k) &:= (\Upsilon(\kappa, \mathcal{T}_k, \ell))_{\ell=1,\dots,6}. \end{aligned}$$

In [27, Theorem 16] it was shown that $\left(A_{\mathbf{y}}^k \mathbf{u}^k\right)_{\text{img}}$ can be written as an application of a kernel to $\mathbf{u}_{\text{img}}^k$ and a multiplication with $\Upsilon(\kappa_h(\cdot, \mathbf{y}), \mathcal{T}_k)$, where $A_{\mathbf{y}}^k$ only considers indices in $\mathcal{I}_V^k \times \mathcal{I}_V^k$. We generalize the previous result to the application of $\overline{A_{\mathbf{y}}^k}$ with indices in $\mathcal{I}_V^k \times \overline{\mathcal{I}_V^k}$ by the use of multiple channels of in the following theorem.

Definition D.1 (Translation). *Let $m \in \mathbb{N}$ be the number of basis functions with overlapping support, i.e., for i the index of an inner node let $m := \left|\{\varphi_j^k : \sup \varphi_i^k \cap \sup \varphi_j^k \neq \emptyset\}\right|$. For $\mathbf{v}_{img}^k \in \mathbb{R}^{\mathcal{I}_U^k}$, let $T\mathbf{v}_{img}^k \in \mathbb{R}^{m \times \mathcal{I}_U^k}$ be defined by*

$$T\mathbf{v}_{img}^k := \left[T^{(1)}\mathbf{v}_{img}^k, \dots, T^{(m)}\mathbf{v}_{img}^k\right],$$

where $T^{(1)}, \dots, T^{(m)}$ defines the translation such that for $i \in \mathcal{I}_U^k$

$$(T\mathbf{v}_{img}^k)_i = \left[(\mathbf{v}_{img}^k)_{i+p_1}, \dots, (\mathbf{v}_{img}^k)_{i+p_m}\right],$$

where $i + p_1, \dots, i + p_m$ denote the indices of the basis functions with overlapping support and $p_1, \dots, p_m$ denote the directions with $p_1 = 0$. Note that the directions $p_1, \dots, p_m$ are constant for all i due to the used uniformly refined meshes as depicted in the first row in Figure 1.2. Note that for the depicted meshes, $m = 7$ holds true and is independent of k.

Based on the definition of the translated images, the following theorem can be formulated.

Theorem D.1. *Let $m \in \mathbb{N}$ be as in Definition D.1. There exist kernels $K^{(\ell)} \in \mathbb{R}^{1 \times m \times 1 \times 1}, \ell = 1, \dots, 6$ such that for*

$$F^k : \mathbb{R}^{(m+7) \times \mathcal{I}_V^k} \to \mathbb{R}^{\mathcal{I}_V^k}, \quad \left(\mathbf{v}_{img}^k, \bar{\boldsymbol{\kappa}}^{(1)}, \dots, \bar{\boldsymbol{\kappa}}^{(6)}\right) \mapsto \sum_{\ell=1}^{6} \bar{\boldsymbol{\kappa}}^{(\ell)} \odot (\mathbf{v}_{img}^k *^{sp} K^{(\ell)}),$$

for $\mathbf{v}^k \in \mathbb{R}^{\overline{\mathcal{I}_V^k}}$, $M^k = 1 \in \mathbb{R}^{\mathcal{I}_V^k}$ it holds that

$$F^k \left(M_{img}^k \odot T\mathbf{v}_{img}^k, \Upsilon(\kappa_h(\cdot, \mathbf{y}), \mathcal{T}_k)\right) = \left(\overline{A_{\mathbf{y}}^k}\, \mathbf{v}^k\right)_{img}.$$

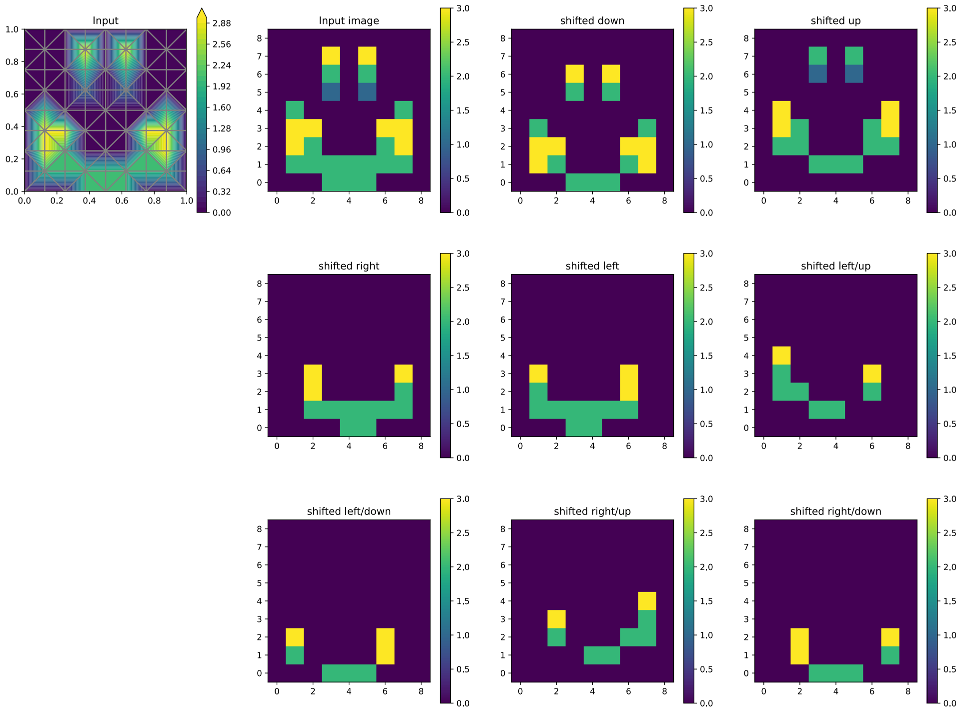


Figure D.1: This is a visualization of the translation Definition D.1. On the left the corresponding function to some input image is plotted. Since the mesh is also plotted, it can be seen that each nodal hat function has an overlapping support with nine other nodal hat functions including itself, i.e. $m = 9$. The output of the translation is a stack of the other nine images plotted. Each image shows the original image shifted in the direction of the nodal hat functions with overlapping support, but only on the support of the input image. This can be interpreted as shifting the image and then multiplying with a mask. Fixing one pixel index, the values of the pixel and all surrounding pixels are now saved in the pixel index in different images instead.

Note that since $\left(\overline{A^k_{\mathbf{y}}}\,\overline{\mathbf{v}^k}\right)_{\text{img}}$ is zero at every index $i \in \mathcal{I}^k_U \backslash \mathcal{I}^k_V$, in contrast to the definition in [27] the convolution here only needs to be applied to nodes $i \in \mathcal{I}^k_V$ since otherwise zero nodes would be assigned nonzero values after one convolution. This submanifold sparse convolution is denoted by $*^{\text{sp}}$ here.

Proof. First, let $i \in \mathcal{I}^k_U \backslash \mathcal{I}^k_V$. Since $\overline{A^k_{\mathbf{y}}} : \mathbb{R}^{\overline{\mathcal{I}^k_V}} \to \mathbb{R}^{\mathcal{I}^k_V}$ only maps to indices in $\mathcal{I}^k_V$, it holds that $(\overline{A^k_{\mathbf{y}}}\,\overline{\mathbf{v}^k})_{\text{img}_i} = 0$. Furthermore, for $\mathbf{w} := M^k_{\text{img}} \odot T\mathbf{v}^k_{\text{img}}$ we have that $\mathbf{w}_i = 0$ and $*^{\text{sp}}$ only acts on the indices $\mathcal{I}^k_V$, leaving everything else at 0. Therefore,

$$(\overline{A^k_{\mathbf{y}}}\,\overline{\mathbf{v}^k})_{\text{img}_i} = 0 = \sum_{\ell=1}^{6} \bar{\boldsymbol{\kappa}}^{(\ell)}_i \cdot 0 = \sum_{\ell=1}^{6} \bar{\boldsymbol{\kappa}}^{(\ell)}_i (\mathbf{w} *^{\text{sp}} K^{(\ell)})_i.$$

Second, for $i \in \mathcal{I}^k_V$ and $j \in \overline{\mathcal{I}^k_V}$ as in [27, proof of Theorem 16] with $C_{ijk} := \int_{T^k_i} \langle \nabla\varphi_i, \nabla\varphi_j \rangle dx$ it holds that

$$(\overline{A^k_{\mathbf{y}}})_{ij} = \int_D \kappa_h(\cdot,\mathbf{y}) \langle \nabla\varphi_i, \nabla\varphi_j \rangle dx = \sum_{\ell=1}^{6} \int_{T^\ell_i} \kappa_h(\cdot,\mathbf{y}) \langle \nabla\varphi_i, \nabla\varphi_j \rangle dx = \sum_{\ell=1}^{6} \Upsilon(\kappa_h(\cdot,\mathbf{y}), \mathcal{T}_k, \ell, i) C_{ij\ell}.$$

Since $C_{ijk} = 0$ if $\varphi^k_i \cap \varphi^k_j \neq \emptyset$, for the corrections $p_1, \ldots, p_m$ defined in Definition D.1 we get that

$$\begin{aligned}(\overline{A^k_{\mathbf{y}}}\mathbf{v}^k)_{\text{img}_i} &= \sum_{j \in \overline{\mathcal{I}^k_V}} \mathbf{v}^k_j \sum_{\ell=1}^{6} \Upsilon(\kappa_h(\cdot,\mathbf{y}), \mathcal{T}_k, \ell, i) C_{ij\ell} = \sum_{\ell=1}^{6} \Upsilon(\kappa_h(\cdot,\mathbf{y}), \mathcal{T}_k, \ell, i) \sum_{j \in \overline{\mathcal{I}^k_V}} \mathbf{v}^k_j C_{ij\ell} \\ &= \sum_{\ell=1}^{6} \Upsilon(\kappa_h(\cdot,\mathbf{y}), \mathcal{T}_k, \ell, i) \sum_{t=1}^{m} \mathbf{v}^k_{i+p_t} C_{i,i+p_t,\ell} = \sum_{\ell=1}^{6} \Upsilon(\kappa_h(\cdot,\mathbf{y}), \mathcal{T}_k, \ell, i) \sum_{t=1}^{m} \mathbf{w}_{t,i} C_{i,i+p_t,\ell}.\end{aligned}$$

Since C_{ijk} only depends on the difference $i - j$, for each ℓ the inner sum can be expressed with the same constant for every $i \in \mathcal{I}^k_V$ by the convolution with a $m \times 1 \times 1$ kernel $K^{(\ell)}$, i.e.,

$$\begin{aligned}(\overline{A^k_{\mathbf{y}}}\mathbf{v}^k)_i &= \sum_{\ell=1}^{6} \Upsilon(\kappa_h(\cdot,\mathbf{y}), \mathcal{T}_k, \ell, i) \sum_{t=1}^{m} \mathbf{w}_{t,i} C_{1,1+p_t,\ell} \\ &= \sum_{\ell=1}^{6} \Upsilon(\kappa_h(\cdot,\mathbf{y}), \mathcal{T}_k, \ell, i)(\mathbf{w} *^{\text{sp}} K^{(\ell)})_i = F^k(\mathbf{w}, \Upsilon(\kappa_h(\cdot,\mathbf{y}), \mathcal{T}_k)).\end{aligned}$$

□

Consequently, for $\kappa^k_h \in V^k$ we have that

$$\Upsilon(\kappa^k_h, \mathcal{T}_k, \ell, i) = \sum_{\{j : \operatorname{supp} \varphi^k_j \cap T^\ell_i \neq \emptyset\}} \boldsymbol{\kappa}^k_j \frac{h^2}{3}.$$

Theorem D.2. *For every $\varepsilon, M > 0$ there exists a CNN $\Psi : \mathbb{R}^{7 \times \mathcal{I}^k_U} \to \mathbb{R}^{\mathcal{I}^k_U}$ of constant size consisting of submanifold sparse convolutions such that*

$$\|\Psi - F\|_{L^\infty\left([-M,M]^{7 \times \mathcal{I}^k_U}\right)} \leq \varepsilon,$$

where $F(M^k_{\mathit{img}} \odot T\mathbf{v}^k_{\mathit{img}}, \Upsilon(\kappa, \mathcal{T})) = (\overline{A^k_{\mathbf{y}}}\,\overline{\mathbf{v}^k})_{\mathit{img}}$.

Proof. Exchanging the convolution with the submanifold sparse convolution $*^{\mathsf{sp}}$, the proof works similarly to the proof of [27, Theorem 18] in three steps with the difference that the kernels have width 1 but m channels opposed to one channel and width 3. This is due to the fact that the input images are one image translated in space in different directions m times to account for surrounding information in each node $i \in \mathcal{I}_V^k$ in multiple channels instead of in the surrounding nodes $j \in \overline{\mathcal{I}_V^k}$ close to i. In this way, the sparse convolution $*^{\mathsf{sp}}$ can act on $i \in \mathcal{I}_V^k$ without losing information. In the first step there exists a one-layer CNN realizing the mapping

$$\left(\mathbf{w}, \bar{\boldsymbol{\kappa}}^{(1)}, \dots, \bar{\boldsymbol{\kappa}}^{(6)}\right) \mapsto \left(\mathbf{w} *^{\mathsf{sp}} K^{(\ell)}, \boldsymbol{\kappa}^{(\ell)}\right)_{\ell=1}^{6}.$$

The pointwise multiplication in the second step can be approximated by a CNN of constant size arbitrarily well. Moreover, the addition of the channels in the third step can be realized by a one-layer CNN with a 1×1 kernel as described in the proof of [27, Theorem 18]. Concatenating these CNNs yields the claim. □

In a similar way the following theorem can be proven.

Theorem D.3. *Let $m \in \mathbb{N}$ be as in Definition D.1. There exist kernels $K^{(\ell)} \in \mathbb{R}^{1\times m\times 1\times 1}, \ell = 1, \dots, 6$ such that for*

$$F^{k\mathsf{T}} : \mathbb{R}^{(m+7)\times \mathcal{I}_V^k} \to \mathbb{R}^{\mathcal{I}_V^k}, \quad \left(\mathbf{v}^k_{\mathit{img}}, \bar{\boldsymbol{\kappa}}^{(1)}, \dots, \bar{\boldsymbol{\kappa}}^{(6)}\right) \mapsto \sum_{\ell=1}^{6} \bar{\boldsymbol{\kappa}}^{(\ell)} \odot \left(\mathbf{v}^k_{\mathit{img}} *^{\mathit{sp}} K^{(\ell)}\right)$$

for $\mathbf{v}^k \in \mathbb{R}^{\overline{\mathcal{I}_V^k}}$, $M^k = 1 \in \mathbb{R}^{\mathcal{I}_V^k}$, it holds that

$$F^{k\mathsf{T}}\left(M^k_{\mathit{img}} \odot T\mathbf{v}^k_{\mathit{img}}, \Upsilon(\kappa_h(\cdot, \mathbf{y}), \mathcal{T}_k)\right) = M^k \odot T\left(\overline{A^k_{\mathbf{y}}}^{\mathsf{T}}\overline{\mathbf{v}^k}\right)_{\mathit{img}}.$$

Furthermore, $F^{k\mathsf{T}}$ can be approximated arbitrarily well by a CNN with submanifold sparse convolutions of constant size.

Remark D.1 (CNN for prolongation and weighted restriction)**.** *As noted in [27, Remark 19], the prolongation and weighted restriction can be represented by the application of a 2 strided convolution to the whole image. We now argue that the kernel can also be applied to nonzero entries as in $*^{sp}$ and still be able to represent the operators.*

1 *Weighted restriction: When applying the kernel only to every other entry, which is nonzero in level k, i.e. $\mathcal{I}_V^k$ instead of every other entry in $\mathcal{I}_U^k$, output values can be set to zero, which would include values of entries between the entries used for the convolution. To account for this error, the translation is applied to the image $T\mathbf{v}^k_{\mathit{img}}$ and the operations are applied to the original image on all nonzero indices of the translation.*

2 *Prolongation: The prolongation can be represented as in [27, Remark 19], only acting on nonzero entries of the input images and multiplied with a mask, which is 1 for entries in $\overline{\mathcal{I}_V^k}$ and 0 otherwise.*

Proof of Theorem 5.2. Let $\boldsymbol{\kappa}_{\mathbf{y}}$ be the coefficient image of the interpolation of $\kappa_h(\cdot,\mathbf{y})$ in U^L. The proof works similarly to the proof of [27, Theorem 6]. We write the levelwise local multigrid Algorithm 2 $\mathrm{LLMG} : \bigtimes_{k=1}^{L} \mathbb{R}^{10\times\mathcal{I}_U^k} \to \bigtimes_{k=1}^{L} \mathbb{R}^{\mathcal{I}_U^k}$ as the concatenation of functions, which can be represented or arbitrarily approximated by CNNs. To simplify notation, denote the masked translations as in Definition D.1 by $\tilde{\mathbf{w}}^k := M^k_{\text{img}} \odot T\tilde{\mathbf{u}}^k_{\text{img}}$ and $\bar{\mathbf{w}}^k := M^k_{\text{img}} \odot T\bar{\mathbf{u}}^k_{\text{img}}$, where the mask $M^k := 1 \in \mathbb{R}^{\mathcal{I}_V^k}$ is applied to every channel.

(i) **Integrating the diffusion coefficient.** Let $K \in \mathbb{R}^{1\times 6\times 3\times 3}$ be defined as in [27, Lemma 15(i)] and define

$$f_{in} : \mathbb{R}^{2\times\mathcal{I}_U^L} \to \mathbb{R}^{7\times\mathcal{I}_U^L}, \quad \begin{pmatrix}\boldsymbol{\kappa}\\ \mathbf{f}\end{pmatrix} \mapsto \begin{pmatrix}\boldsymbol{\kappa} * K\\ \mathbf{f}\end{pmatrix}.$$

Then $f_{in}([\boldsymbol{\kappa}_{\mathbf{y}}, \mathbf{f}]) = [\Upsilon(\kappa_h(\cdot,\mathbf{y}), \mathcal{T}_L), \mathbf{f}]$.

(ii) **Smoothing iteration (Line 3, Line 9 in Algorithm 2).** For each level $k = 1,\dots,L$, define the smoothing function

$$f^k_{\text{sm}} : \bigtimes_{\ell=1}^{k} \mathbb{R}^{(4m+7)\times\mathcal{I}_U^\ell} \to \bigtimes_{\ell=1}^{k} \mathbb{R}^{(4m+7)\times\mathcal{I}_U^\ell}$$

by its action on the level k input images in $\mathbb{R}^{(4m+7)\times\mathcal{I}_U^k}$ with $\mathbf{v}, \tilde{\mathbf{v}}, \bar{\mathbf{v}} \in \mathbb{R}^{m\times\mathcal{I}_U^k}, \mathbf{f} \in \mathbb{R}^{\mathcal{I}_U^k}, \bar{\boldsymbol{\kappa}} \in \mathbb{R}^{6\times\mathcal{I}_U^k}$

$$\begin{pmatrix}\mathbf{v}\\ \tilde{\mathbf{v}}\\ \bar{\mathbf{v}}\\ 0\\ \bar{\boldsymbol{\kappa}}\\ \mathbf{f}\end{pmatrix} \mapsto \begin{pmatrix}\mathbf{v} + \omega(\mathbf{f} - [F^k(\mathbf{v}+\tilde{\mathbf{v}}, \bar{\boldsymbol{\kappa}}) + \bar{\mathbf{v}}])\\ \tilde{\mathbf{v}}\\ \bar{\mathbf{v}}\\ 0\\ \bar{\boldsymbol{\kappa}}\\ \mathbf{f}\end{pmatrix}.$$

Except for the operation on $\mathbf{v}$, the other inputs are directly passed to the output. Then, for any $\mathbf{v} \in \bigtimes_{\ell=1}^{k-1} \mathbb{R}^{(4m+7)\times\mathcal{I}_U^\ell}$ Theorem D.1 and since $\tilde{\mathbf{w}}^k_{\ell i} = 0$ for $\varphi^k_{i+p_\ell} \notin V^k$,

$$\begin{aligned} &f^k_{\text{sm}}\left(\begin{bmatrix} T\mathbf{u}^k_{\text{img}}, \tilde{\mathbf{w}}^k, \bar{\mathbf{w}}^k, 0, \Upsilon(\kappa_h(\cdot,\mathbf{y}), \mathcal{T}_k), \mathbf{f}\\ \mathbf{v}\end{bmatrix}\right)\\ &= \begin{bmatrix} T(\mathbf{u}^k + \omega(\mathbf{f} - [\overline{A^k_{\mathbf{y}}}(\overline{\mathbf{u}^k} + \tilde{\mathbf{u}}^k) + \bar{\mathbf{u}}^k|_{\mathcal{I}_V^k}]))_{\text{img}}, \tilde{\mathbf{w}}^k, \bar{\mathbf{w}}^k, 0, \Upsilon(\kappa_h(\cdot,\mathbf{y}), \mathcal{T}_k), \mathbf{f}\\ \mathbf{v}\end{bmatrix}. \end{aligned}$$

In Theorem D.2 it is shown that this operation can be approximated arbitrarily well by a CNN using submanifold sparse convolutions $*^{\text{sp}}$ on level k.

(iii) **Update of $\bar{\mathbf{u}}$ and restriction (Line 5 in Algorithm 2).** We also define the update $\bar{\mathbf{u}}^k$ and the restriction to the coarser level

$$f^k_{\text{upd}}, f_{\text{rest}} : \bigtimes_{\ell=1}^{k} \mathbb{R}^{(4m+7)\times\mathcal{I}_U^\ell} \to \bigtimes_{\ell=1}^{k} \mathbb{R}^{(4m+7)\times\mathcal{I}_U^\ell}.$$

The update function f^k_{upd} is defined by its action on the level k input images

$$\begin{pmatrix}\mathbf{v}\\ \tilde{\mathbf{v}}\\ \bar{\mathbf{v}}\\ 0\\ \bar{\boldsymbol{\kappa}}\\ \mathbf{f}\end{pmatrix} \mapsto \begin{pmatrix}\mathbf{v}\\ \tilde{\mathbf{v}}\\ \bar{\mathbf{v}}\\ \bar{\mathbf{v}} + F^{k^\intercal}(\mathbf{v},\bar{\boldsymbol{\kappa}})\\ \bar{\boldsymbol{\kappa}}\\ \mathbf{f}\end{pmatrix},$$

where again the other inputs are passed to the output as they are. Then, with Theorem D.3 it holds that

$$f^k_{\text{upd}}\left(\begin{bmatrix}\mathbf{u}^k_{\text{img}}, \tilde{\mathbf{w}}^k, \bar{\mathbf{w}}^k, 0, \bar{\boldsymbol{\kappa}}, \mathbf{f}\\ \mathbf{v}\end{bmatrix}\right) = \begin{bmatrix}\mathbf{u}^k_{\text{img}}, \tilde{\mathbf{w}}^k, \bar{\mathbf{w}}^k, M^k_{\text{img}} \odot T(\bar{\mathbf{u}}^k + \overline{A^k_{\mathbf{y}}}^\intercal \overline{\mathbf{u}^k})_{\text{img}}, \bar{\boldsymbol{\kappa}}, \mathbf{f}\\ \mathbf{v}\end{bmatrix}.$$

The operation can be approximated by a CNN due to Theorem D.3. Furthermore, we define the restriction f_{rest} by its action on the inputs on level k and $k-1$ by

$$\begin{pmatrix}\mathbf{v}^k\\ \tilde{\mathbf{v}}^k\\ \bar{\mathbf{v}}^k\\ \mathbf{z}^k\\ \bar{\boldsymbol{\kappa}}^k\\ \mathbf{f}^k\end{pmatrix} \times \begin{pmatrix}\mathbf{v}^{k-1}\\ \tilde{\mathbf{v}}^{k-1}\\ \bar{\mathbf{v}}^{k-1}\\ 0\\ \bar{\boldsymbol{\kappa}}^{k-1}\\ \mathbf{f}^{k-1}\end{pmatrix} \mapsto \begin{pmatrix}\mathbf{v}^{k-1}\\ \tilde{\mathbf{v}}^{k-1}\\ P^\intercal_{k-1}\mathbf{z}^k\\ 0\\ \bar{\boldsymbol{\kappa}}^{k-1}\\ \mathbf{f}^{k-1}\end{pmatrix},$$

where the weighted restriction $P^\intercal$ is applied to every image in $\mathbf{v}^k$. The first to $(k-2)$th inputs are passed to the output unaltered. Then, for any $\mathbf{v}, \tilde{\mathbf{v}}, \bar{\mathbf{v}} \in \mathbb{R}^{m\times\mathcal{I}^k_U}, \mathbf{f} \in \mathbb{R}^{\mathcal{I}^{k-1}_U}, \bar{\boldsymbol{\kappa}} \in \mathbb{R}^{6\times\mathcal{I}^{k-1}_U}, \mathbf{z} \in \mathbb{R}^{\times_{\ell=1}^{k-2}(4m+7)\times\mathcal{I}^\ell_U}$

$$(f^k_{\text{rest}} \circ f^k_{\text{upd}})\left(\begin{bmatrix}T\mathbf{u}^k_{\text{img}}, \tilde{\mathbf{w}}^k, \bar{\mathbf{w}}^k, 0, \bar{\boldsymbol{\kappa}}^k, \mathbf{f}^k\\ \mathbf{v}, \tilde{\mathbf{v}}, \bar{\mathbf{v}}, 0, \bar{\boldsymbol{\kappa}}, \mathbf{f}\\ \mathbf{z}\end{bmatrix}\right) = \begin{bmatrix}\mathbf{v}, \tilde{\mathbf{v}}, \bar{\mathbf{u}}^{k-1}_{\text{img}}, 0, \bar{\boldsymbol{\kappa}}, \mathbf{f}\\ \mathbf{z}\end{bmatrix}.$$

Due to Remark D.1, f^k_{rest} can be represented by a CNN.

(iv) **Update $\tilde{\mathbf{u}}$ with coarse grid solution and prolongation (Line 11 in Algorithm 2).** For the last recursion step, define the prolongation to update the auxiliary vector $\bar{\mathbf{u}}$

$$f^k_{\text{prol}} : \bigtimes_{\ell=1}^{k} \mathbb{R}^{(4m+7)\times\mathcal{I}^\ell_U} \to \bigtimes_{\ell=1}^{k} \mathbb{R}^{(4m+7)\times\mathcal{I}^\ell_U}$$

by its action on input functions from level k and $k-1$

$$\begin{pmatrix}\mathbf{v}^k\\ \tilde{\mathbf{v}}^k\\ \bar{\mathbf{v}}^k\\ \mathbf{z}^k\\ \bar{\boldsymbol{\kappa}}^k\\ \mathbf{f}^k\end{pmatrix} \times \begin{pmatrix}\mathbf{v}^{k-1}\\ \tilde{\mathbf{v}}^{k-1}\\ \bar{\mathbf{v}}^{k-1}\\ \mathbf{z}^{k-1}\\ \bar{\boldsymbol{\kappa}}^{k-1}\\ \mathbf{f}^{k-1}\end{pmatrix} \mapsto \begin{pmatrix}\mathbf{v}^k\\ P_{k-1}(\tilde{\mathbf{v}}^{k-1} + \mathbf{v}^{k-1})\\ \bar{\mathbf{v}}^k\\ \mathbf{z}^k\\ \bar{\boldsymbol{\kappa}}^k\\ \mathbf{f}^k\end{pmatrix} \times \begin{pmatrix}\mathbf{v}^{k-1}\\ \tilde{\mathbf{v}}^{k-1}\\ \bar{\mathbf{v}}^{k-1}\\ \mathbf{z}^{k-1}\\ \bar{\boldsymbol{\kappa}}^{k-1}\\ \mathbf{f}^{k-1}\end{pmatrix},$$

where the prolongation is only applied to indices in $\overline{\mathcal{I}_V^{k-1}}$. For any $(\mathbf{v}^k, \tilde{\mathbf{v}}^k, \bar{\mathbf{v}}^k, \mathbf{z}^k, \bar{\boldsymbol{\kappa}}^k, \mathbf{f}^k) \in \mathbb{R}^{(4m+7)\times\mathcal{I}_U^k}, \mathbf{v}^{k-1}, \mathbf{z}^{k-1} \in \mathbb{R}^{m\times\mathcal{I}_U^{k-1}}, \bar{\boldsymbol{\kappa}}^{k-1} \in \mathbb{R}^{6\times\mathcal{I}_U^{k-1}}, \mathbf{f}^{k-1} \in \mathbb{R}^{\mathcal{I}_U^{k-1}}$ and $z \in \mathbb{R}^{\times_{\ell=1}^{k-2}(4m+7)\times\mathcal{I}_U^\ell}$ it holds that

$$f_{\text{prol}}^k\left(\begin{bmatrix} \mathbf{v}^k, \tilde{\mathbf{v}}^k, \bar{\mathbf{v}}^k, \mathbf{z}^k, \bar{\boldsymbol{\kappa}}^k, \mathbf{f}^k \\ \mathbf{v}^{k-1}, \tilde{\mathbf{w}}^{k-1}, \bar{\mathbf{w}}^{k-1}, \mathbf{z}^{k-1}, \bar{\boldsymbol{\kappa}}^{k-1}, \mathbf{f}^{k-1} \\ \mathbf{z} \end{bmatrix}\right) = \begin{bmatrix} \mathbf{v}^k, \tilde{\mathbf{u}}^k, \bar{\mathbf{v}}^k, \mathbf{z}^k, \bar{\boldsymbol{\kappa}}^k, \mathbf{f}^k \\ \mathbf{v}^{k-1}, \tilde{\mathbf{w}}^{k-1}, \bar{\mathbf{w}}^{k-1}, \mathbf{z}^{k-1}, \bar{\boldsymbol{\kappa}}, \mathbf{f} \\ \mathbf{z} \end{bmatrix}.$$

With Remark D.1 it can be seen that this operation can be represented by a CNN.

(v) **Return solution.** The last required function is the output function

$$f_{\text{out}} : \bigtimes_{\ell=1}^{k} \mathbb{R}^{(4m+7)\times\mathcal{I}_U^\ell} \to \bigtimes_{\ell=1}^{k} \mathbb{R}^{\mathcal{I}_U^\ell}$$

defined for each level $k = 1, \dots, L$ by

$$\begin{pmatrix} \mathbf{v}^k \\ \tilde{\mathbf{v}}^k \\ \bar{\mathbf{v}}^k \\ \mathbf{z}^k \\ \boldsymbol{\kappa}^k \\ \mathbf{f}^k \end{pmatrix} \mapsto \left(\mathbf{v}_0^k\right),$$

where $\mathbf{v}_0^k \in \mathbb{R}^{\mathcal{I}_U^k}$ denotes the first image of $\mathbf{v}^k \in \mathbb{R}^{m\times\mathcal{I}_U^k}$. Then, for $\mathbf{u} \in \mathbb{R}^{\times_{k=1}^{L}\mathcal{I}_V^k}, \mathbf{v} \in \mathbb{R}^{(2m+8)\times\mathcal{I}_U^k}$ $f_{\text{out}}(\bigtimes_{k=1}^{L} T\mathbf{u}_{img}^k \times \mathbf{v}^k) = \bigtimes_{k=1}^{L} \mathbf{u}_{\text{img}}^k$.

Eventually combining the algorithmic components described above, the LLMG for k levels can be expressed as

$$\begin{aligned} \mathrm{LMGV}^k &= f_{\text{sm}}^k \circ f_{\text{prol}}^k \circ (Id_k, LMG^{k-1}) \circ f_{\text{rest}}^k \circ f_{\text{upd}}^k \circ f_{\text{sm}}^k \\ \mathrm{LMGV}^1 &= f_{\text{sm}}^1, \end{aligned}$$

where Id_k passes the the inputs on level $k, \dots, L$ to the output and $\mathrm{LLMG}^m = f_{\text{out}} \circ (\bigcirc_{i=1}^m \mathrm{LMGV}^L) \circ f_{\text{in,res}} \circ f_{\text{in}}$. Here,

$$f_{in,res}(\Upsilon(\kappa_y(\cdot, \mathbf{y}), \mathcal{T}_L), \mathbf{f}) = \left[0, 0, 0, 0, \bar{\kappa}^k\right].$$

Since every component can be approximated by a CNN with constant size, Lemma D.1 implies that the whole algorithm can be approximated by a CNN with constant size. □

D.2 Estimator approximation

The strong residual images and jump images can be approximated on any level by a CNN.

Theorem D.4. *For any $k \in [L]$ and $\varepsilon > 0$ there exists a CNN Ψ such that for all $\boldsymbol{\kappa}^k_{\mathbf{y}\,\textit{img}}, \mathbf{f}^k_{\textit{img}}, \mathbf{u}^k_{\textit{img}} \in [-M, M]^{\mathcal{I}^k_U}$*

$$\left\| \Psi(\mathbf{u}^k_{\textit{img}}, \boldsymbol{\kappa}^k_{\mathbf{y}\,\textit{img}_h}, \mathbf{f}^k_{\textit{img}}) - \left(\mathrm{r}^2_{k,T^q}, \mathrm{j}^2_{k,T^q}\right)_{q=1,2} \right\|_\infty \leq \varepsilon,$$

where the strong residual images and jump images are defined as in Definition 5.1 with respect to the input coefficients. For Ψ, there exist three fixed bias and kernel sizes with width and height at most 5.

Proof. We first note that for $q = 1, 2$ and each triangle $T = T^q_{k,i}$ as illustrated in Figure 5.1,

$$\begin{aligned}(\mathrm{r}^2_{k,T^q})_i &= \|f_h + \nabla \cdot (\kappa_h \nabla u_h)\|^2_{L^2(T)} \\ &= \|f_h\|^2_{L^2(T)} + 2\,\langle f_h, \nabla \cdot (\kappa_h \nabla u_h)\rangle_{L^2(T)} + \|\nabla \cdot (\kappa_h \nabla u_h)\|^2_{L^2(T)}\,.\end{aligned}$$

Here,

$$\begin{aligned}\|f_h\|^2_{L^2(T)} &= \sum_{m,n \in \text{nodes}(T)} \mathbf{f}_m \mathbf{f}_n e_{mn} && \text{for}\quad e_{mn} := \int_T \varphi_m \varphi_n \mathrm{d}x, \\ \langle f_h, \nabla \cdot (\kappa_h \nabla u_h)\rangle_{L^2(T)} &= \sum_{j \in \text{nodes}(T)} \mathbf{f}_j c_j \sum_{m,n \in \text{nodes}(T)} \boldsymbol{\kappa}_m \mathbf{u}_n d_{mn} && \text{for}\quad c_j := \int_T \varphi_j \mathrm{d}x, \quad d_{mn} := \nabla\varphi_m \cdot \nabla\varphi_n, \\ \|\nabla \cdot (\kappa_h \nabla u_h)\|^2_{L^2(T^q)} &= \left(\sum_{m,n \in \text{nodes}(T)} \boldsymbol{\kappa}_m \mathbf{u}_n d_{mn}\right)^2 \int_T \mathrm{d}x.\end{aligned}$$

Note that c_m, d_{jm} and $\int_T \mathrm{d}x$ are independent of the node i, i.e., $d_{mn} = d_{m-i,n-i}$, $c_j = c_{j-i}$, $e_{mn} = e_{m-i,n-i}$. Furthermore,

$$(\mathrm{j}^2_{k,T^q})_i = \|[\![\kappa_h \nabla u_h]\!]\|^2_{L^2(\partial T^{(q)})} = \sum_{K \in \text{edges}(T)} \|\kappa_h\|^2_{L^2(K)}\, [\![u_h]\!]^2_{(K)}.$$

With triangle $\tilde{T}_K$ such that $K \in$ edges$(\tilde{T}_K)$ for $K \in$ edges(T),

$$\begin{aligned}\|\kappa_h\|^2_{L^2(K)} &= \frac{\sqrt{2}h}{3} \sum_{j,m \in \text{nodes}(K)} \boldsymbol{\kappa}_j \boldsymbol{\kappa}_m, \\ ([\![u_h]\!]_{(K)})^2 &= \left(\frac{1}{h} \sum_{j \in \text{nodes}(\tilde{T}) \cup \text{nodes}(T)} (-1)^{1-\chi_K(j)} \mathbf{u}_j\right)^2.\end{aligned}$$

Since shifting and addition can be represented by convolutional kernels, multiplication and squaring can be approximated by the concatenation of a convolutional kernel, the application of the activation function and another convolutional kernel, every operation can be represented or approximated by a CNN. Since the addition and shifting always only includes nodes in the vicinity of the considered node, the kernels have a small bounded width and height. □

D.3 Adaptive FEM approximation

The solver and the estimator are combined with a marking strategy and a refinement implemented with masks to arrive at a CNN approximation of the whole AFEM Algorithm 1.

Proof of Theorem 5.4. To show that Algorithm 1 can be approximated entirely, the required steps are considered separately. Starting with $V = U^1$ and $\mathbf{u} = 0$, the algorithm consist of the following steps.

(i) **Update $\mathbf{u} \leftarrow \mathbf{u} + \mathbf{v}$.**

By Corollary 5.1, the solution to $A_{\mathbf{y}}\mathbf{v} = \mathbf{f} - A_{\mathbf{y}}\mathbf{u}$ can be approximated up to any ε_{sol} by a CNN Ψ_{sol} with input images $\boldsymbol{\kappa}_{\mathbf{y}_{\text{img}}}, (\mathbf{f} - A_{\mathbf{y}}\mathbf{u})_{\text{img}}$ and number of parameters bounded by $M(\Psi_{\text{sol}}) \lesssim L\log(\varepsilon_{\text{sol}}^{-1})/\log(c_L^{-1})$. Using Lemma D.1, for any $\varepsilon_{\text{cor}} > 0$ there exists a CNN Ψ_{cor} with parameters bounded by $M(\Psi_{\text{cor}}) \lesssim L\log(\varepsilon_{\text{cor}}^{-1})/\log(c_L^{-1})$ such that

$$\left\|\Psi_{\text{cor}}\left(\bigtimes_{k=1}^{L}(\mathbf{u}_{\text{img}}^k, \tilde{\mathbf{u}}_{\text{img}}^k, \bar{\mathbf{u}}_{\text{img}}^k, \Upsilon(\kappa_h, \mathcal{T}_k), \mathbf{f}^k)\right) - (\mathbf{u}+\mathbf{v})\right\|_\infty \leq \varepsilon_{\text{cor}}.$$

(ii) **Local error estimator η_T^2.**

The error estimator for the updated solution $\mathbf{u}$ can be approximated with Theorem 5.3. For any $\varepsilon_{\text{eta}} > 0$ there exists a CNN Ψ_{est} with number of parameters bounded by $M(\Psi_{\text{est}}) \lesssim L$ such that

$$\left\|\Psi_{\text{est}}\left(\bigtimes_{k=1}^{L}(\mathbf{u}_{\text{img}}^k, \boldsymbol{\kappa}_{\mathbf{y}_{\text{img}}}^k, \mathbf{f}_{\text{img}}^k)\right)[\ell] - \eta_\ell^2\right\|_\infty \leq \varepsilon_{\text{eta}}. \tag{D.1}$$

(iii) **Marking.**

Different marking strategies can be considered. Here, a threshold marking strategy as in Definition 2.3 is used. The operator mapping estimator images to marker images inside a CNN is defined by the mapping $\eta_k^2 \in \mathbb{R}^{2\times\mathcal{I}_U^k} \mapsto M^k \in \{0,1\}^{2\times\mathcal{I}_U^k}$ on each level $k = 1,\dots,L$, where for $q = 1,2$ $M^k[q]_i = 1$ if $\eta_k^2[q]_i > \delta_k$ and $M^k[q]_i = 0$ otherwise for $i \in \mathcal{I}_U^k$. Let the CNN Ψ_{mark} be composed of a convolutional layer subtracting the threshold and approximation error of the error estimator $\delta_k - \varepsilon_{\text{eta}}$ followed by the application of a heavyside activation function $h_{0,1}$. It then maps the approximated error estimator $\tilde{\eta}_k^2$ to marker images $\Psi_{\text{mark}}(\tilde{\eta}_k^2) \in \{0,1\}^{2\times\mathcal{I}_U^k}$, such that the inequality

$$M^k \leq \Psi_{\text{mark}}(\tilde{\eta}_k^2)$$

holds entrywise. This can be derived by considering that $M^k[q]_i = 1$ implies $\eta_k^2[q]_i > \delta_k$ and $\tilde{\eta}_k^2[q]_i \geq \eta_k^2[q]_i - \varepsilon_{\text{eta}}$ holds with (D.1). This yields $\tilde{\eta}_k^2[q]_i - (\delta_k - \varepsilon_{\text{eta}}) > 0$ and therefore $\Psi_{\text{mark}}(\tilde{\eta}_k^2)[q]_i = h_{0,1}(\tilde{\eta}_k^2[q] - (\delta_k - \varepsilon_{\text{eta}})) = 1$ for $q = 1,2$ and $i \in \mathcal{I}_U^k$ for $k = 1,\dots,L$. Therefore, all triangles marked with the threshold marking strategy with the true estimator of the true solution are also marked by the CNN. As discussed in Remark 5.1, other global marking strategies such as Dörfler marking could be implemented outside of the network. In each step the chosen strategy has to incorporate the error in the estimator approximation. For instance, in addition to the markings based on the selected strategy and the approximated estimator, one could mark all triangles for which the approximated error estimator is larger than the lowest approximated estimator of the already marked triangles minus twice the estimated approximation error.

(iv) **Refinement.**

The refinement is incorporated in the CNN using the marking masks $M^k \in \{0,1\}^{2\times\mathcal{I}_U^k}$ on each level, corresponding to piecewise constant functions discretized as in (4.1). These are mapped to a mask $M_V^{k+1} \in \{0,1\}^{\mathcal{I}_U^{k+1}}$ corresponding to continuous piecewise linear functions as in (3.1). To mimic the refinement, they should fulfill $(M_V^{k+1})_i = 1$ if $\operatorname{supp}\varphi_i^{k+1} \cap T_{kj}^q \neq \emptyset$ for some $q = 1,2$, $j \in \mathcal{I}_U^k$ with $M^k[q]_j = 1$ as described in Section 2.4.

This mapping can be constructed in two steps. First, M^k is mapped to some $\bar{M}^{k+1} \in \mathbb{R}^{\mathcal{I}_U^{k+1}}$, which is > 0 on on the nodes corresponding to the required φ_i^{k+1} and zero otherwise. This can be done by applying a transpose convolution with stride 2 and a kernel of size $2\times 1\times 3\times 3$. Secondly, the heaviside function $h_{0,1}$ can be applied entrywise to arrive at the desired $0/1$-masks M_V^{k+1}.

The derived function spaces satisfy $\tilde{V}^k \supset V^k$ since the approximate marking covers the exact marking based on the true error estimator of the true Galerkin solution on the current space.

We can now combine all above estimations for the components of the AFEM. Concatenating these steps into one CNN as in Lemma D.1 leads to a CNN Ψ such that

$$\begin{aligned}
&\left\|\mathcal{C}\left(\Psi\left(\bigtimes_{k=1}^{L}(\mathbf{u}_{\text{img}}^k, \tilde{\mathbf{u}}_{\text{img}}^k, \bar{\mathbf{u}}_{\text{img}}^k, \Upsilon(\kappa_h, \mathcal{T}_k), \mathbf{f}^k)\right)\right) - u\right\|_{H_0^1} \\
&\leq \left\|\mathcal{C}\left(\Psi\left(\bigtimes_{k=1}^{L}(\mathbf{u}_{\text{img}}^k, \tilde{\mathbf{u}}_{\text{img}}^k, \bar{\mathbf{u}}_{\text{img}}^k, \Upsilon(\kappa_h, \mathcal{T}_k), \mathbf{f}^k)\right)\right) - \mathcal{C}(\mathbf{u}_{\tilde{V}})\right\|_{H_0^1} + \|\mathcal{C}(\mathbf{u}_{\tilde{V}}) - u\|_{H_0^1} \\
&\leq \varepsilon_{\text{cor}} + \|\mathcal{C}(\mathbf{u}_V) - u\|_{H_0^1} \\
&= \varepsilon_{\text{cor}} + \left\|\mathrm{AFEM}(U^1, L) - u\right\|_{H_0^1},
\end{aligned}$$

where $\mathbf{u}_W$ is defined as the coefficients of the Galerkin projection of $u(\cdot, \mathbf{y})$ onto W for some $W \subset H_0^1$, V is the space constructed by the AFEM after $L \in \mathbb{N}$ steps and $\tilde{V}$ is the the space constructed by the CNN after the same number of steps. □