Technical Report Series
Center for Data and Simulation Science

Alexander Heinlein, Axel Klawonn, Martin Lanser, Janine Weber

Combining Machine Learning and Adaptive Coarse Spaces - A Hybrid Approach for Robust FETI-DP Methods in Three Dimensions

Technical Report ID: CDS-2020-1
Available at `https://kups.ub.uni-koeln.de/id/eprint/11285`
Submitted on June 16, 2020

COMBINING MACHINE LEARNING AND ADAPTIVE COARSE SPACES – A HYBRID APPROACH FOR ROBUST FETI-DP METHODS IN THREE DIMENSIONS

ALEXANDER HEINLEIN[*][†], AXEL KLAWONN[*][†], MARTIN LANSER[*][†], AND JANINE WEBER[*]

June 16, 2020

Abstract. The hybrid ML-FETI-DP algorithm combines the advantages of adaptive coarse spaces in domain decomposition methods and certain supervised machine learning techniques. Adaptive coarse spaces ensure robustness of highly scalable domain decomposition solvers, even for highly heterogeneous coefficient distributions with arbitrary coefficient jumps. However, their construction requires the setup and solution of local generalized eigenvalue problems, which is typically computationally expensive. The idea of ML-FETI-DP is to interpret the coefficient distribution as image data and predict whether an eigenvalue problem has to be solved or can be neglected while still maintaining robustness of the adaptive FETI-DP method. For this purpose, neural networks are used as image classifiers. In the present work, the ML-FETI-DP algorithm is extended to three dimensions, which requires both a complex data preprocessing procedure to construct consistent input data for the neural network as well as a representative training and validation data set to ensure generalization properties of the machine learning model. Numerical experiments for stationary diffusion and linear elasticity problems with realistic coefficient distributions show that a large number of eigenvalue problems can be saved; in the best case of the numerical results presented here, 97 % of the eigenvalue problems can be avoided to be set up and solved.

Key words. ML-FETI-DP, FETI-DP, machine learning, domain decomposition methods, adaptive coarse spaces, finite elements

AMS subject classifications. 65F10, 65N30, 65N55, 68T05

1. Introduction.

Domain decomposition methods are highly scalable, iterative solvers for the solution of large systems of equations such as arriving, e.g., from the discretization of partial differential equations by finite elements. Among the most commonly used domain decomposition methods are the FETI-DP (Finite Element Tearing and Interconnecting - Dual Primal) [12, 11, 48, 49], BDDC (Balancing Domain Decomposition by Constraints) [7, 8, 51, 53, 52] and overlapping Schwarz [61] methods. All of these methods have successfully been applied to a wide range of problems and have been shown to be parallel scalable up to hundred of thousands of cores and beyond [26, 62, 2, 1, 39, 44, 40, 38, 37, 27]. In the present article, we are mostly interested in the solution of highly heterogeneous stationary diffusion or linear elasticity problems with high contrasts in the material distributions. For such cases, including those with arbitrary jumps in the diffusion coefficient or the Young modulus, the convergence rate of standard domain decomposition methods typically deteriorates severely. In particular, the classic condition number bounds for standard domain decomposition methods are only valid under relatively restrictive assumptions concerning the coefficient function or the material distribution. Thus, in case of highly complex coefficient functions, coarse space enhancements are necessary to guarantee a robust algorithm while retaining a good condition number bound to preserve numerical scalability. For FETI-DP algorithms considered in the present

[*] Department of Mathematics and Computer Science, University of Cologne, Weyertal 86-90, 50931 Köln, Germany, alexander.heinlein@uni-koeln.de, axel.klawonn@uni-koeln.de, martin.lanser@uni-koeln.de, janine.weber@uni-koeln.de, url: http://www.numerik.uni-koeln.de

[†] Center for Data and Simulation Science, University of Cologne, Germany, url: http://www.cds.uni-koeln.de

This manuscript is for review purposes only.

article, such a condition number bound would only depend polylogarithmically on the size of the subproblems while being independent of the contrast of the values of the relevant coefficient functions. To accomplish this in three dimensions for material discontinuities aligned with the interface between subdomains, additional averages along edges and first order moments to enhance the coarse space have been proposed in [48, 49]. This approach has been heuristically extended in [43] for material discontinuities not aligned with the interface using certain weighted averages. We have further generalized this technique in [23]. Here, we have integrated generalized weighted averages along faces and/or edges into the coarse space. Using this approach as a default can help to make FETI-DP and BDDC algorithms more robust for a number of realistic application problems. However, all aforementioned approaches are generally not robust for arbitrary coefficient functions, e.g., with numerous discontinuities along and across the interface between subdomains. Hence, adaptive coarse spaces have been developed for different domain decomposition algorithms [4, 35, 34, 33, 58, 57, 3, 6, 54, 55, 42, 41, 31, 13, 14, 10, 9, 59, 60, 19, 18, 20]. In adaptive coarse spaces, eigenvectors originating from the solution of certain local generalized eigenvalue problems on parts of the interface, e.g., faces or edges, are used to enhance the coarse space. For these techniques, condition number bounds that are robust with respect to arbitrary coefficient distributions can be proven. In particular, the resulting condition number bounds only depend on a user-defined tolerance, which is used as a threshold for the selection of the eigenvectors based on their corresponding eigenvalues, and on geometrical constants. As a drawback, in a parallel implementation, the setup and the solution of the eigenvalue problems take up a significant amount of time; cf. [50, 36]. However, for many realistic coefficient distributions, a large number of eigenvalue problems which are not necessary for a robust convergence of the adaptive domain decomposition method is solved; they are not necessary in the sense that no corresponding eigenvectors are being selected. In order to account for this, in [24], we introduced the concept of training a neural network to make an automatic decision whether the solution of a specific local eigenvalue problem is necessary for robustness. In particular, we have focused on the two-dimensional case and the adaptive coarse space introduced in [54] for the FETI-DP method, which is based on local eigenvalue problems on edges between neighboring subdomains; see [42] for the first theoretical proof of a robust condition number bound for this algorithm. We were able to significantly reduce the number of necessary eigenvalue problems on edges by using samples of the coefficient function in the adjacent subdomains as input for the machine learning model; in [25], we have shown that it is actually sufficient to sample in neighborhoods close to the edges. Throughout this paper, we denote the algorithm combining adaptive FETI-DP and machine learning introduced in [24] by ML-FETI-DP.

Here, we extend the two-dimensional ML-FETI-DP approach [24] to three-dimensional problems. The main concept is very similar, however, the interface between the adaptive FETI-DP algorithm and the neural network, i.e., the preprocessing of the input data for the neural network, requires substantial modifications and enhancements. In particular, handling faces of three-dimensional unstructured domain decompositions, e.g., obtained from METIS [30], is much more complex compared to edges in two dimensions. Moreover, in the generation of training and validation data, we use METIS domain decompositions and adapt the generation of coefficient distributions based on randomization as described for two dimensions in [24] to three dimensions. The main focus of this paper is thus on the description of the preprocessing of the data. As a machine learning approach, we will again - as in [24] -

employ feedforward neural networks with Rectified Linear Unit (ReLU) activation and dropout layers.

The remainder of this paper is organized as follows. First, we introduce our model problems, namely stationary diffusion problems and linear elasticity problems. Second, we briefly recapitulate the FETI-DP domain decomposition method and the specific adaptive coarse space approach [33, 54, 55] for three dimensions. Afterwards, we present our machine learning based approach ML-FETI-DP using feedforward neural networks. We then describe the preprocessing of the input data and how we generate appropriate training and validation data for the training process of the neural network. Finally, we show numerical results for different relevant elliptic problems. At first, we consider a coefficient distribution with five balls of different radii in the unit cube. Second, we consider an RVE (Representative Volume Element) representing the microstructure of a dual-phase steel.

2. Model problems and adaptive FETI-DP domain decomposition algorithms.

In this section, we give a brief introduction of our model problems and shortly describe the classic FETI-DP domain decomposition method [12, 11, 48, 49]. Finally, in subsection 2.3.2, we describe the employed adaptive FETI-DP coarse space technique for three dimensions; see [33, 54, 55].

2.1. Diffusion, elasticity, and finite elements.

As a first model problem, we consider a stationary, linear, scalar diffusion problem with a highly heterogeneous coefficient function $\rho : \Omega \to \mathbb{R}$ and homogeneous Dirichlet boundary conditions on $\partial\Omega$. Thus, the model problem in its variational form can be written as: find $u \in H_0^1(\Omega)$ such that

$$\int_\Omega \rho \nabla u \cdot \nabla v \ dx = \int_\Omega f \ v \ dx \ \forall v \in H_0^1(\Omega). \tag{2.1}$$

Various examples of coefficient functions are discussed in section 4.

As a second model problem, we consider the equations of linear elasticity. It consists of finding the displacement $\mathbf{u} \in \mathbf{H}_0^1(\Omega) := (H_0^1(\Omega))^3$ such that

$$\int_\Omega G \ \varepsilon(\mathbf{u}) : \varepsilon(\mathbf{v}) \ d\mathbf{x} + \int_\Omega G\beta \ \mathrm{div}\mathbf{u} \ \mathrm{div}\mathbf{v} \ d\mathbf{x} = \int_\Omega \mathbf{f}^T \mathbf{v} \ d\mathbf{x} \tag{2.2}$$

for all $\mathbf{v} \in \mathbf{H}_0^1(\Omega)$, given material functions $G : \Omega \to \mathbb{R}$ and $\beta : \Omega \to \mathbb{R}$, and a volume force $f \in (L^2(\Omega))^3$.

By a finite element discretization of (2.1) and (2.2) on Ω, we obtain the respective linear system of equations

$$K_g u_g = f_g. \tag{2.3}$$

We denote the finite element space by V^h and we have u_g, $f_g \in V^h$. Note that, throughout this paper, we assume that the coefficient functions ρ, G, and β are constant on each finite element but may have large jumps from element to element. For simplicity, in the present article, we only consider tetrahedrons and linear finite elements.

2.2. Standard FETI-DP.

Let us briefly describe the classic FETI-DP method and introduce some necessary notation.

2.2.1. Domain decomposition. We assume a decomposition of Ω into $N \in \mathbb{N}$ nonoverlapping subdomains $\Omega_i,\ i = 1, ..., N$, i.e., $\Omega = \bigcup_{i=1}^{N} \Omega_i$. Each of the subdomains is a union of finite elements. The finite element subspaces associated with $\Omega_i,\ i = 1, ..., N$, are denoted by $W_i,\ i = 1, ..., N$. We obtain local finite element problems $K^{(i)}\, u^{(i)} = f^{(i)}$ with $K^{(i)} : W_i \to W_i$ and $f^{(i)} \in W_i$ by restricting the considered differential equation (2.1) or (2.2) to Ω_i and discretizing its variational formulation in the finite element space W_i. Let us remark that the matrices $K^{(i)}$ are, in general, not invertible for subdomains, which have no contact to the Dirichlet boundary.

We introduce the simple restriction operators $R_i : V^h \to W_i,\ i = 1, ..., N$, the block vectors $u^T := \left(u^{(1)T}, ..., u^{(N)T}\right)$ and $f^T := \left(f^{(1)T}, ..., f^{(N)T}\right)$, and the block matrices $R^T := \left(R_1^T, ..., R_N^T\right)$ and $K = \operatorname{diag}\left(K^{(1)}, ..., K^{(N)}\right)$. We then obtain the identities

$$K_g = R^T K R \quad \text{and} \quad f_g = R^T f. \tag{2.4}$$

The application of R^T in (2.4) thus has the effect of a finite element assembly process of local finite element functions on the interface $\Gamma := \left(\bigcup_{i=1}^{N} \partial\Omega_i\right) \setminus \partial\Omega$.

The block matrix K is not invertible as soon as a single subdomain has no contact to the Dirichlet boundary. Therefore, the system $Ku = f$ has no unique solution and, more precisely, an unknown vector u might be discontinuous on the interface. We now proceed to describe how the solution u_g is obtained using FETI-DP, i.e., how the continuity of $u \in W := W_1 \times ... \times W_N$ on the interface is enforced.

2.2.2. The FETI-DP saddle point system. Let us assume, we have sorted and decomposed an unknown vector u from the product space W into interface variables u_Γ and all remaining interior variables u_I, i.e., $u^T = \left(u_I^T, u_\Gamma^T\right) \in W$. We further subdivide the degrees of freedoms on the interface u_Γ into primal variables u_Π and dual variables u_Δ. Throughout this paper, we select all subdomain vertices to be primal. Continuity in the primal variables is enforced by a finite element assembly process, while continuity in the dual variables is enforced iteratively by Lagrange multipliers.

For the primal assembly process we introduce the operator R_Π^T, which is similar to R^T, but assembles only in primal variables. We denote the corresponding primally assembled finite element space by $\widetilde{W}$. Thus, we have $R_\Pi : \widetilde{W} \to W$ and any $\tilde{u} \in \widetilde{W}$ is of the structure $\tilde{u}^T = \left(u_I^T, u_\Delta^T, \tilde{u}_\Pi^T\right)$, where $\tilde{u}_\Pi$ is now a vector of global variables. The vector $\tilde{u}_\Pi$ can also be seen as a coarse solution and the corresponding Schur complement system in the primal variables constitutes the global coarse problem or second level problem in FETI-DP. We define primally coupled operators by

$$\tilde{f} = R_\Pi^T f \quad \text{and} \quad \widetilde{K} = R_\Pi^T K R_\Pi. \tag{2.5}$$

Let us remark that $\widetilde{K} : \widetilde{W} \to \widetilde{W}$ will be a regular matrix if enough primal constraints are chosen.

Enforcing continuity in the dual variables is done by the constraint $B\tilde{u} = 0$, using a linear jump operator $B = [B^{(1)}, ..., B^{(N)}]$; see, e.g., [44] for a detailed definition of B. Each row of B evaluates the jump between two degrees of freedom on the interface belonging to the same physical node but different subdomains. Thus, each row of B contains exactly one 1 and one -1 and the remaining entries are zero. Enforcing the jump condition via Lagrange multipliers, we obtain the FETI-DP saddle point system

$$\begin{pmatrix} \widetilde{K} & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \tilde{u} \\ \lambda \end{pmatrix} = \begin{pmatrix} \tilde{f} \\ 0 \end{pmatrix}, \tag{2.6}$$

where λ is the vector of the Lagrange multipliers. Let us remark that, due to the constraint $B\tilde{u} = 0$, the solution $\tilde{u} \in \widetilde{W}$ is continuous on the interface Γ.

2.2.3. Iterative solution of the FETI-DP system.

By a block elimination in (2.6) we derive the system

$$F\lambda = d \tag{2.7}$$

with $F = B\widetilde{K}^{-1}B^T$ and $d = B\widetilde{K}^{-1}\tilde{f}$. Equation (2.7) is solved iteratively with a preconditioned CG or GMRES approach using an additional Dirichlet preconditioner M_D^{-1}; see also [61]. The preconditioner M_D^{-1} is a weighted sum of local Schur complements in the dual variables. Let

$$S_{\Delta\Delta}^{(i)} = K_{\Delta\Delta}^{(i)} - K_{\Delta I}^{(i)} K_{II}^{(i)-1} K_{\Delta I}^{(i)T} \tag{2.8}$$

be the Schur complement of $K^{(i)}, i = 1, ..., N$ in the dual variables and

$$B_D = \left(D^{(1)T} B^{(1)}, ..., D^{(N)} B^{(N)T} \right) \tag{2.9}$$

a scaled jump matrix, where the scaling matrices $D^{(i)}$, $i = 1, ..., N$ are usually defined by the PDE coefficients. We further define $B_{D,\Delta}$, the restriction of B_D to the dual variables, and $S_{\Delta\Delta} = \text{diag}(S_{\Delta\Delta}^{(1)}, ..., S_{\Delta\Delta}^{(N)})$. The preconditioner is then defined by

$$M_D^{-1} = B_{D,\Delta} S_{\Delta\Delta} B_{D,\Delta}^T. \tag{2.10}$$

The application of M_D^{-1} is embarrassingly parallel due to the block diagonal structure of S. The desired solution $\tilde{u}$ is finally obtained by solving

$$\widetilde{K}\,\tilde{u} = \tilde{f} - B^T\lambda. \tag{2.11}$$

2.2.4. Condition number bound.

For scalar elliptic partial differential equations as well as for linear elasticity problems, the classic polylogarithmic condition number bound

$$\kappa(M_D^{-1}F) \leq C \left(1 + \log\left(\frac{H}{h}\right)\right)^2 \tag{2.12}$$

holds, with C independent of H and h; see, e.g., [47, 49, 48]. In (2.12) H is the maximum diameter over all subdomains, h the maximum diameter over all finite elements, and thus H/h is a measure for the number of finite elements per subdomain. In general, different coefficient distributions in two and three dimensions can be captured by different coarse spaces and scalings in the preconditioner M_D^{-1}. Please note that in three dimensions additional coarse constraints based on averages over edges and/or faces are necessary to retain the same logarithmic condition number bound as in (2.12); see, e.g., [48, 49] and for experimental results [12, 46].

However, for completely arbitrary and complex coefficient distributions, (2.12) does not hold anymore. In recent years, adaptive coarse spaces have been developed to overcome this limitation [4, 35, 34, 33, 58, 57, 3, 6, 54, 55, 42, 41, 31, 13, 14, 10, 9, 59, 60]. In these algorithms, local eigenvalue problems on parts of the interface, i.e., edges or faces, are solved and selected eigenvectors are used to construct appropriate adaptive constraints. The FETI-DP coarse space is then enriched with these additional primal constraints in the setup phase before the iterative solution phase starts.

2.3. Adaptive FETI-DP in three dimensions.

2.3.1. Enhancing the coarse space with additional constraints. In general, different approaches to implement coarse space enrichments for the FETI-DP method exist. The two most common approaches are a deflation or balancing approach [45, 42] and a transformation of basis approach [48, 44]. In the present paper, we always use the balancing preconditioner to enhance the coarse space with additional constraints, regardless if adaptive or frugal constraints are enforced; see subsection 2.3.2 and subsection 2.4, respectively. Please see [24, Sec. 2.3.1] or [45, 42] for a detailed description of the deflation and balancing approach.

2.3.2. The adaptive constraints. The main idea of adaptive coarse spaces in domain decomposition methods is to enrich the FETI-DP or BDDC coarse space with additional primal constraints, obtained by solving certain local generalized eigenvalue problems on faces or edges, before the iteration starts. In the following, we give a brief description of the algorithm considered in [33, 55] for the convenience of the reader. In three dimensions, for certain equivalence classes $\mathcal{X}_{ij}$, i.e., faces F_{ij} or edges E_{ij}, we thus solve the generalized eigenvalue problem

$$\langle P_{D_{ij}} v_{ij},\ S_{ij} P_{D_{ij}} w_{ij} \rangle = \mu_{ij} \langle v_{ij},\ S_{ij} w_{ij} \rangle \quad \forall v_{ij} \in (\operatorname{Ker} S_{ij})^{\perp}\,;$$

see [33]. Note that, in this approach, we only have to solve edge eigenvalue problems on edges that belong to more than three subdomains.

Here, we define $S_{ij} = \operatorname{diag}(S_i, S_j)$ and $P_{D_{ij}} = B_{D,\mathcal{X}_{ij}}^T B_{\mathcal{X}_{ij}}$ as a local version of the jump operator $P_D = B_D^T B$ with $B_{\mathcal{X}_{ij}} = \left(B_{\mathcal{X}_{ij}}^{(i)}, B_{\mathcal{X}_{ij}}^{(j)} \right)$ and $B_{D,\mathcal{X}_{ij}} = \left(B_{D,\mathcal{X}_{ij}}^{(i)}, B_{D,\mathcal{X}_{ij}}^{(j)} \right)$. We then select all eigenvalues $\mu_{ij} \geq \text{TOL}$ for a user-defined tolerance TOL and use the corresponding eigenvectors $v_{\mathcal{X}_{ij}}$ to automatically design the coarse space. New coarse components then enforce the constraint $(B_{D,\mathcal{X}_{ij}} S_{ij} P_{D_{ij}} w_{\mathcal{X}_{ij}})^T B_{\mathcal{X}_{ij}} v_{\mathcal{X}_{ij}} = 0$ in each iteration, e.g., with projector preconditioning or transformation of basis. When enforcing these constraints for all faces and edges between subdomains, that do not share a face, we obtain the condition number estimate:

$$\boldsymbol{\kappa}(M^{-1}K) \leq 4 \max\{N_{\mathcal{F}}, N_{\mathcal{E}} M_{\mathcal{E}}\}^2 TOL; \tag{2.13}$$

see [33] for the proof. Here, we denote by $N_{\mathcal{F}}$ the maximum number of faces of a subdomain, by $N_{\mathcal{E}}$ the maximum number of edges of a subdomain, and by $M_{\mathcal{E}}$ the maximum multiplicity of an edge. The condition number bound thus only depends on geometrical constants of the domain decomposition and, in particular, not on the contrast of the coefficient function. The choice of the tolerance value is user-dependent and should be selected with reference to the spectral gap of the eigenvalues of the preconditioned solver; see also [28].

Let us note that the first rigorous proof for the condition number bound was given in [42] for two dimensions and was extendend to three dimensions in [33] with the additional use of edge eigenvalue problems. In [54, 55] a condition number indicator was presented for the first time, both for two and three dimensions.

2.4. Frugal constraints. As in [24] for the two-dimensional case, we here also consider frugal constraints [23] on certain faces. Frugal constraints build a coarse space which is robust for many diffusion or elasticity problems with jumps in the coefficient function, and the setup of the frugal coarse space is rather cheap compared with the setup of adaptive coarse spaces. This is due to the absence of any local

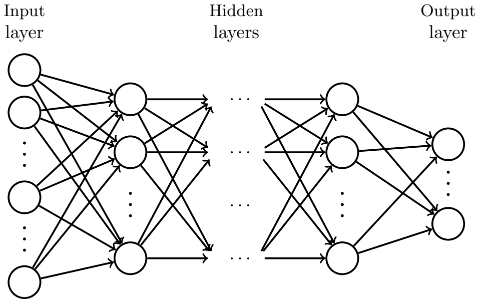

FIG. 1. *Structure of a dense feed forward neural network with several hidden layers.*

eigenvalue problems in the computation of the constraints; see [23] for a detailed discussion of frugal constraints. Let us remark that, for each face or edge, a single frugal constraint can be computed in case of a diffusion problem, but up to six in case of a linear elasticity problem. Nevertheless, for many coefficient distributions, using frugal constraints exclusively is not sufficient to obtain robustness, and hence, the frugal coarse space has to be enriched by additional adaptive constraints on some specific faces and/or edges. The combination of both coarse spaces, i.e., the adaptive and the frugal coarse space, and how we can exploit the benefits of both approaches using machine learning in the ML-FETI-DP approach, is elaborated later; cf. subsections 3.3 and 4.3.

3. Selecting necessary eigenvalue problems using machine learning.

Here, we extend our approach from [24] to three-dimensional problems. In [24], we have used machine learning techniques to predict which eigenvalue problems are necessary for robustness of highly heterogeneous two-dimensional linear diffusion and elasticity problems. This approach is based on the observation that, for many realistic problems with highly heterogeneous coefficient distributions, a large share of the eigenvalue problems will not yield large eigenvalues $\mu_{ij} > TOL$. In particular, the number of large eigenvalues that correspond to an equivalence class $\mathcal{X}_{ij}$ can be attributed to the local part of the coefficient function in the adjacent subdomains. Hence, in two dimensions, we used a sampling procedure, to construct, for each edge, an image representation of the coefficient function in the two adjacent subdomains; cf. [24, Sect. 3.1] and Figure 2 (left). By 'sampling procedure', we understand a specific sequence of evaluations of the coefficient function. Using the resulting image representation as input, a machine learning model was trained to classify whether the corresponding eigenvalue problems yield a large eigenvalue or not. Then, only those eigenvalue problems that are classified as necessary are solved. As already mentioned in the introduction, we denote this hybrid algorithm which combines the adaptive FETI-DP algorithm and a machine learning model as ML-FETI-DP. As an extension to this binary classification, we also introduced a three-class model, which reduces the number of computed eigenvalues even further by additionally using frugal constraints; cf. [23], [24, Sect. 3.3], as well as subsections 2.4 and 3.3.

As described in subsection 2.3.2, edge as well as face eigenvalue problems have to be solved for robustness in three dimensions. However, the edges typically only possess a relatively small number of nodes, and hence, the corresponding eigenvalue problems are rather small; cf. [21]. Therefore, in our three-dimensional approach, we restrict ourselves to the identification of necessary face eigenvalue problems and solve all eigenvalue problems for edges which belong to more than three subdomains. Note that, for unstructured domain decompositions, these edges are rather rare. As in the two-dimensional case, we will additionally consider a three-class approach; cf. subsection 3.3. As input for the machine learning model, we now sample from the three-dimensional coefficient function creating a three-dimensional image representation; see subsection 3.1 and Figure 2 (left). This step is significantly extended compared to the two-dimensional case.

The eigenvalue classification problem in our approach is essentially an image classification task. Therefore, as in [24], we will employ neural networks, which have been proven to be powerful models for image classification. In general, classification is a task of supervised machine learning. Supervised learning models approximate the nonlinear functional relationship between input and output data $F : I \to O$, with the input space I being a product of $\mathbb{R}$, $\mathbb{N}$, and boolean vector spaces. For classification problems, as considered here, the output space is typically an $\mathbb{N}$ vector space. A detailed description of supervised learning and feedforward neural networks (or multilayer perceptrons, respectively), can, e.g., be found in [17]. Here, we will use dense neural networks, which means that each neuron in a given layer is generally connected with all neurons in the previous layer; cf. Figure 1 for a visualization of a dense feedforward neural network. However, to further improve the generalization properties for our neural network, we use dropout layers with a dropout rate of 20%; see also [24]. Let us note that analogously to [24], we choose the ReLU (Rectified Linear Unit) function [29, 56, 16] as our activation $\alpha(x)$ in all our numerical experiments, which is defined by $\alpha(x) = \max\{0, x\}$. The training and validation procedure for the neural network model is described in subsection 3.2 and first results on the training and validation data are presented in subsection 3.4.

For a complete description of the employed machine learning framework, please refer to [24, Sec. 3] and the references therein.

3.1. Data preprocessing. In analogy to [24], we aim to train and test our neural network for both regular domain decompositions as well as for domain decompositions obtained from the graph partitioning software METIS [30]. We will observe that extending our methods introduced in [24] from two dimensions to three dimensions causes additional challenges and additional effort is needed preprocessing the input data for our machine learning model. The preprocessing of input data is at the core of our hybrid ML-FETI-DP algorithm. Hence, the preprocessing of the three-dimensional input data is one of the main novelties compared to the two-dimensional case. Generally, the sampling should cover all elements in a neighborhood of the respective interface component. Therefore, in order to prevent an incorrect or incomplete picture of the material distribution resulting from gaps in the sampling grid, a smoothing procedure for irregular edges, in two dimensions, or irregular faces, in three dimensions, is necessary; see [24, Fig.4] for a graphical representation of the smoothing procedure in two dimensions. Moreover, an additional challenge in the sampling procedure for irregular faces, such as faces obtained via METIS (METIS faces), with an arbitrary orientation in the three-dimensional space, arises. In particular, a consistent ordering of the sampling points is neither a priori given nor obvious. More precisely, there is no natural ordering of a grid of points on an irregular face, such as going from the lower left corner to the upper right corner. A consistent ordering of the sampling points is, however, essential when using them as input data to train a neural network. In particular, since neural networks rely on input data with a fixed structure, an important requirement of our data preprocessing is to provide samples

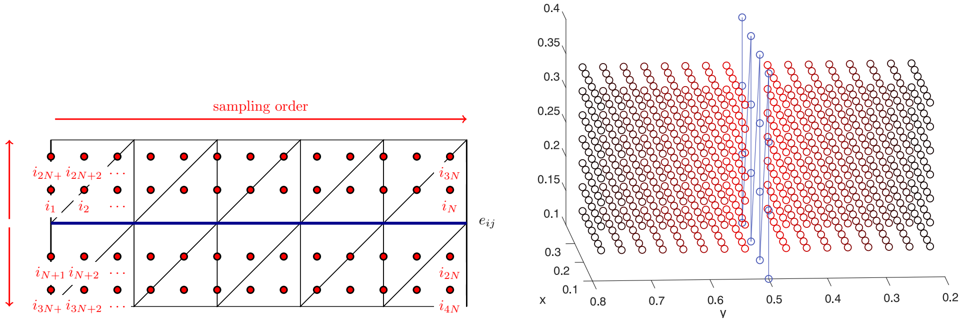

Fig. 2. **Left:** *Visualization of the ordering of the sampling points in 2D (red) for a straight edge (blue). Figure from [24, Fig. 3].* **Right:** *Visualization of the computed sampling points in 3D (red) for a regular face (blue) between two neighboring subdomains. The different shades correspond to increasing distance of the sampling points to the face and therefore to a higher numbering of the sampling points.*

of the coefficient distribution with a consistent spatial structure in relation to each face in our domain decomposition, even though the faces may vary in their location, orientation, and shape. In this approach, some sampling points may lay outside the two subdomains adjacent to a face. We encode these points using a specific dummy value which differs clearly from all true coefficient values. Since all coefficient values are positive, we encode sampling points outside the adjacent subdomains by the value -1. This is essential to ensure that we always generate input data of a fixed length for the neural network; see also [24].

3.1.1. Sampling procedure for regular faces. In case of regular faces, the procedure is fairly similar to the approach for straight edges in a two-dimensional domain decomposition; see [24]. Basically, we compute a tensor product sampling grid by sampling in both tangential directions of a face as well as in the direction orthogonal to the face. This results in a box-shaped structure of the sampling points in both neighboring subdomains of the face; see also Figure 2 (right). A required consistent ordering of the sampling points is provided by passing through the sampling points 'layer by layer' with growing distance relative to the face.

3.1.2. Sampling procedure for METIS faces. Our sampling procedure for METIS faces consists of two essential steps. First, we construct a consistently ordered two-dimensional auxiliary grid on a planar projection of each face. Second, we extrude this auxiliary grid into the two adjacent subdomains of the face. The resulting three-dimensional sampling grid has both a fixed size and a consistent ordering for all faces. Sampling points which do not lie on the face or within the two adjacent subdomains are encoded using the dummy value -1.

First step – Construction of a consistently ordered auxiliary grid for METIS faces. In order to construct the auxiliary grid for a METIS face, we first compute a projection of the original face represented in the three-dimensional Euclidean space onto an appropriate two-dimensional plane. In particular, we project a given METIS face onto a two-dimensional plane, such that we obtain a consistently sorted grid covering the face. This grid is induced by a tensor product grid on the two-dimensional projection plane. Note that since we use tetrahedral finite elements in three dimensions, each METIS face is naturally decomposed into triangles. Due to the projection from three

This manuscript is for review purposes only.

dimensions to two dimensions, elements, i.e., triangles of the face, can be degraded or deformed, i.e., they can have a large aspect ratio. We can also obtain flipped triangles; see Figure 3 for an example where both cases occur. Hence, we have to regularize the two-dimensional projection of the face before constructing the sampling grid.

To obtain a well-shaped projection of the face which is appropriate for our purpose, we numerically solve an optimization problem with respect to the two-dimensional projection of the face. More precisely, the objective functions of the optimization problem are carefully designed such that flipped triangles (phase 1) as well as sharp-angled triangles (phase 2) are prevented:

$$\min_x \sum_{T_j} \lambda_1 \cdot e^{-\lambda_2 \cdot \det(T_j(x))} + \lambda_{\text{reg}} \cdot \|d(x)\|_2^2 \quad \textbf{(phase 1)} \quad \text{and}$$

$$\min_x \sum_{T_j} \frac{{l_{p_j}}^2(x) + {l_{q_j}}^2(x)}{2 \cdot A_j(x)} \quad \textbf{(phase 2)}.$$

Here, we denote by x the coordinate vector of all corner points of all triangles of a given face after projection onto the two-dimensional plane, by $A_j(x)$ the area of a given triangle $T_j(x)$, and by $l_{p_j}(x), l_{q_j}(x)$ the lengths of two of its edges. By $d(x)$ we denote the displacement vector containing the displacements of all points x from the initial state prior to the optimization process. Furthermore, we denote by $\det(T_j(x))$ the determinant of the transformation matrix which belongs to the affine mapping from the unit triangle, i.e., the triangle with the corner points $(0,0), (1,0)$, and $(0,1)$, to a given triangle $T_j(x)$. We also introduce scalar weighting factors λ_1, λ_2, and λ_{reg} to control the ratio of the different terms within the objective functions. The concrete values for these weights were chosen heuristically and for all our computations, we used the values $\lambda_1 = 1, \lambda_2 = 50$, and $\lambda_{\text{reg}} = 10$.

Let us briefly motivate our objective functions in more details. Prior to the optimization of phase 1, we locally reorder the triangle corners, such that $\det(T_j(x))$ is negative for all flipped triangles. In order to do so, we start with one triangle and define it either as flipped or non-flipped. Then, we go through the remaining triangles of the projected face and classify them based on the following equivalence relation: two adjacent triangles are equivalent if and only if they do not overlap. Depending on the label of the initial triangle, we obtain two values for the objective function of phase 1, and we choose our classification into flipped and non-flipped triangles such that we start with the lower value. After this, flipped triangles can always be identified by a negative determinant of the respective transformation matrix. Therefore, we explicitly penalize such negative determinants in phase 1 of our optimization by minimizing the factors $\lambda_1 \cdot e^{-\lambda_2 \cdot \det(T_j(x))}$. Note that we also add the regularization term $\lambda_{\text{reg}} \cdot \|d(x)\|_2^2$ to the objective function to prevent that the projection can be arbitrarily shifted or rotated in the given plane. In phase 2, we minimize the sum of all fractions $\frac{{l_{p_j}}^2(x)+{l_{q_j}}^2(x)}{2 \cdot A_j(x)}$. This specific fraction is inspired by geometrical arguments; see also [15, Sect. 4]. It is minimized to obtain equilateral triangles, i.e., a high value in this fraction corresponds to a triangle with large aspect ratio. The fraction may actually be infinity if $A_j(x) = 0$. This may happen if a triangle is initially projected onto a straight line. However, in the first optimization phase, small areas are penalized in terms of the determinant, such that we do not obtain values close to zero in the second phase.

We start the optimization procedure with the initial projection onto the plane $x = 0$, $y = 0$, or $z = 0$ that results in the lowest objective value when adding

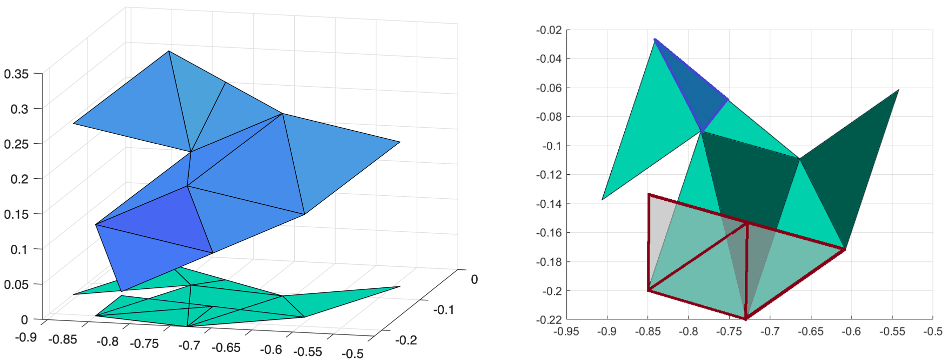

Fig. 3. **Left:** *Example of a typical METIS face in the three-dimensional space (blue triangles) and its corresponding projection onto the two-dimensional plane $z = 0$ (green triangles).* **Right:** *Due to the projection, we obtain both flipped triangles, which are marked in grey with red edges, and degraded triangles with a large aspect ratio, from which one is marked in blue. Let us remark that the different shades of green are only introduced for visualization purpose and do not have any physical meaning, e.g., different coefficients.*

the objective functions of phase 1 and phase 2. Then, we use the gradient descent algorithm as an iterative solver and optimize, i.e., minimize, alternating in succession the two aforementioned objective functions. The optimization procedure is stopped if the norm of the relative change of the coordinate vector of the triangles with respect to the prior iteration is below a factor of $1e - 6$ in both phases. Please see Figure 4 and Figure 5 for an example of the different steps of the optimization procedure in phase 1 and phase 2, respectively, for an exemplary METIS face consisting of ten triangles. Let us note that for all tested faces in section 4 the optimization procedure did always converge in phase 1 and phase 2, respectively, before the maximum number of iterations was reached, which we set to 500. Additionally, in almost all cases, only optimizing twice in phase 1 and once in phase 2 - alternating in succession - was necessary to obtain an appropriate projection of a given METIS face.

As the next step, we construct the smallest possible two-dimensional tensor product grid aligned with the coordinate axes covering the obtained optimized two-dimensional projection of the face; see also Figure 6 (left) for an example. Let us remark that this grid has a natural ordering of the grid points starting in the lower left corner and proceeding row by row to the upper right corner. We then make use of barycentric coordinates to map the grid, together with the corresponding ordering, back into the original triangles in the three-dimensional space. Based on the ordering of the grid points in two dimensions, we can now establish a natural ordering of the points in three dimension; see also Figure 6 (right).

Let us summarize the complete process to obtain the auxiliary grid points for each triangle of a specific face with a consistent ordering. First, we project the face from the three-dimensional space (Figure 3 (left: blue face)) onto a two-dimensional plane (Figure 3 (left: green face)). Second, we remove all flipped triangles (phase 1) and optimize the shape of all triangles (phase 2) of the projected face in an iterative optimization process; see Figures 4 and 5. Finally, we cover this optimized face by a two-dimensional tensor product grid with a natural ordering (Figure 6 (left)) and project these points back to the original face in three dimensions (Figure 6 (right)). Therefore, local barycentric coordinates can be used.

Let us note that, in our numerical experiments in section 4, this procedure was

This manuscript is for review purposes only.

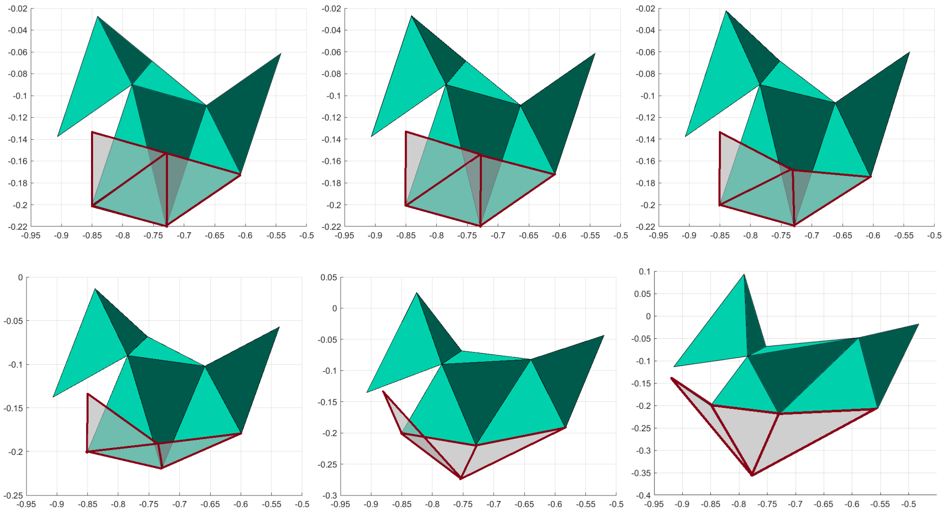

Fig. 4. *Visualization of the optimization process of the original projection in two dimensions in **phase 1** after 0, 20, 30, 50, 100 and 150 iteration steps (from upper left to lower right).*

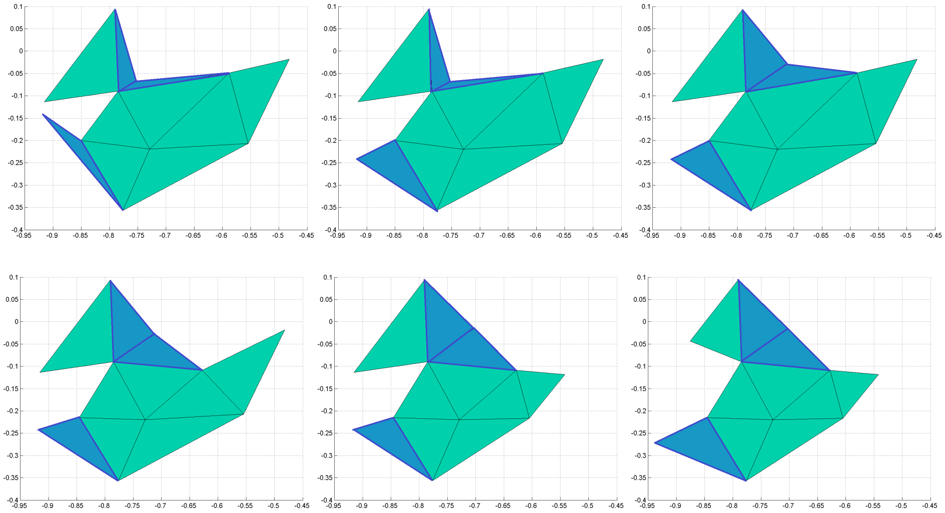

Fig. 5. *Visualization of the optimization process of the original projection in two dimensions in **phase 2** after 0, 10, 30, 50, 70 and 100 iteration steps (from upper left to lower right). Let us remark that the initial state here is the same as the final state from Figure 4.*

always successful. However, in general, there may be rare cases where our optimization does not converge to an acceptable two-dimensional triangulation. For instance, it is possible that a subdomain is completely enclosed by another subdomain, such that the face between the two subdomains is actually the complete boundary of the interior subdomain. In this case, we cannot remove all flipped triangles without changing the structure of the face. If we detect that our optimization does not converge to an acceptable solution, we can still proceed in the two following ways: either we mark

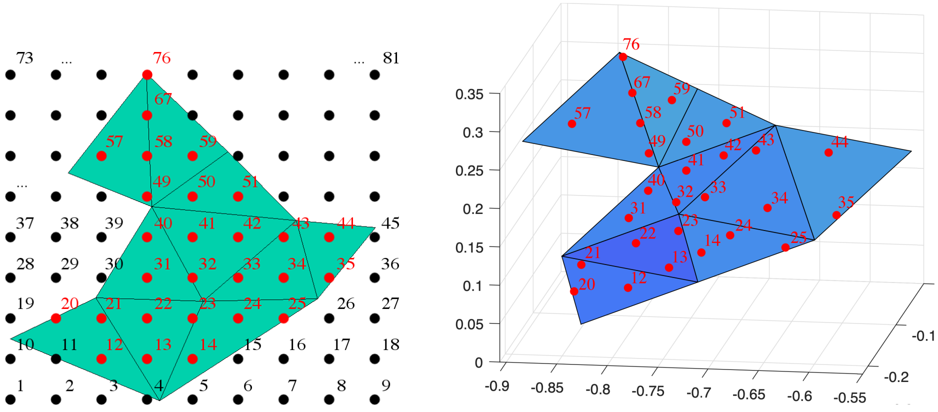

FIG. 6. **Left:** *Two-dimensional projection of the original face (depicted on the right) after both optimization phases have been carried out; the optimized projection is covered by a regular grid with natural ordering; same face as in the last picture of* Figure 5. **Right:** *Original face in three dimensions with corresponding grid points; numbers are obtained by a projection from two dimensions (left) back to three dimensions using barycentric coordinates.*

the eigenvalue problem corresponding to the face as necessary, or we split the face into smaller faces and consider each of the smaller faces separately in our ML-FETI-DP algorithm.

Second step – Extrusion of the auxiliary grid into three dimensions. Starting from the ordered auxiliary points on the face, we can now build a three-dimensional sampling grid. For this purpose, for each of the auxiliary points on the face, we first define a *sampling direction vector* pointing into one of the two adjacent subdomains. Second, we extrude the two-dimensional auxiliary grid on the actual METIS face into the two neighboring subdomains along the sampling directions, resulting in a three-dimensional sampling grid. Note that the first layer of sampling nodes does not lie on the face itself but next to it; cf. Figure 2. Moreover, we neglect all points of the auxiliary grid, which are outside the METIS face, and encode all corresponding points in the three-dimensional sampling grid by the dummy value -1. Similar as for edges obtained by a two-dimensional METIS decomposition, choosing the normal vectors of the triangles as sampling direction vectors in the extrusion process leads to gaps in the three-dimensional sampling grid close to the face; see also [24, Sec. 3.1, Fig. 4] for a two-dimensional graphical representation. This is caused by the fact that, in general, METIS faces are not smooth. As we have already shown for edges in the two-dimensional case, the neighborhood of an interface component will be the most important for the decision if adaptive constraints are necessary or not and therefore the aforementioned gaps have to be minimized; see [25]. To avoid these gaps and thus to obtain sampling points in most finite elements close to the face, we suggest the use of sampling directions obtained by a moving average iteration over the normal vectors of the face. In some sense, this can be interpreted as a smoothing of the face or, more precisely, the field of normal vectors on the face.

The following procedure turned out to be the most appropriate for our purposes in the sense that, on average, for each face and each neighboring subdomain, it results in the highest number of sampled elements relative to the overall number of elements in the subdomain. Here, we first uniformly refine all triangles of a given METIS face once by subdividing each triangle of the face into four new regular triangles. For

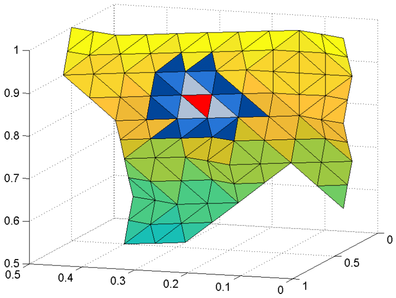

FIG. 7. *Visualization of the moving average procedure to obtain the sampling direction vectors for the extrusion of the auxiliary grid. For the red triangle, all grey, light blue and dark blue triangles are considered recursively as grouped by colors for a moving average with a window length of 3.*

each of the resulting finer triangles, we compute the normal vector originating in its centroid. We then use the normal vectors of the refined triangulation to compute a single sampling direction for each triangle of the original triangulation of the face. For this purpose, we first 'smooth' the field of normal vectors of the refined triangulation by using a component-wise moving average, applied twice recursively with a fixed window length of 3. Subsequently, by computing the average of the resulting normal vectors of all four related finer triangles we finally obtain the sampling direction of the original triangles. Then, we use the same sampling direction for all points of the auxiliary grid which are located in the same triangle.

Let us briefly describe the moving average approach and the meaning of the window length in more details. For each triangle of the refined face, one after another, we replace the normal vector by a component-wise average of the normal vector itself and the normal vectors of certain surrounding triangles. The triangles considered in the averaging process are aggregated recursively as follows. In a first step, for a given triangle, we add all neighboring triangles that share an edge with the given triangle to obtain a patch with a window length of 1. Recursively, for an increasing window length, we add all triangles that share an edge with a triangle that has been selected in the previous step. Please see also Figure 7 for an exemplary visualization of all considered triangles for a moving average with the window length of 3.

Finally, we use the obtained sampling directions to compute the final three-dimensional sampling grid in the two neighboring subdomains of the face. In Figure 8, we visualize all sampled (middle) and non-sampled (right) finite elements using the described procedure for an exemplary METIS face; we call a finite element "sampled" if it contains at least one sampling point. We can observe that, especially in the close neighborhood of the face, we obtain sampling points in almost all finite elements.

As final input for our neural network, we use a vector containing evaluations of the coefficient function ρ or the Young modulus E, respectively, for all points in the sampling grid.

3.2. Training and validation phase.

For the training and validation of the neural network, we use a data set containing approximately 3 000 configurations of pairs of coefficient functions and subdomain geometries for two subdomains sharing a face. To obtain the output data, i.e., the correct classification labels, for the training of the neural network, we solve the eigenvalue problem described in subsection 2.3.2 for each of these configuration. Note that the correct classification label for a specific face does not only depend on the geometry and the coefficient distribution but also

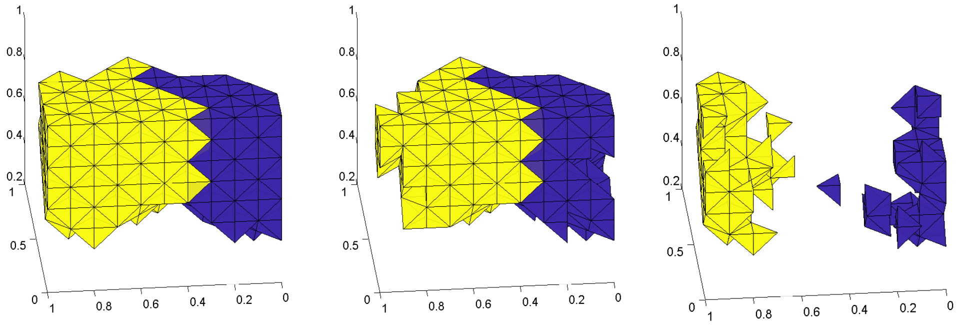

FIG. 8. *Visualization of a METIS face between two neighboring subdomains **(left)** and all sampled **(middle)** and non-sampled **(right)** FE's when using the described sampling procedure.*

on the underlying PDE. Therefore, we will use the same configurations for diffusion and elasticity problems but compute the correct classification labels separately.

In [24], for the two-dimensional case, we used only two edge geometries, i.e., a regular edge and an edge with a single jag, and combined them with a set of carefully designed coefficient distributions, resulting in a total of 4 500 configurations; in [22], this data set based on manually designed coefficient distributions was also referred to as 'smart data'. Since both the domain decomposition and the coefficient distribution may be more complex in three dimensions compared to two dimensions, we use a different approach for the generation of our training and validation data. In particular, we consider six different meshes resulting from regular domain decompositions of the unit cube into $4 \times 4 \times 4 = 64$ or $6 \times 6 \times 6 = 216$ subdomains of size $H/h = 6, 7, 8$. For each of these meshes, we generate 30 different randomly generated coefficient distributions based on the approach discussed in [22]. More precisely, we control the ratio of high vs. low coefficient voxels and impose some light geometrical structure. In particular, we build connected stripes of high coefficient with a certain length in x, y, or z direction, and additionally combine them by a pairwise superimposition; cf. [22] for a more detailed description for the two-dimensional case and Figure 9 for an exemplary coefficient distribution in three dimensions. In analogy to [22], we refer to this set of coefficient functions as *random data*.

For each combination of mesh and coefficient distribution, we now consider the aforementioned regular domain decomposition as well as a corresponding irregular domain decomposition into 64 or 216 subdomains, respectively, obtained using the graph partitioning software METIS [30]. Finally, we consider the eigenvalue problems corresponding to all resulting faces combined with the different coefficient distributions. As mentioned before, we obtain a total of approximately 3 000 configurations.

Note that, in general, using a smaller number of METIS subdomains, we obtained face geometries which resulted in worse generalization properties of our neural networks. Moreover, in contrast to the two-dimensional case, where we needed at least 4 500 configurations, we are here able to obtain very good results for total of only roughly 3 000 configurations. This is likely due to the much smaller numbers of finite elements per subdomain used, compared to our two-dimensional experiments in [24].

For the sampling, we select 22 points in both of the two tangential directions of the auxiliary grid of a face and 22 points in orthogonal direction for each of the two adjacent subdomains; hence, we obtain approximately two sampling points in each finite element when using a subdomain size of $H/h = 10$.

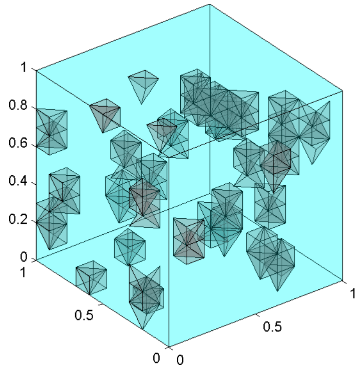

Fig. 9. *Example of a randomly distributed coefficient function in the unit cube obtained by using the same randomly generated coefficient for a horizontal or vertical beam of a maximum length of 4 finite element voxels. The grey voxels correspond to a high coefficient and we have a low coeffient of* 1 *otherwise. Visualization for* $2 \times 2 \times 2$ *subdomains and* $H/h = 5$.

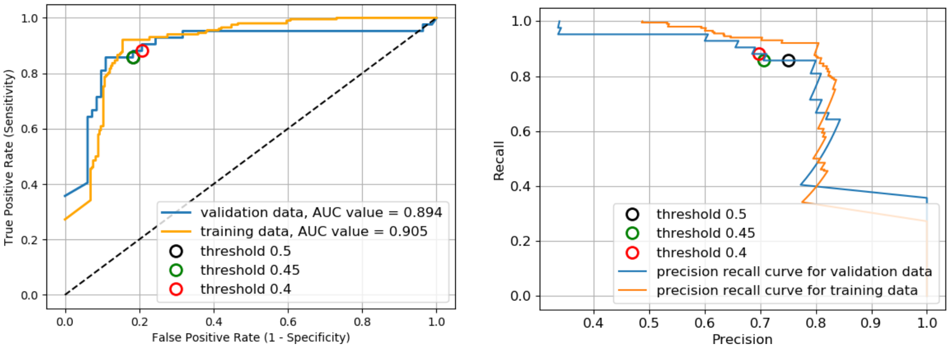

Fig. 10. *ROC curve and precision-recall plot for the optimal model obtained by a grid search. We define precision as true positives divided by (true positives+false positives), and recall as true positives divided by (true positives+false negatives). The thresholds used in section 4 are indicated as circles.*

As in [24], we train the neural network using the Adam (Adaptive moments) [32] optimizer, a variant of the Stochastic Gradient Descent (SGD) method with adaptive learning rate. The hyper parameters for the training process and the neural network architecture are again chosen based on a grid search with cross-validation. More precisely, we compare the training and generalization properties of different neural networks for several random splittings of our entire data set into 80 % training and 20 % validation data; cf. [24] for details on the hyper parameter search space and finally chosen set of hyper parameters. The Receiver Operating Characteristic (ROC) curve and a precision-recall plot of the neural network with optimal hyper parameters are shown in Figure 10. Let us note that we use the same neural network for both, regular and METIS decompositions, in our numerical experiments.

3.3. Extension to three-class classification using frugal face constraints.

As described in subsection 2.4, we can replace the adaptive constraints by less costly frugal constraints on faces, where only a single constraint (in case of a stationary diffusion problem) or less than or equal to six constraints (in case of linear elasticity) are necessary. Please see [23] for a detailed description of the resulting frugal con-

classification type	**threshold**	**fp**	**fn**	**acc**
two-class classification	0.45	2.76%	1.76%	95.5%
	0.5	1.70%	3.40%	94.9%
three-class classification	0.4	5.2%	1.7%	93.1%
	0.5	2.1%	2.3%	95.6%

Table 1

Results on the complete training and validation data set. We define the accuracy (acc) as the number of true positives and true negatives divided by the total number of training and validation configurations.

straints in two and three dimensions. Consequently, if known a priori, we can omit the eigenvalue problem to compute the adaptive constraints on these faces. Thus, we also propose an extended three-class classification approach for faces, analogously to the three-class approach for edges in [24]. Here, we train a neural network which distinguishes between the following three classes: faces, where the eigenvalue problem is unnecessary (class 0), where the eigenvalue problem results in exactly one (for stationary diffusion) or less than or equal to six (linear elasticity) constraints (class 1), and where the eigenvalue results in more than one or six, respectively, constraints (class 2). For all faces assigned to class 0, we do not enforce any face constraint. For all faces assigned to class 1, we enforce the frugal face constraints as described in [23]. Only for the remaining faces, we solve the corresponding eigenvalue problems and enforce the computed adaptive constraints.

3.4. Results on the training data. On the complete set of training and validation data, we obtain the results listed in Table 1. As in [24], we used the classification thresholds 0.45 and 0.5 for the two-class classification and 0.4 and 0.5 for the three-class classification, respectively. For the two-class classification, we observe nearly the same accuracy values when using the classification threshold 0.5 and 0.45. For the three-class classification, however, lowering the threshold to 0.4 results in a lower accuracy value than for using the threshold of 0.5. In both cases, the number of false negative faces, which corresponds to the number of critical faces not detected by the algorithm and which are critical for the convergence of the iterative FETI-DP solver, can be reduced by decreasing the threshold. We denote this approach to improve the robustness as 'overshooting'; cf. [24]. In section 4, we will always compare the results for the default threshold, 0.5, and the overshooting threshold, 0.45 and 0.4 for the two-class and three-class model, respectively.

4. Numerical results for ML-FETI-DP.

In this section, we provide comparative results for the classical FETI-DP, adaptive FETI-DP, and our ML-FETI-DP method. We present numerical results for different coefficient functions ρ in model problem (2.1) and different distributions for the Young modulus E in (2.2). We always use structured tetrahedral meshes of the unit cube constructed from discretizing each voxel of a regular voxel mesh by five piecewise linear tetrahedral finite elements; all coefficient distributions are chosen to be constant on each voxel. For all our numerical computations, we use the preconditioned conjugate gradient (PCG) algorithm. As the stopping criterion for PCG we use a relative reduction of the preconditioned residual by a factor of 1e-8. For adaptive FETI-DP, we use the tolerance $TOL = 100$. In our comparison, we consider both domain decompositions into regular, cubic subdomains as well as irregular domain decompositions obtained from METIS [30]. Please note that the configurations appearing in the numerical experiments in this section are

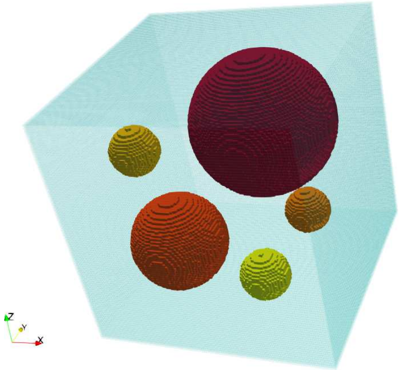

Fig. 11. *Five spheres with different radii in the unit cube. Resolution of* $128 \times 128 \times 96$ *voxel corresponding to computations with* $H/h = 16$ *and* $8 \times 8 \times 6 = 384$ *subdomains. Figure from [23, Fig. 11].*

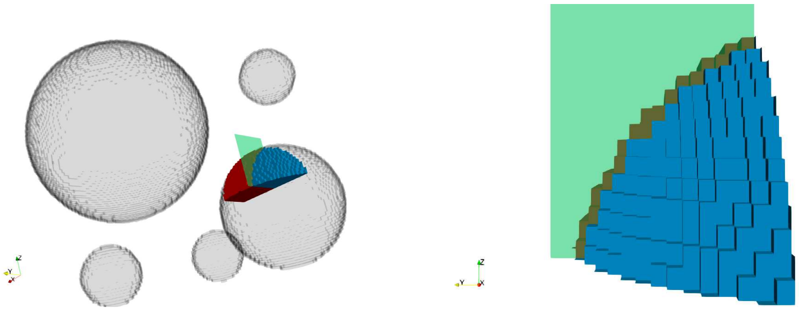

Fig. 12. ***Left:*** *Visualization of the area with a high coefficient for two neighboring subdomains, marked in red and blue, and the face between those subdomains, marked in green; see Figure 11 for the complete coefficient function.* ***Right:*** *Zoom-in of the coefficient jump along the green face between two neighboring subdomains. Figure from [23, Fig. 12].*

generally not part of our training and validation data set. In particular, we have chosen both different coefficient distributions as well as combinations of the numbers of elements and subdomains.

4.1. Coefficient functions. For stationary diffusion, we consider a coefficient function based on five spherical inclusions of different radii in the unit cube; see Figure 11. Here, all voxels within the five spheres have an identical high coefficient ρ, whereas the remaining voxels all have a small coefficient.

As the second model problem, we consider a linear elastic representative volume element (RVE) of a dual-phase steel representing the microstructure of a DP600 steel and obtained by an EBDS (electronic backscatter diffraction) measurement. This dual-phase steel consists of a martensitic phase and a ferritic phase. In our computations, we use a high coefficient in the martensitic phase and a low coefficient in the ferritic phase of the material. The most realistic model problem considered here is the case of a coefficient contrast of $1e3$. Let us note that the RVE is part of a larger microstructure which was presented in [5].

4.2. Two-class model. Let us first discuss our two-class model. Here, the neural network distinguishes between faces, where the eigenvalue problem results in at

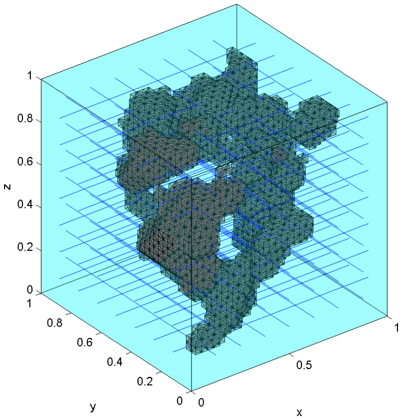

FIG. 13. *Coefficient distribution on a representative volume element (RVE) of a dual-phase steel in the unit cube. This RVE is part of a larger structure presented in [5]. Visualization of the domain decomposition into* 512 *subdomains and* $H/h = 4$*. High coefficients are shown in black, and subdomains are shown by blue slices. The higher coefficient is* $E_1 = 1e3$ *and the lower coefficient is* $E_2 = 1$*, with* $\nu = 0.3$ *everywhere.*

least one (stationary diffusion) or six (linear elasticity) additional adaptive constraints and faces, where the eigenvalue is unnecessary. In analogy to [24], we will refer to the latter case as "negative" or "negative face" and to the first case as "positive" or "positive face". For the adaptive algorithm, we always use a tolerance of $TOL = 100$.

4.2.1. Regular domain decompositions. Let us first consider the stationary diffusion problem, where the coefficient distribution is given by the spherical inclusions depicted in Figure 12. We partition the cubic domain into $4 \times 4 \times 4$ regular subdomains with subdomain size $H/h = 10$. In Table 2, we compare the iteration counts and condition number estimates for the classical FETI-DP, adaptive FETI-DP, and the new ML-FETI-DP method. As already done in [24], we also report the number of false negative and false positive faces resulting from our machine learning classification for two different ML thresholds τ; cf. the discussions in subsection 3.4 and [24]. Let us note that only false negative faces are critical for the convergence of the ML-FETI-DP method, whereas false positive faces correspond to eigenvalue problems, which are solved even though they are not necessary for the robustness of the algorithm. We observe that, when using the ML threshold of $\tau = 0.5$, we obtain two false negative faces. This leads to a worse condition number estimate, while the iteration number of the algorithm is still satisfactory. By lowering the ML decision threshold to $\tau = 0.45$, we are able to eliminate all false negative faces and thus to correctly identify all critical faces, where the eigenvalue problem is necessary. In particular, using our ML-FETI-DP approach with overshooting, we solve only 12 eigenvalue problems in contrast to 144 eigenvalue problems for the fully adaptive approach. Nonetheless, we are still able to retain the same condition number estimate and iteration count. This is indeed a major saving in the number of eigenvalue problems and thus computation time.

We further provide numerical results for the linear elasticity problem using the RVE in Figure 13 as material distribution. Here, we decompose our domain into $8 \times 8 \times 8$ regular subdomains with a reduced size $H/h = 6$. Let us note again that we use a neural network different from the one used for diffusion problems since the correct labels may differ; cf. subsection 3.2. We summarize the comparative results for this model problem in Table 3. As for the stationary diffusion problem

Model Problem	Algorithm	τ	cond	it	evp$_{\mathcal{F}}$	fp	fn	acc
Five Spheres	standard	-	-	>350	0	-	-	-
	adaptive	-	44.97	63	144	-	-	-
	ML	0.5	2.73e4	67	7	2	2	0.97
	ML	0.45	44.97	63	12	5	0	0.96

Table 2

Comparison of standard FETI-DP, adaptive FETI-DP, and ML-FETI-DP for **regular domain decompositions** *for the* **two-class model***. Decomposition of the unit cube into* $4 \times 4 \times 4$ *subdomains and* $H/h = 10$*.* ***Stationary diffusion*** *problem. We show the ML threshold* (τ)*, the condition number (cond), the number of CG iterations (it), the number of solved eigenvalue problems on faces (evp*$_{\mathcal{F}}$*), the number of false positives (fp), the number of false negatives (fn), and the accuracy of the classification (acc).*

Model Problem	Algorithm	τ	cond	it	evp$_{\mathcal{F}}$	fp	fn	acc
RVE Problem	standard	-	-	>350	0	-	-	-
	adaptive	-	16.89	39	1344	-	-	-
	ML	0.5	3.76e4	45	52	10	5	0.98
	ML	0.45	16.89	40	66	19	0	0.98

Table 3

Comparison of standard FETI-DP, adaptive FETI-DP, and ML-FETI-DP for **regular domain decompositions** *for the* **two-class model***. Decomposition of the unit cube into* $8 \times 8 \times 8$ *subdomains and* $H/h = 6$*.* ***Linear elasticity*** *problem. See Table 2 for the column labeling.*

in Table 2, we are able to eliminate all false negative faces and thus obtain a robust algorithm when using an ML threshold of 0.45. Furthermore, we observe that only 66 eigenvalue problems have to be solved for ML-FETI-DP in comparison to 1344 eigenvalue problems solved for the fully adaptive FETI-DP method.

4.2.2. METIS domain decompositions. We also consider an irregular domain decomposition obtained via METIS for the same stationary diffusion and linear elasticity problems as in subsection 4.2.1. The corresponding results are summarized in Table 4 and Table 5, respectively. For the stationary diffusion problem, the ML algorithm misses 4 critical faces when using the ML threshold $\tau = 0.5$. However, when lowering the ML threshold to $\tau = 0.45$, we are again able to identify all critical faces. Consequently, we retain nearly the same convergence behavior as for the adaptive FETI-DP method, while solving only 38 instead of 288 eigenvalue problems; see Table 4. For the elasticty problem, the results are fairly comparable; see Table 5. Again, when using the ML threshold of $\tau = 0.45$, we are able to identify all faces which are critical for the convergence of the algorithm. In particular, we only have to solve 92 eigenvalue problems instead of 1547 for the adaptive FETI-DP approach.

4.3. Three-class model. Let us now discuss the results for our three-class model; cf. subsection 3.3. Let us note once more that, in the three-class approach, we now construct frugal face constraints instead of solving an eigenvalue problem if our neural network labels a face as class 1; cf. subsection 2.4. Thus, we do not need to solve any eigenvalue problems for the corresponding faces. As for the two-class model, we always use a tolerance of $TOL = 100$ in the adaptive algorithm. We consider the same coefficient functions and material distributions, respectively, as in subsection 4.2.

4.3.1. Regular domain decompositions. The results for the stationary diffusion problem are summarized in Table 6. As for the two-class model, we are able

Model Problem	**Algorithm**	τ	**cond**	**it**	**evp**$_{\mathcal{F}}$	**fp**	**fn**	**acc**
Five Spheres	standard	-	-	>350	0	-	-	-
	adaptive	-	30.24	49	288	-	-	-
	ML	0.5	3.17e4	55	27	5	4	0.97
	ML	0.45	30.25	50	38	12	0	0.96

TABLE 4

Comparison of standard FETI-DP, adaptive FETI-DP, and ML-FETI-DP for **METIS domain decompositions** *for the* **two-class model***. Decomposition of the unit cube into* 64 *subdomains and* $H/h = 10$*.* ***Stationary diffusion*** *problem. See Table 2 for the column labeling.*

Model Problem	**Algorithm**	τ	**cond**	**it**	**evp**$_{\mathcal{F}}$	**fp**	**fn**	**acc**
RVE Problem	standard	-	-	>350	0	-	-	-
	adaptive	-	20.13	41	1547	-	-	-
	ML	0.5	3.57e4	47	77	10	6	0.98
	ML	0.45	20.13	41	91	18	0	0.98

TABLE 5

Comparison of standard FETI-DP, adaptive FETI-DP, and ML-FETI-DP for **METIS domain decompositions** *for the* **two-class model***. Decomposition of the unit cube into* 512 *subdomains and* $H/h = 6$*.* ***Linear elasticity*** *problem. See Table 2 for the column labeling.*

to eliminate all false negative faces for the three-class model, when using the ML threshold $\tau = 0.4$. However, in comparison to the two-class model in Table 4, we now only need to solve 9, instead of 12, of the original 144 face eigenvalue problems. Thus, due to the computation of the frugal constraints, we are able to further reduce the number of necessary eigenvalue problems, while retaining a robust algorithm.

For the linear elasticity problem, the results are again fairly similar; see Table 7. In this case, we are able to further reduce the number of necessary eigenvalue problems from 66 in Table 6 to 32 by using frugal face constraints for all faces classified to class 1.

4.3.2. METIS domain decompositions. Using the three-class model and METIS domain decompositions, we obtain similar results compared to those for regular domain decompositions in subsection 4.3. In Table 8, we present the results for the stationary diffusion problem. We observe that, for a robust choices of the ML threshold, the number of necessary face eigenvalue problems can be further reduced from 38, for the two-class model in Table 4 and $\tau = 0.45$, to 19, for $\tau = 0.4$. Considering the results for the linear elasticity problem in Table 9 does not change the picture. Using the three-class classification, we only need to solve 45 out of originally 1 547 eigenvalue problems on faces for $\tau = 0.45$ while retaining robustness and fast convergence of the algorithm.

5. Conclusion. We have extended our hybrid ML-FETI-DP approach, which combines adaptive FETI-DP methods and machine learning, to three dimensions. Using this approach, the number of necessary eigenvalue problems in an adaptive FETI-DP method for heterogeneous coefficient distributions may be significantly reduced. The extension to three dimensions required a rather complex but computationally relatively inexpensive preprocessing procedure to generate structured input data of the neural network, even for unstructured geometries. We have used both the two-class and the three-class classification approaches from [24], where the three-class approach utilizes the frugal constraints introduced in [23] to reduce the number of

Model Problem	**Algorithm**	τ	**cond**	**it**	**evp$_{\mathcal{F}}$**	**fp**	**fn**	**acc**
Five Spheres	standard	-	-	>350	0	-	-	-
	adaptive	-	44.97	63	144	-	-	-
	ML	0.5	1.36e4	66	5	4	3	0.95
	ML	0.4	46.77	64	9	13	0	0.91

Table 6

Comparison of standard FETI-DP, adaptive FETI-DP, and ML-FETI-DP for **regular domain decompositions** *for the* **three-class model**. *Decomposition of the unit cube into* $4 \times 4 \times 4$ *subdomains and* $H/h = 10$. ***Stationary diffusion*** *problem. See Table 2 for the column labeling.*

Model Problem	**Algorithm**	τ	**cond**	**it**	**evp$_{\mathcal{F}}$**	**fp**	**fn**	**acc**
RVE Problem	standard	-	-	>350	0	-	-	-
	adaptive	-	16.89	39	1344	-	-	-
	ML	0.5	4.27e3	44	27	11	5	0.98
	ML	0.4	18.49	40	32	26	0	0.98

Table 7

Comparison of standard FETI-DP, adaptive FETI-DP, and ML-FETI-DP for **regular domain decompositions** *for the* **three-class model**. *Decomposition of the unit cube into* $8 \times 8 \times 8$ *subdomains and* $H/h = 6$. ***Linear elasticity*** *problem. See Table 2 for the column labeling.*

eigenvalue problems even further than the two-class approach.

We have provided numerical results comparing the new three-dimensional ML-FETI-DP algorithm to classical and adaptive FETI-DP methods for diffusion and linear elasticity problems and realistic coefficient distributions. Using an overshooting strategy, we have always obtained a robust method with a low condition number estimate. When using the three-class approach and frugal constraints, we have been thus able to reduce the number of necessary eigenvalue problems by at least 86 %. In the best case, we even have been able to reduce the number of eigenvalue problems of the plain adaptive FETI-DP method from 1344 to 32 using the three-class approach; this corresponds to a reduction by more than 97 %.

REFERENCES

[1] S. Badia, A. F. Martín, and J. Principe, *On the scalability of inexact balancing domain decomposition by constraints with overlapped coarse/fine corrections*, Parallel Comput., 50 (2015), pp. 1–24.

[2] S. Badia, A. F. Martín, and J. Principe, *Multilevel balancing domain decomposition at extreme scales*, SIAM J. Sci. Comput., 38 (2016), pp. C22–C52.

[3] L. Beirão da Veiga, L. F. Pavarino, S. Scacchi, O. B. Widlund, and S. Zampini, *Adaptive selection of primal constraints for isogeometric BDDC deluxe preconditioners*, SIAM J. Sci. Comput., 39 (2017), pp. A281–A302.

[4] P. E. Bjørstad, J. Koster, and P. Krzyżanowski, *Domain decomposition solvers for large scale industrial finite element problems*, in PARA2000 Workshop on Applied Parallel Computing, Lecture Notes in Computer Science 1947, Springer-Verlag, 2000.

[5] D. Brands, D. Balzani, L. Scheunemann, J. Schröder, H. Richter, and D. Raabe, *Computational modeling of dual-phase steels based on representative three-dimensional microstructures obtained from ebsd data*, Arch. Appl. Mech., 86 (2016), pp. 575–598.

[6] J. G. Calvo and O. B. Widlund, *An adaptive choice of primal constraints for BDDC domain decomposition algorithms*, Electron. Trans. Numer. Anal., 45 (2016), pp. 524–544.

[7] J.-M. Cros, *A preconditioner for the Schur complement domain decomposition method*, in Domain Decomposition Methods in Science and Engineering, O. W. I. Herrera, D. Keyes and R. Yates, eds., National Autonomous University of Mexico (UNAM), Mexico City, Mexico, ISBN 970-32-0859-2, 2003, pp. 373–380. Proceedings of the 14th International

Model Problem	Algorithm	τ	cond	it	evp$_{\mathcal{F}}$	fp	fn	acc
Five Spheres	standard	-	-	>350	0	-	-	-
	adaptive	-	30.24	49	288	-	-	-
	ML	0.5	3.75e4	56	12	8	4	0.96
	ML	0.4	33.52	50	19	15	0	0.95

Table 8

Comparison of standard FETI-DP, adaptive FETI-DP, and ML-FETI-DP for **METIS domain decompositions** *for the* **three-class model**. *Decomposition of the unit cube into* 64 *subdomains and* $H/h = 10$. ***Stationary diffusion*** *problem. See* Table 2 *for the column labeling.*

Model Problem	Algorithm	τ	cond	it	evp$_{\mathcal{F}}$	fp	fn	acc
RVE Problem	standard	-	-	>350	0	-	-	-
	adaptive	-	20.13	41	1547	-	-	-
	ML	0.5	4.12e4	47	28	11	5	0.98
	ML	0.4	24.22	42	45	28	0	0.98

Table 9

Comparison of standard FETI-DP, adaptive FETI-DP, and ML-FETI-DP for **METIS domain decompositions** *for the* **three-class model**. *Decomposition of the unit cube into* 512 *subdomains and* $H/h = 6$. ***Linear elasticity*** *problem. See* Table 2 *for the column labeling.*

Conference on Domain Decomposition Methods in Science and Engineering.

[8] C. R. Dohrmann, *A preconditioner for substructuring based on constrained energy minimization*, SIAM J. Sci. Comput., 25 (2003), pp. 246–258.

[9] V. Dolean, F. Nataf, R. Scheichl, and N. Spillane, *Analysis of a two-level Schwarz method with coarse spaces based on local Dirichlet-to-Neumann maps*, Comput. Methods Appl. Math., 12 (2012), pp. 391–414.

[10] Y. Efendiev, J. Galvis, R. Lazarov, and J. Willems, *Robust domain decomposition preconditioners for abstract symmetric positive definite bilinear forms*, ESAIM Math. Model. Numer. Anal., 46 (2012), pp. 1175–1199.

[11] C. Farhat, M. Lesoinne, P. LeTallec, K. Pierson, and D. Rixen, *FETI-DP: a dual-primal unified FETI method. I. A faster alternative to the two-level FETI method*, Internat. J. Numer. Methods Engrg., 50 (2001), pp. 1523–1544.

[12] C. Farhat, M. Lesoinne, and K. Pierson, *A scalable dual-primal domain decomposition method*, Numer. Linear Algebra Appl., 7 (2000), pp. 687–714. Preconditioning techniques for large sparse matrix problems in industrial applications (Minneapolis, MN, 1999).

[13] J. Galvis and Y. Efendiev, *Domain decomposition preconditioners for multiscale flows in high-contrast media*, Multiscale Modeling & Simulation, 8 (2010), pp. 1461–1483.

[14] J. Galvis and Y. Efendiev, *Domain decomposition preconditioners for multiscale flows in high contrast media: reduced dimension coarse spaces*, Multiscale Modeling & Simulation, 8 (2010), pp. 1621–1644.

[15] R. V. Garimella, M. J. Shashkov, and P. M. Knupp, *Triangular and quadrilateral surface mesh quality optimization using local parametrization*, Computer Methods in Applied Mechanics and Engineering, 193 (2004), pp. 913–928.

[16] X. Glorot, A. Bordes, and Y. Bengio, *Deep sparse rectifier neural networks*, in Proceedings of the fourteenth int. conf. on artificial intelligence and statistics, 2011, pp. 315–323.

[17] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*, vol. 1, MIT press Cambridge, 2016.

[18] A. Heinlein, A. Klawonn, J. Knepper, and O. Rheinbach, *An adaptive GDSW coarse space for two-level overlapping Schwarz methods in two dimensions*, in Domain Decomposition Methods in Science and Engineering XXIV, vol. 125 of LNCSE, Springer, Cham, 2018, pp. 373–382.

[19] A. Heinlein, A. Klawonn, J. Knepper, and O. Rheinbach, *Multiscale coarse spaces for overlapping Schwarz methods based on the ACMS space in 2D*, Electronic Transactions on Numerical Analysis (ETNA), 48 (2018), pp. 156–182.

[20] A. Heinlein, A. Klawonn, J. Knepper, and O. Rheinbach, *Adaptive GDSW coarse spaces for overlapping Schwarz methods in Three Dimensions*, SIAM Journal on Scientific Com-

puting, 41:5 (2019), pp. A3045–A3072.

[21] A. Heinlein, A. Klawonn, M. J. Kühn, M. Lanser, O. Rheinbach, and J. Weber, *On the size of the eigenvalue problems in adaptive FETI-DP and BDDC*. In preparation.

[22] A. Heinlein, A. Klawonn, M. Lanser, and J. Weber, *Machine Learning in Adaptive FETI-DP - A Comparison of Smart and Random Training Data*, (2018). TR series, Center for Data and Simulation Science, University of Cologne, Germany, Vol. 2018-5. http://kups.ub.uni-koeln.de/id/eprint/8645. Accepted for publication in the proceedings of the International Conference on Domain Decomposition Methods 25, Springer LNCSE, May 2019.

[23] A. Heinlein, A. Klawonn, M. Lanser, and J. Weber, *A Frugal FETI-DP and BDDC Coarse Space for Heterogeneous Problems*, (2019). TR series, Center for Data and Simulation Science, University of Cologne, Germany, Vol. 2019-18. http://kups.ub.uni-koeln.de/id/eprint/10363. Submitted to ETNA.

[24] A. Heinlein, A. Klawonn, M. Lanser, and J. Weber, *Machine Learning in Adaptive Domain Decomposition Methods - Predicting the Geometric Location of Constraints*, SIAM J. Sci. Comput., 41 (2019), pp. A3887–A3912.

[25] A. Heinlein, A. Klawonn, M. Lanser, and J. Weber, *Machine Learning in Adaptive FETI-DP - Reducing the Effort in Sampling*, (2019). TR series, Center for Data and Simulation Science, University of Cologne, Germany, Vol. 2019-19. http://kups.ub.uni-koeln.de/id/eprint/10439. Accepted for publication in the proceedings of the ENUMATH 2019 conference, Springer LNCSE. Accepted May 2020.

[26] A. Heinlein, A. Klawonn, and O. Rheinbach, *A parallel implementation of a two-level overlapping schwarz method with energy-minimizing coarse space based on trilinos*, SIAM Journal on Scientific Computing, 38 (2016), pp. C713–C747.

[27] A. Heinlein, A. Klawonn, O. Rheinbach, and O. B. Widlund, *Improving the parallel performance of overlapping schwarz methods by using a smaller energy minimizing coarse space*, in Domain Decomposition Methods in Science and Engineering XXIV, P. E. Bjørstad, S. C. Brenner, L. Halpern, H. H. Kim, R. Kornhuber, T. Rahman, and O. B. Widlund, eds., Cham, 2018, Springer International Publishing, pp. 383–392.

[28] A. Heinlein, M. Kühn, and A. Klawonn, *Local spectra of adaptive domain decomposition methods*, (2018). TR series, Center for Data and Simulation Science, University of Cologne, Germany, Vol. ????-?. http://kups.ub.uni-koeln.de/id/eprint/9019. Accepted for publication in the proceedings of the International Conference on Domain Decomposition Methods 25, Springer LNCSE, April 2019.

[29] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun, *What is the best multi-stage architecture for object recognition?*, 2009, pp. 2146–2153.

[30] G. Karypis and V. Kumar, *METIS: Unstructured graph partitioning and sparse matrix ordering system, version 4.0.*

[31] H. H. Kim and E. T. Chung, *A BDDC algorithm with enriched coarse spaces for two-dimensional elliptic problems with oscillatory and high contrast coefficients*, Multiscale Model. Simul., 13 (2015), pp. 571–593.

[32] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, arXiv preprint arXiv:1412.6980, (2014).

[33] A. Klawonn, M. Kühn, and O. Rheinbach, *Adaptive coarse spaces for FETI-DP in three dimensions*, SIAM J. Sci. Comput., 38 (2016), pp. A2880–A2911.

[34] A. Klawonn, M. Kühn, and O. Rheinbach, *Adaptive coarse spaces for FETI-DP in three dimensions with applications to heterogeneous diffusion problems*, in Domain decomposition methods in science and engineering XXIII, vol. 116 of Lect. Notes Comput. Sci. Eng., Springer, Cham, 2017, pp. 187–196.

[35] A. Klawonn, M. Kühn, and O. Rheinbach, *Adaptive FETI-DP and BDDC methods with a generalized transformation of basis for heterogeneous problems*, Electron. Trans. Numer. Anal., 49 (2018), pp. 1–27.

[36] A. Klawonn, M. J. Kühn, and O. Rheinbach, *Parallel adaptive FETI-DP using lightweight asynchronous dynamic load balancing*, International Journal for Numerical Methods in Engineering, 121 (2020), pp. 621–643, https://doi.org/10.1002/nme.6237.

[37] A. Klawonn, M. Lanser, and O. Rheinbach, *Toward extremely scalable nonlinear domain decomposition methods for elliptic partial differential equations*, SIAM J. Sci. Comput., 37 (2015), pp. C667–C696.

[38] A. Klawonn, M. Lanser, and O. Rheinbach, *A highly scalable implementation of inexact nonlinear FETI-DP without sparse direct solvers*, in Numerical mathematics and advanced applications—ENUMATH 2015, vol. 112 of Lect. Notes Comput. Sci. Eng., Springer, [Cham], 2016, pp. 255–264.

[39] A. Klawonn, M. Lanser, and O. Rheinbach, *Nonlinear BDDC methods with approximate solvers*, Electron. Trans. Numer. Anal., 49 (2018), pp. 244–273.

[40] A. Klawonn, M. Lanser, O. Rheinbach, and M. Uran, *Nonlinear FETI-DP and BDDC methods: a unified framework and parallel results*, SIAM J. Sci. Comput., 39 (2017), pp. C417–C451.

[41] A. Klawonn, P. Radtke, and O. Rheinbach, *FETI-DP methods with an adaptive coarse space*, SIAM J. Numer. Anal., 53 (2015), pp. 297–320.

[42] A. Klawonn, P. Radtke, and O. Rheinbach, *A comparison of adaptive coarse spaces for iterative substructuring in two dimensions*, Electron. Trans. Numer. Anal., 45 (2016), pp. 75–106.

[43] A. Klawonn and O. Rheinbach, *Robust FETI-DP methods for heterogeneous three dimensional elasticity problems*, Comput. Methods Appl. Mech. Engrg., 196 (2007), pp. 1400–1414.

[44] A. Klawonn and O. Rheinbach, *Highly scalable parallel domain decomposition methods with an application to biomechanics*, ZAMM Z. Angew. Math. Mech., 90 (2010), pp. 5–32.

[45] A. Klawonn and O. Rheinbach, *Deflation, projector preconditioning, and balancing in iterative substructuring methods: connections and new results*, SIAM J. Sci. Comput., 34 (2012), pp. A459–A484.

[46] A. Klawonn, O. Rheinbach, and O. B. Widlund, *Some computational results for Dual-Primal FETI methods for elliptic problems in 3D*, in Domain Decomposition Methods in Science and Engineering, Springer, 2005, pp. 361–368.

[47] A. Klawonn, O. Rheinbach, and O. B. Widlund, *An analysis of a FETI-DP algorithm on irregular subdomains in the plane*, SIAM J. Numer. Anal., 46 (2008), pp. 2484–2504.

[48] A. Klawonn and O. B. Widlund, *Dual-primal FETI methods for linear elasticity*, Comm. Pure Appl. Math., 59 (2006), pp. 1523–1572.

[49] A. Klawonn, O. B. Widlund, and M. Dryja, *Dual-primal FETI methods for three-dimensional elliptic problems with heterogeneous coefficients*, SIAM J. Numer. Anal., 40 (2002), pp. 159–179.

[50] M. J. Kühn, *Adaptive FETI-DP and BDDC methods for highly heterogeneous elliptic finite element problems in three dimensions*, PhD thesis, Universität zu Köln, Cologne, 2018.

[51] J. Li and O. B. Widlund, *FETI-DP, BDDC, and block Cholesky methods*, Internat. J. Numer. Methods Engrg., 66 (2006), pp. 250–271.

[52] J. Mandel and C. R. Dohrmann, *Convergence of a balancing domain decomposition by constraints and energy minimization*, Numer. Linear Algebra Appl., 10 (2003), pp. 639–659. Dedicated to the 70th birthday of Ivo Marek.

[53] J. Mandel, C. R. Dohrmann, and R. Tezaur, *An algebraic theory for primal and dual substructuring methods by constraints*, Appl. Numer. Math., 54 (2005), pp. 167–193.

[54] J. Mandel and B. Sousedík, *Adaptive selection of face coarse degrees of freedom in the BDDC and the FETI-DP iterative substructuring methods*, Comput. Methods Appl. Mech. Engrg., 196 (2007), pp. 1389–1399.

[55] J. Mandel, B. Sousedík, and J. Sístek, *Adaptive BDDC in three dimensions*, Math. Comput. Simulation, 82 (2012), pp. 1812–1831.

[56] V. Nair and G. E. Hinton, *Rectified linear units improve restricted boltzmann machines*, in Proceedings of the 27th int. conf. on machine learning (ICML-10), 2010, pp. 807–814.

[57] D.-S. Oh, O. B. Widlund, S. Zampini, and C. R. Dohrmann, *BDDC algorithms with deluxe scaling and adaptive selection of primal constraints for Raviart-Thomas vector fields*, Math. Comp., 87 (2018), pp. 659–692.

[58] C. Pechstein and C. R. Dohrmann, *A unified framework for adaptive BDDC*, Electron. Trans. Numer. Anal., 46 (2017), pp. 273–336.

[59] N. Spillane, V. Dolean, P. Hauret, F. Nataf, C. Pechstein, and R. Scheichl, *Abstract robust coarse spaces for systems of PDEs via generalized eigenproblems in the overlaps*, Numer. Math., 126 (2014), pp. 741–770.

[60] N. Spillane and D. J. Rixen, *Automatic spectral coarse spaces for robust finite element tearing and interconnecting and balanced domain decomposition algorithms*, Internat. J. Numer. Methods Engrg., 95 (2013), pp. 953–990.

[61] A. Toselli and O. Widlund, *Domain decomposition methods—algorithms and theory*, vol. 34 of Springer Series in Computational Mathematics, Springer-Verlag, Berlin, 2005.

[62] S. Zampini, *PCBDDC: a class of robust dual-primal methods in PETSc*, SIAM J. Sci. Comput., 38 (2016), pp. S282–S306.