SIAM J. Sci. Comput.
Vol. 41, No. 6, pp. A3887–A3912

MACHINE LEARNING IN ADAPTIVE DOMAIN DECOMPOSITION METHODS—PREDICTING THE GEOMETRIC LOCATION OF CONSTRAINTS[*]

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

Abstract. Domain decomposition methods are robust and parallel scalable, preconditioned iterative algorithms for the solution of the large linear systems arising in the discretization of elliptic partial differential equations by finite elements. The convergence rate of these methods is generally determined by the eigenvalues of the preconditioned system. For second-order elliptic partial differential equations, coefficient discontinuities with a large contrast can lead to a deterioration of the convergence rate. A remedy can be obtained by enhancing the coarse space with elements, which are often called constraints, that are computed by solving small eigenvalue problems on portions of the interface of the domain decomposition, i.e., edges in two dimensions or faces and edges in three dimensions. In the present work, without restriction of generality, the focus is on two dimensions. In general, it is difficult to predict where these constraints have to be added, and therefore the corresponding local eigenvalue problems have to be computed, i.e., on which edges. Here, a machine learning based strategy using neural networks is suggested to predict the geometric location of these edges in a preprocessing step. This reduces the number of eigenvalue problems that have to be solved before the iteration. Numerical experiments for model problems and realistic microsections using regular decompositions as well as decompositions from graph partitioners are provided, showing very promising results.

Key words. machine learning, domain decomposition, FETI-DP, adaptive coarse spaces

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

DOI. 10.1137/18M1205364

1. Introduction.

Domain decomposition methods are highly scalable, iterative, and robust implicit solvers for partial differential equations, which have been discretized using, e.g., finite elements. In general, a geometric decomposition of the computational domain into overlapping or nonoverlapping subdomains is used to construct a preconditioned system, which is solved by, e.g., the CG (conjugate gradient) or the GMRES (generalized minimal residual) method. Widely used representatives are, e.g., GDSW (generalized Dryja–Smith–Widlund) [11, 10], BDDC (balancing domain decomposition by constraints) [9, 50, 52, 51], and FETI-DP (finite element tearing and interconnecting-dual primal) [17, 16, 48, 49] methods. Parallel scalability up to tens or even hundreds of thousands of compute cores has been confirmed for several model problems [28, 66, 3, 2, 40, 45, 41, 39, 38, 29]. All these methods obtain their numerical robustness from a well-designed coarse space or, in other words, a second level. For classical coarse spaces, usually built by using information on the geome-

[*]Submitted to the journal's Methods and Algorithms for Scientific Computing section August 6, 2018; accepted for publication (in revised form) September 17, 2019; published electronically December 12, 2019.
https://doi.org/10.1137/18M1205364

Funding: This work was supported in part by the Deutsche Forschungsgemeinschaft (DFG) through the Priority Programme 1648 "Software for Exascale Computing" (SPPEXA) DFG project 230723766.

[†]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, http://www.numerik.uni-koeln.de).

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

try and the coefficient distribution, condition number bounds have been proven for a variety of model problems [48, 49, 44, 47, 55, 10, 12]. Thus, fast convergence of the iterative method can be guaranteed.

Considering more general problem settings, such as, e.g., diffusion or elasticity problems with heterogeneous coefficient functions having large jumps along or across the interface between subdomains or phenomena such as incompressibility and plastification in problems from solid mechanics, the classical condition number bounds do not hold anymore and the convergence of the classical domain decomposition approaches deteriorates. In recent years, several adaptive coarse space techniques have been developed to cope with these issues [5, 37, 36, 35, 59, 58, 4, 8, 53, 54, 43, 42, 33, 18, 19, 15, 13, 62, 63, 26, 24, 25, 20]. In these approaches, local eigenvalue problems on parts of the interface, e.g., edges or faces, are solved in advance and eigenvectors belonging to certain eigenvalues are used to automatically design a coarse space. For these coarse spaces, the domain decomposition approach is again robust with respect to arbitrary coefficient functions in diffusion or elasticity problems. Condition number bounds usually only depend on a given tolerance, which is reflected in the selection of the eigenvectors, and some geometric constants, such as, e.g., the maximum number of edges or faces in a subdomain, but is independent of the contrast of the coefficient function.

The eigenvalue problems occurring in the adaptive approaches are typically small and related only to a small number of neighboring subdomains; the exact number depends strongly on the specific approach. Hence, it is feasible to parallelize the solution of the different eigenvalue problems and thus the computation of the adaptive constraints. Nevertheless, on single compute cores, a significant number of subsequent eigenvalue problems can occur. Thus, in a parallel implementation, building the coarse space using adaptive techniques can make up the larger part of the total time to solution. However, for many realistic coefficient distributions, only a few adaptive constraints on a few edges or faces are necessary for a robust coarse space. Therefore, many of the eigenvalue problems are indeed unnecessary to be solved. Unfortunately, it is not known in advance which of the eigenvalue problems have to be considered. To decide which eigenvalue problems can be omitted, a heuristic approach based on the coefficient jumps as well as the residual after one step of the FETI-DP or BDDC method are already considered in [36, 37]; see also [42] for a first preliminary result in this direction. A heuristic approach for the detection of necessary eigenvalue problems and the construction of adaptive constraints without the solution of eigenvalue problems for overlapping Schwarz methods is proposed in [23, 26].

In the present paper, we discuss a different and more general approach. We propose to train a neural network to make the decision whether we have to solve a certain eigenvalue problem or not. Therefore, we introduce a sampling procedure for the coefficient function, which generates the input data for the neural network. The proposed sampling procedure is independent of the specific geometry and discretization and can therefore also be applied to unstructured meshes and domain decompositions. This approach can reduce the computational effort of adaptive domain decomposition methods while preserving the guaranteed robustness. In order to optimize the hyperparameters of the neural network, we apply a grid search procedure.

For a first discussion of our approach, we use a certain adaptive FETI-DP algorithm in two dimensions; see [53, 54]. Therefore, we consider eigenvalue problems on edges and train the machine learning algorithm accordingly. Our approach can also be applied to other adaptive coarse spaces and even to different domain decomposition approaches, e.g., the GDSW algorithm. Also three-dimensional problems will be considered in the near future.

This paper is organized as follows. In section 2, we first introduce classical and adaptive FETI-DP methods, followed, in section 3, by a description of our approach on how to predict, using a neural network, whether an eigenvalue problem has to be solved. We finally provide numerical results comparing all approaches, i.e., FETI-DP, adaptive FETI-DP, and adaptive FETI-DP, using our new strategy, denoted by ML-FETI-DP; see section 4. We consider several diffusion problems with various coefficient distributions and show the robustness of our approach for regular and irregular domain decompositions in two dimensions. Our computations are performed using the machine learning and data analysis implementations in TensorFlow [1] and Scikit-learn [60] and our MATLAB implementation of the adaptive FETI-DP method applied in the present work.

2. Algorithms and model problems.

In this section, we provide a brief description of our model problem and the classical FETI-DP (finite element tearing and interconnecting-dual primal) [17, 16, 48, 49] method. Finally, in subsection 2.3.2, we summarize the construction of an adaptive FETI-DP coarse space following [53, 54].

2.1. A simple model problem.

As a model problem, we consider a diffusion problem in variational form with various and a highly heterogeneous diffusion coefficient functions $\rho : \Omega \to \mathbb{R}$. We always choose $\Omega = [0,1] \times [0,1] \subset \mathbb{R}^2$ and a homogeneous Dirichlet boundary condition on the complete boundary $\partial\Omega$ throughout this paper. Therefore, the model problem writes as follows: Find $u \in H_0^1(\Omega)$ such that

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

Examples of different coefficient functions are discussed in detail in the section on numerical results; see section 4.

As a second model problem, we consider a linear elasticity problem. The problem of linear elasticity consists in finding the displacement $\mathbf{u} \in \mathbf{H}_0^1(\Omega) := (H_0^1(\Omega))^2$, such that

$$\int_\Omega G \; \varepsilon(\mathbf{u}) : \varepsilon(\mathbf{v}) \; d\mathbf{x} + \int_\Omega G\beta \; \mathrm{div}\mathbf{u} \; \mathrm{div}\mathbf{v} \; d\mathbf{x} = \langle \mathbf{F}, \mathbf{v} \rangle \tag{2.2}$$

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 the right-hand side

$$\langle \mathbf{F}, \mathbf{v} \rangle = \int_\Omega \mathbf{f}^T \mathbf{v} \; d\mathbf{x} + \int_{\partial\Omega_N} \mathbf{g}^T \mathbf{v} \; d\sigma.$$

See, e.g., [6] or [7] for more details.

Discretizing (2.1) and (2.2) with finite elements, we obtain the 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 $u_g, \, f_g \in V^h$.

2.2. Standard FETI-DP.

Let us briefly describe the standard FETI-DP domain decomposition method and introduce some relevant notation.

2.2.1. Domain decomposition. We assume a decomposition of Ω into $N \in \mathbb{N}$ nonoverlapping subdomains $\Omega_i,\ i = 1, \ldots, N$, i.e., $\overline{\Omega} = \bigcup_{i=1}^N \overline{\Omega}_i$. Each of the subdomains is the union of finite elements such that we have matching finite element nodes on the interface $\Gamma := \left(\bigcup_{i=1}^N \partial\Omega_i\right) \setminus \partial\Omega$. We denote by $W^{(i)}$ the local finite element space associated with Ω_i. The finite element nodes on the interface are either vertex nodes, belonging to the boundary of more than two subdomains, or edge nodes, belonging to the boundary of exactly two subdomains. All finite element nodes inside a subdomain Ω_i are denoted as interior nodes.

2.2.2. The FETI-DP algorithm. In this section, we will briefly describe the standard FETI-DP algorithm; for more details, see, e.g., [64, 48, 45]. Here, we follow the compact presentation in [43].

For each subdomain Ω_i, we subassemble the corresponding finite element stiffness matrix $K^{(i)}$. We partition the finite element variables $u^{(i)} \in W^{(i)}$ into interior variables $u_I^{(i)}$, and on the interface into dual variables $u_\Delta^{(i)}$ and primal variables $u_\Pi^{(i)}$. In the present article, we always choose the primal variables as those belonging to vertices. Hence, the dual variables always belong to edges. Note that other choices are possible. For each local stiffness matrix $K^{(i)}$, ordering the interior variables first, followed by the dual and primal variables, yields

$$K^{(i)} = \begin{bmatrix} K_{II}^{(i)} & K_{\Delta I}^{(i)T} & K_{\Pi I}^{(i)T} \\ K_{\Delta I}^{(i)} & K_{\Delta\Delta}^{(i)} & K_{\Pi\Delta}^{(i)T} \\ K_{\Pi I}^{(i)} & K_{\Pi\Delta}^{(i)} & K_{\Pi\Pi}^{(i)} \end{bmatrix}, \quad u^{(i)} = \begin{bmatrix} u_I^{(i)} \\ u_\Delta^{(i)} \\ u_\Pi^{(i)} \end{bmatrix}, \quad \text{and } f^{(i)} = \begin{bmatrix} f_I^{(i)} \\ f_\Delta^{(i)} \\ f_\Pi^{(i)} \end{bmatrix}.$$

It is often also convenient to introduce the union of interior and dual degrees of freedom as an extra set of degrees of freedom denoted by the index B. This leads to a more compact notation, and we can define the following matrices and vectors

$$K_{BB}^{(i)} = \begin{bmatrix} K_{II}^{(i)} & K_{\Delta I}^{(i)T} \\ K_{\Delta I}^{(i)} & K_{\Delta\Delta}^{(i)} \end{bmatrix}, \quad K_{\Pi B}^{(i)} = \begin{bmatrix} K_{\Pi I}^{(i)} & K_{\Pi\Delta}^{(i)} \end{bmatrix}, \quad \text{and } f_B^{(i)} = \begin{bmatrix} f_I^{(i)T} & f_\Delta^{(i)T} \end{bmatrix}^T.$$

Next, we introduce block diagonal matrices $K_{BB} = \operatorname{diag}_{i=1}^N K_{BB}^{(i)}$, $K_{II} = \operatorname{diag}_{i=1}^N K_{II}^{(i)}$, $K_{\Delta\Delta} = \operatorname{diag}_{i=1}^N K_{\Delta\Delta}^{(i)}$, and $K_{\Pi\Pi} = \operatorname{diag}_{i=1}^N K_{\Pi\Pi}^{(i)}$. Analogously, we obtain the block vector $u_B = [u_B^{(1)T}, \ldots, u_B^{(N)T}]^T$ and the block right-hand side $f_B = \left[f_B^{(1)T}, \ldots, f_B^{(N)T}\right]^T$.

To enforce continuity in the primal variables in each iteration of the FETI-DP algorithm, we assemble in those primal variables; this introduces a global coupling in a small number of degrees of freedom. To describe this assembly process, we introduce assembly operators $R_\Pi^{(i)T}$ consisting only of zeros and ones. This yields the matrices $\widetilde{K}_{\Pi\Pi} = \sum_{i=1}^N R_\Pi^{(i)T} K_{\Pi\Pi}^{(i)} R_\Pi^{(i)}$, $\widetilde{K}_{\Pi B} = \left[R_\Pi^{(1)T} K_{\Pi B}^{(1)}, \ldots, R_\Pi^{(N)T} K_{\Pi B}^{(N)}\right]$ and the right-hand side $\widetilde{f} = \left[f_B^T, \left(\sum_{i=1}^N R_\Pi^{(i)T} f_\Pi^{(i)}\right)^T\right]^T$. Since no assembly is carried out in the dual degrees of freedom, we need a continuity condition on this part of the interface. Hence, we introduce a jump matrix $B_B = [B_B^{(1)} \ldots B_B^{(N)}]$ with $B_B^{(i)}$ having zero entries for the interior degrees of freedom and entries out of $\{-1, 1\}$ for the dual degrees of freedom. The entries for the dual degrees of freedom are chosen such that $B_B u_B = 0$ if u_B is continuous across the interface. This continuity condition is enforced by Lagrange multipliers λ. Then, we consider

$$\begin{pmatrix} K_{BB} & \widetilde{K}_{\Pi B}^T & B_B^T \\ \widetilde{K}_{\Pi B} & \widetilde{K}_{\Pi\Pi} & O \\ B_B & O & O \end{pmatrix} \begin{pmatrix} u_B \\ \tilde{u}_\Pi \\ \lambda \end{pmatrix} = \begin{pmatrix} f_B \\ \tilde{f}_\Pi \\ 0 \end{pmatrix}. \tag{2.4}$$

Solving (2.4) and assembling u_B then gives the solution of (2.3). To solve (2.4), the variables u_B and $\tilde{u}_\Pi$ are eliminated, resulting in a linear system for the Lagrange multipliers λ. This is carried out in two steps, first eliminating u_B and then $\tilde{u}_\Pi$. The local elimination of u_B yields the following Schur complement for the primal variables $\widetilde{S}_{\Pi\Pi} = \widetilde{K}_{\Pi\Pi} - \widetilde{K}_{\Pi B} K_{BB}^{-1} \widetilde{K}_{\Pi B}^T$. The FETI-DP system is then defined as

$$F\lambda = d, \tag{2.5}$$

with

$$\begin{aligned} F &= B_B K_{BB}^{-1} B_B^T + B_B K_{BB}^{-1} \widetilde{K}_{\Pi B}^T \widetilde{S}_{\Pi\Pi}^{-1} \widetilde{K}_{\Pi B} K_{BB}^{-1} B_B^T \\ \text{and } d &= B_B K_{BB}^{-1} f_B + B_B K_{BB}^{-1} \widetilde{K}_{\Pi B}^T \widetilde{S}_{\Pi\Pi}^{-1} \left(\left(\sum_{i=1}^{N} R_\Pi^{(i)T} f_\Pi^{(i)} \right) - \widetilde{K}_{\Pi B} K_{BB}^{-1} f_B \right). \end{aligned}$$

To define the FETI-DP algorithm, we also need a preconditioner for the FETI-DP system matrix F. In the present work, we use the Dirichlet preconditioner given by

$$M_D^{-1} = B_{B,D} \left[0 \quad I_\Delta\right]^T \left(K_{\Delta\Delta} - K_{\Delta I} K_{II}^{-1} K_{\Delta I}^T \right) \left[0 \quad I_\Delta\right] B_{B,D}^T = B_D \widetilde{S} B_D^T.$$

Here, I_Δ is the identity matrix on the dual degrees of freedom. The matrices $B_{B,D}$ and B_D are scaled variants of B_B and B, respectively; in the simplest case, they are scaled by the inverse multiplicity of the nodes, e.g., $1/2$ in two dimensions. This is also denoted as multiplicity scaling. In order to obtain more robustness with respect to coefficient jumps, we also consider the ρ-scaling approach; see, e.g., [44]. Therefore, we first introduce the following notation. For each finite element node $x \in \partial\Omega_j \cap \Gamma$, $j = 1, \ldots, N$, we denote by $\omega(x)$ the support of the finite element basis functions associated with x. Next, we introduce scaling weights

$$d_j(x) := \rho_j(x) / \sum_{i \in N_x} \rho_i(x),$$

where $\rho_j(x) := \max_{y \in \omega(x) \cap \Omega_j} \rho(y)$ and N_x denotes for each interface node x the set of indices of subdomains which have x on their boundaries. Each row of $B^{(i)}$ with a nonzero entry connects a finite element node on $\Gamma^{(i)}$ with the corresponding finite element node of a neighboring subdomain $x \in \Gamma^{(i)} \cap \Gamma^{(j)}$. Multiplying each such row with $d_j(x)$ for each $B^{(i)}$, $i = 1, \ldots, N$, results in the scaled operator B_D. We will refer to this scaling as ρ-scaling. For coefficients that are constant on each subdomain but possibly discontinuous across the interface, this approach reduces to the classical ρ-scaling; see, e.g., [64]. In general, we can write $B_D = [D^{(1)T} B^{(1)} \ldots D^{(N)T} B^{(N)}]$ with scaling matrices $D^{(i)}$ defined accordingly. Note that there also exist nondiagonal scaling matrices, e.g., resulting from deluxe scaling; see [43] and references therein.

2.2.3. Condition number bound. For scalar elliptic as well as various other model problems, e.g., linear elasticity problems, the polylogarithmic condition number bound

$$\kappa(M_D^{-1} F) \le C \left(1 + \log\left(\frac{H}{h} \right) \right)^2 \tag{2.6}$$

holds under certain assumptions; see, e.g., [47, 49, 48]. In (2.6), H/h is the maximum of $H_i/h_i, i = 1, \ldots, N$, where H_i is the diameter of Ω_i, h_i is the maximum finite element diameter in Ω_i, and thus H/h is a measure for the number of finite elements per subdomain and thus for the number of unknowns in each subdomain. The constant C is independent of H_i and h_i. Different coefficient functions ρ in two and three dimensions can be successfully treated by appropriate coarse spaces and scalings in the preconditioner M_D^{-1}. For further details, see, e.g., [64]. For our model problem, using only primal vertex constraints and ρ-scaling, the constant C is independent of ρ, e.g., if ρ is constant on the complete domain, if ρ is constant on subdomains but discontinuous across the interface, or if inclusions of higher coefficients are completely enclosed in single subdomains without touching the interface.

However, for arbitrary and complex coefficient distributions, (2.6) does not hold anymore. In recent years, adaptive coarse spaces have been developed to overcome this limitation [5, 37, 36, 35, 59, 58, 4, 8, 53, 54, 43, 42, 33, 18, 19, 15, 13, 62, 63, 26, 24, 25, 20]. In these algorithms, additional coarse modes or primal constraints are computed by solving localized eigenvalue problems on edges, local interfaces, or subdomains. The FETI-DP coarse space is then enriched with these additional primal constraints before the iteration starts.

2.3. FETI-DP with an adaptive coarse space. We will now describe how additional primal constraints can be implemented in FETI-DP and, subsequently, the specific adaptive approach we discuss in this paper. In general, there are several approaches to enforce coarse constraints in FETI-DP. Common are a transformation of basis [48, 45] or a deflation approach [46, 43]. We use the latter one. Let us remark that subsections 2.3.1 to 2.3.3 are based on [43].

2.3.1. Enhancing the coarse space using a balancing preconditioner. A set of additional primal constraints, such as, e.g., averages or first-order moments over certain edges, can be aggregated as columns of a matrix U; see, e.g., [46, 30]. To enforce $U^T Bu = 0$, e.g., averages of the jump with weights defined by the columns of U, we introduce the F-orthogonal projection $P = U(U^T FU)^{-1}U^T F$. The deflated and singular but consistent system $(I-P)^T F\lambda = (I-P)^T d$ now replaces the original system $F\lambda = d$ (see (2.5)). Let λ^* be the exact solution of $F\lambda = d$ and λ the solution of $M_D^{-1}(I-P)^T F\lambda = M_D^{-1}(I-P)^T d$ which has been obtained by applying the PCG (preconditioned conjugate gradient) method. We then define

$$\overline{\lambda} = U(U^T FU)^{-1}U^T d = PF^{-1}d = P\lambda^*$$

and compute $\lambda^* = \overline{\lambda} + (I-P)\lambda \in \ker(I-P) \oplus \operatorname{range}(I-P)$. The matrices $P^T F\,(= FP)$ and $(I-P)^T F\,(= F(I-P))$ are symmetric, and the spectrum is thus not changed by projecting the correction onto range$(I-P)$ in each iteration; cf. [46]. Therefore, we obtain the symmetric projector preconditioner

$$M_{PP}^{-1} = (I-P)M_D^{-1}(I-P)^T.$$

Adding the correction, we compute $\lambda^* = \overline{\lambda} + \lambda$, where λ is the PCG solution of $M_{PP}^{-1}F\lambda = M_{PP}^{-1}d$. An alternative approach is the inclusion of the computation of $\overline{\lambda}$ into the preconditioner. This results in the balancing preconditioner

$$M_{BP}^{-1} = (I-P)M_D^{-1}(I-P)^T + PF^{-1}. \tag{2.7}$$

Since $PF^{-1} = U(U^T FU)^{-1}U^T$, this preconditioner is symmetric and can be efficiently computed. Here, depending on the number of additional primal constraints, $U^T FU$ is usually of much smaller dimension than F. Throughout this paper, we use the balancing preconditioner in all our numerical experiments and all constraints added to the a priori vertex constraints are included using U.

2.3.2. The adaptive constraints. Here, we consider an approach which has been successfully used in FETI-DP and BDDC for some time [53, 54, 43]. In the following, we give a brief description of the algorithm introduced in [53] for the convenience of the reader. First, we introduce the relevant notation and the eigenvalue problem on an edge. Second, in subsection 2.3.3, we give an estimate of the condition number for two-dimensional problems where all the vertex variables are primal in the initial coarse space; this, in particular, holds for our model problem (2.1).

For each edge, a single eigenvalue problem has to be solved. Let E_{ij} be the edge between subdomains Ω_i and Ω_j. We first restrict the jump matrix B to this edge. Let $B_{E_{ij}} = \left(B^{(i)}_{E_{ij}},\ B^{(j)}_{E_{ij}}\right)$ be the submatrix of $\left(B^{(i)},\ B^{(j)}\right)$ with the rows that consist of exactly one 1 and one -1 and are zero otherwise. Let $B_{D,E_{ij}} = \left(B^{(i)}_{D,E_{ij}},\ B^{(j)}_{D,E_{ij}}\right)$ be obtained by taking the same rows of $\left(B^{(i)}_{D},\ B^{(j)}_{D}\right)$; see the end of subsection 2.2.2 for the definition of B_D. Let

$$S_{ij} = \begin{pmatrix} S^{(i)} & \\ & S^{(j)} \end{pmatrix},$$

where $S^{(i)}$ and $S^{(j)}$ are the Schur complements of $K^{(i)}$ and $K^{(j)}$, respectively, with respect to the interface variables. We further define the operator $P_{D_{ij}} = B^T_{D,E_{ij}} B_{E_{ij}}$.

Then, we solve the following local generalized eigenvalue problem: find $w_{ij} \in (\operatorname{Ker} S_{ij})^{\perp}$ such that

$$\langle P_{D_{ij}} v_{ij},\ S_{ij} P_{D_{ij}} w_{ij}\rangle = \mu_{ij} \langle v_{ij},\ S_{ij} w_{ij}\rangle \quad \forall v_{ij} \in (\operatorname{Ker} S_{ij})^{\perp}. \tag{2.8}$$

For an explicit expression of the positive definite right-hand side operator on the subspace $(\operatorname{Ker} S_{ij})^{\perp}$, two orthogonal projections are used; see, e.g., [43]. We assume that R eigenvectors w^r_{ij}, $r = 1, \ldots, R$, belong to eigenvalues which are larger than a given tolerance TOL. Then, we enhance the FETI-DP coarse space with the constraints $B_{D_{ij}} S_{ij} P_{D_{ij}} w^r_{ij}$, $r = 1, \ldots, R$, using the balancing preconditioner described above.

2.3.3. Condition number bound. Computing adaptive constraints as presented in subsection 2.3.2 and enhancing the FETI-DP coarse space with these constraints using a balancing preconditioner M^{-1}_{BP}, we obtain the condition number bound

$$\kappa(M^{-1}_{BP} F) \leq N_E^2 \mathrm{TOL},$$

which was first proved in [43, Theorem 5.1]. Here, N_E is the maximum number of edges of a subdomain.

2.3.4. Alternative edge constraints. Later, in subsection 3.3, we will introduce an approach which distinguishes between three different classes of edges: edges where no additional constraint is necessary, edges where only one single additional constraint is necessary, and edges where more than a single constraint has to be added. For the second class, we replace the eigenvalue problem and the resulting eigenvector by a single edge constraint designed using ρ. Let us briefly describe these constraints. For each finite element node x on E_{ij}, we compute $\rho^{(i)}(x) = \max_{y \in \omega(x) \cap \Omega_i} \rho(y)$ and $\rho^{(j)}(x) = \max_{y \in \omega(x) \cap \Omega_j} \rho(y)$. Now, we define $v^{(i)}_{E_{ij}}$ and $v^{(j)}_{E_{ij}}$ on $\partial\Omega_i$ and $\partial\Omega_j$, respectively, by

$$v^{(i)}_{E_{ij}}(x) = \begin{cases} \rho^{(i)}(x), & x \in E_{ij}, \\ 0 & \text{elsewhere} \end{cases} \quad \text{and} \quad v^{(j)}_{E_{ij}}(x) = \begin{cases} \rho^{(j)}(x), & x \in E_{ij}, \\ 0 & \text{elsewhere.} \end{cases}$$

Defining $v_{E_{ij}}^T := (v_{E_{ij}}^{(i)T}, -v_{E_{ij}}^{(j)T})$, we obtain the edge constraint $B_{D_{ij}} S_{ij} P_{D_{ij}} v_{E_{ij}}$ and add it to the coarse space using again the balancing approach. This coarse space itself can be interpreted as a generalization of the weighted edge averages suggested in [44]. It can be combined with an arbitrary FETI-DP scaling and is robust for a broader range of heterogeneities; see [27] for a detailed discussion.

2.3.5. Computational effort. In a parallel FETI-DP implementation, the solution of the eigenvalue problems on the edges can be distributed to the compute cores, due to the local nature of the eigenvalue problems. Nonetheless, more than one single eigenvalue problem has to be solved on several compute cores. The subsequent assembly and solution of several eigenvalue problems can take up the larger part of the total time to solution, and also the sending of Schur complements, which is in general necessary, can put a high pressure on the network of the parallel computer. The assembly of the local eigenvalue problems includes the computation of local Schur complements, which, in our experience, takes up a significant part of the total computing time. In our implementation of the FETI-DP method, the construction of these Schur complements is therefore avoided and only necessary for the computation of the adaptive coarse space. Therefore, it is beneficial to reduce the number of necessary eigenvalue problems to a necessary minimum, e.g., by filtering out eigenvalue problems which do not add any new constraints for a given tolerance TOL. A heuristic approach based on the coefficient jumps on the considered faces and edges as well as the residual after one step of the FETI-DP or BDDC method is already considered in [36, 37]. Our approach is to train a neural network to make this decision automatically.

3. Machine learning.

From a high-level point of view, supervised machine learning models approximate nonlinear functions, which associate input and output data:

$$F : I \to O.$$

Here, the input space I can be a product of $\mathbb{R}$, $\mathbb{N}$, and Boolean vector spaces. On the other hand, the output space is typically either an $\mathbb{R}$ vector space for regression problems or an $\mathbb{N}$ vector space for classification problems.

In order to compute a machine learning model, a large set of a priori known data is necessary. This data is typically partitioned into training and validation data. In the training and optimization phase, the model is trained to fit the training data while the validation data is used to control the generalization properties of the model, i.e., to ensure that the model is not fitted too closely to the training data but also able to accurately predict the output for new input data; cf., e.g., [65, sect. 6.4] and [56, pp. 25–29]. In that way, over- or underfitting should be minimized. Finally, the model can be evaluated for new input data to predict the unknown output.

While the training of machine learning models can be computationally very expensive, the evaluation of the model is typically cheap. In particular, the training of a machine learning model corresponds to a nonlinear high-dimensional optimization problem. However, the training can be performed a priori in an offline phase and the resulting model is then saved for online use.

In this work, we will focus on the use of dense feedforward neural networks or, more precisely, multilayer perceptrons; see, e.g., [22, Chap. 4], [56, pp. 104–119], and [65, sect. 5.1.4]. A graphical representation of neural networks is depicted in Figure 1. A feedforward neural network can be interpreted as an acyclic directed graph $\mathcal{G} =$

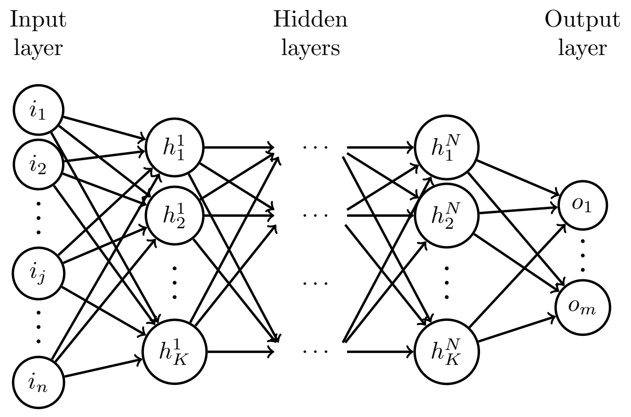

FIG. 1. *Structure of feedforward neural networks with N hidden layers and K neurons per layer.*

$(\mathcal{V}, \mathcal{E})$ with a set of nodes $\mathcal{V}$, a set of edges $\mathcal{E}$, and a weight function $w : \mathcal{E} \to \mathbb{R}$; see, e.g., [61, Chap. 20.1]. The neural network is assumed to be organized in layers; i.e., the set of nodes $\mathcal{V}$ can be represented as the union of nonempty, disjoint subsets $\mathcal{V}_i \subset \mathcal{V}, i = 0, \ldots, N+1$. These sets are defined such that for each edge $e \in \mathcal{E}$ there exists an $i \in \{0, \ldots, N\}$ with e being an edge between a node in $\mathcal{V}_i$ and one in $\mathcal{V}_{i+1}$; see [61, Chap. 20.1]. The nodes in a neural network are called neurons and, in dense feedforward neural networks, each neuron (in a chosen layer) is influenced by all neurons from the previous layer. In particular, the relation between two consecutive layers is the conjunction of a linear mapping and a nonlinear activation function. Among the many different choices for the activation function α, we choose the rectified linear unit (ReLU) [31, 57, 21] given by

$$\alpha(x) = \max\{0, x\}.$$

This function is almost linear, and its evaluation is very cheap. However, its nonlinearity is sufficient for the approximation of many nonlinear relations. Consequently, the output of the kth layer of the neural network can be written as

$$y = \alpha^k(x, W^k, b^k) = \max\left\{0, (W^k)^T x + b^k\right\},$$

where $W^k = (w_{ij}^k)_{i,j}$ and b^k are the weight matrix and the bias vector, respectively. Note that an entry w_{ij}^k of W^k corresponds to the value of the weight function w associated with the corresponding edge between layer $\mathcal{V}_{k-1}$ and $\mathcal{V}_k$. Then, the application of a complete neural network with N hidden layers to an input vector $i \in I$ is given by

$$\begin{aligned} h^1 &= \alpha^1(i, W^1, b^1), \\ h^{k+1} &= \alpha^{k+1}(h^k, W^{k+1}, b^{k+1}), \quad 1 \le k < N, \\ o &= (W^{N+1})^T h^N + b^{N+1}, \end{aligned}$$

where h^k is the output of the kth hidden layer and $o \in O$ is the (final) output vector. The computation of the output vector o is performed without an additional application of the activation function. Since we use dense neural networks, all entries of the matrices w and W^k, $k = 1, \ldots, N$, are nonzero, but using a dropout rate, a certain number of randomly chosen entries are set to zero.

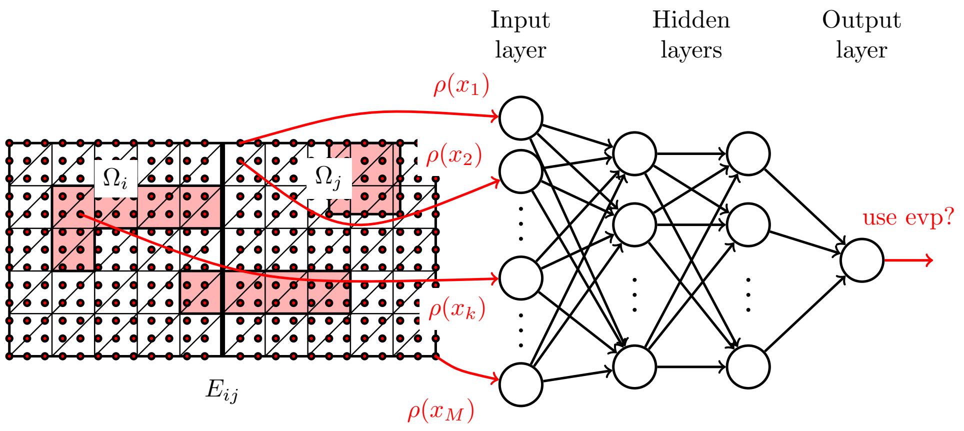

FIG. 2. *Sampling of the coefficient function ρ; white color corresponds to a low coefficient and red color to a high coefficient. In this representation, the samples are used as input data for a neural network with two hidden layers. Color is available online only.*

In our software framework, we employ machine learning and data analysis implementations in TensorFlow [1] and Scikit-learn [60].

3.1. Detecting critical edges using a neural network. Although the construction and solution of the local eigenvalue problems in adaptive domain decomposition solvers can be parallelized quite well, this part typically consumes a significant portion of the total runtime. However, for many realistic coefficient distributions, far fewer adaptive coarse constraints are needed than local generalized eigenvalue problems have to be solved. Therefore, we introduce a preprocessing step to identify the critical edges of the adaptive FETI-DP algorithm using a classification neural network. Consequently, we only compute the local eigenvalue problems on edges which are classified as critical by the neural network. On all uncritical edges, we do not enforce any constraints.

As input for the neural network, we use samples of the coefficient function within the two subdomains adjacent to an edge; cf. Figure 2. As output, we obtain the information whether an adaptive coarse constraint has to be computed on the corresponding edge or not.

Our sampling is independent of the underlying finite element discretization since we use a fixed number of sampling points for all mesh resolutions. In particular, we choose the sampling to be finer than all meshes used in our computations. As a rule of thumb, we assume that the sampling grid resolves all geometric details of the coefficient function. The location and order of the sampling points are depicted in Figure 3; note that other orderings are possible as well. In this way, the input vector for the neural network is of fixed length, i.e., number of sampling points in direction of the edge times number of sampling points orthogonal to the edge, and all input values are real numbers. In addition to that, we scale all input values using a min-max-scaling before the training of the neural network. Thus, we obtain input values which range only between zero and one. This is beneficial for the use in a neural network.

Since our sampling grid is oriented to the tangential and orthogonal direction of the edge, our sampling strategy is not restricted to the case of square subdomains, as indicated in Figure 2, but can also be applied to more general geometries. In this case,

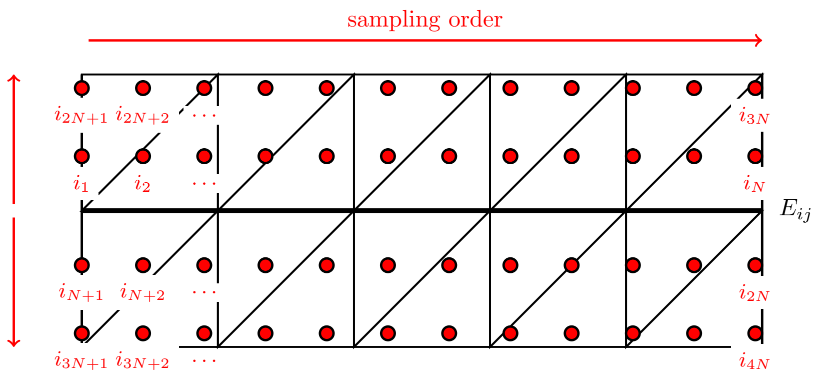
sampling order
i_{2N+1} i_{2N+2} $\cdots$ i_{3N}
i_1 i_2 $\cdots$ i_N
E_{ij}
i_{N+1} i_{N+2} $\cdots$ i_{2N}
i_{3N+1} i_{3N+2} $\cdots$ i_{4N}

FIG. 3. *Sampling of the coefficient function ρ for an edge e_{ij} between the subdomains Ω_i and Ω_j. The samples are first ordered in direction of the edge and second in orthogonal direction of the edge; we alternate sampling on the left-hand and right-hand sides of the edge.*

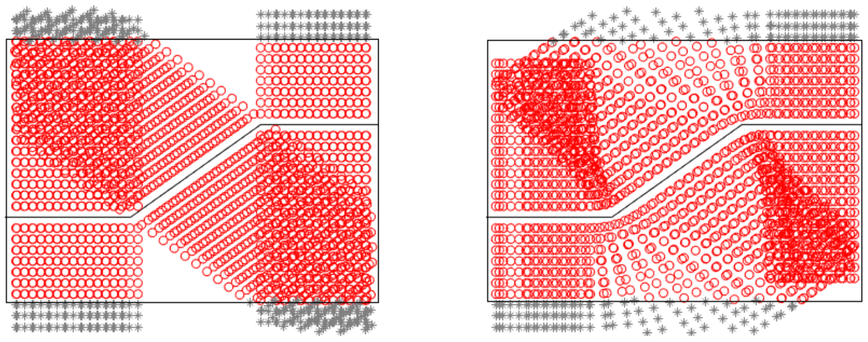

FIG. 4. *Sampling points for an irregular edge without smoothing the edge (left) and using the smoothing strategy shown in Figure 5 (right).*

we make sure to only use sampling points within the two subdomains adjacent to the edge in order to reflect the structure of the edge eigenvalue problems; cf. Figure 4. For all sampling points which fall outside the two subdomains, we use zero as input data.

However, nonsmooth edges may lead to gaps within the sampling grid; cf. Figure 4 (left). Therefore, we use a moving average to smooth out discontinuities in the tangential and normal vectors of the edge. In particular, we use a fixed window length of five sampling points and slide this window stepwise along the edge while computing the average of the subset of sampling points in each local window. As shown in Figure 5, we smooth out kinks twice using a moving average recursively. In fact, instead of applying the moving average to the full edge, it is sufficient to consider a neighborhood of a kink. To identify all kinks, we compute an approximation of the discrete second derivative for the entire edge.

3.2. Training and validation phase. For the training and validation of the neural network, we use a data set containing a total of 4 500 configurations varying the coefficient function and the edge geometry for two subdomains sharing this edge. To generate the output data that is necessary for the training of the neural network, we solve the eigenvalue problem described in subsection 2.3.2 for each of the 4 500

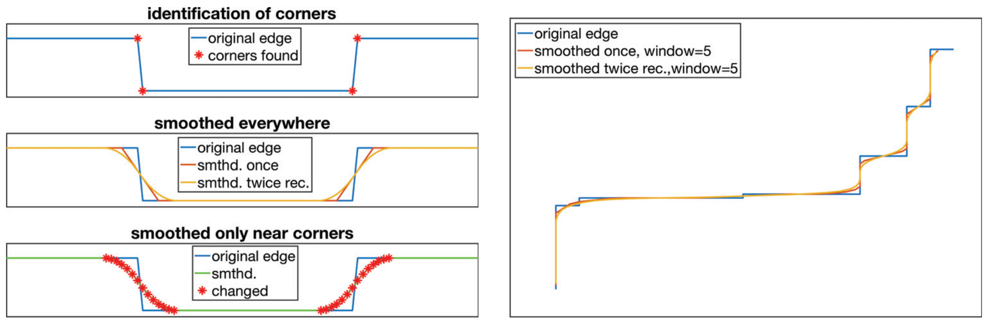

FIG. 5. Left: *Smoothing a jagged edge by identifying the kinks and using a moving average close to the kinks of the edge.* Right: *Smoothing of an example for an irregular edge.*

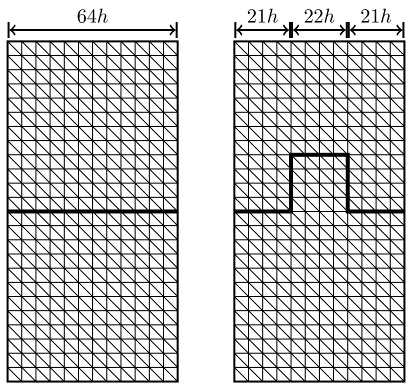

FIG. 6. *Geometric configurations used in the training data: straight and jagged edges; subdomain size $H/h = 64$.*

training and validation configurations.

As it turns out, it is sufficient to train on two geometric configurations, i.e., two regular subdomains sharing a straight edge and two regular subdomains sharing a jagged edge (see Figure 6), in order to generalize to arbitrary shapes of subdomains; cf. subsections 4.2.2 and 4.3.2. For the sampling, we select 127 points in the direction of the edge and 2×127 points in the orthogonal direction. Thus, we roughly have two sampling points in each finite element because here the subdomain size is defined by $H/h = 64$. We combine the two geometric configurations depicted in Figure 6 with coefficient functions of the types depicted in Figure 7. These coefficient functions are inspired by the coefficient functions used in [43, 26]. In order to obtain the full set of training data, the inclusions, channels, boxes, and combs with high coefficient are varied in size, location, and orientation, actually leading to more configurations than the nine basic ones given in Figure 7.

In the training of the neural network, we minimize the softmax cross-entropy loss function

$$g(b^1, \ldots, b^{N+1}, W^1, \ldots, W^{N+1}) = \sum_{c=1}^{C} \sum_{p \in \omega_c} \left(\log \left(\sum_{j=1}^{C} e^{o_j(p)} \right) - o_c(p) \right)$$

with respect to the weights and bias vectors of the neural network; cf. [65, sect. 6.3].

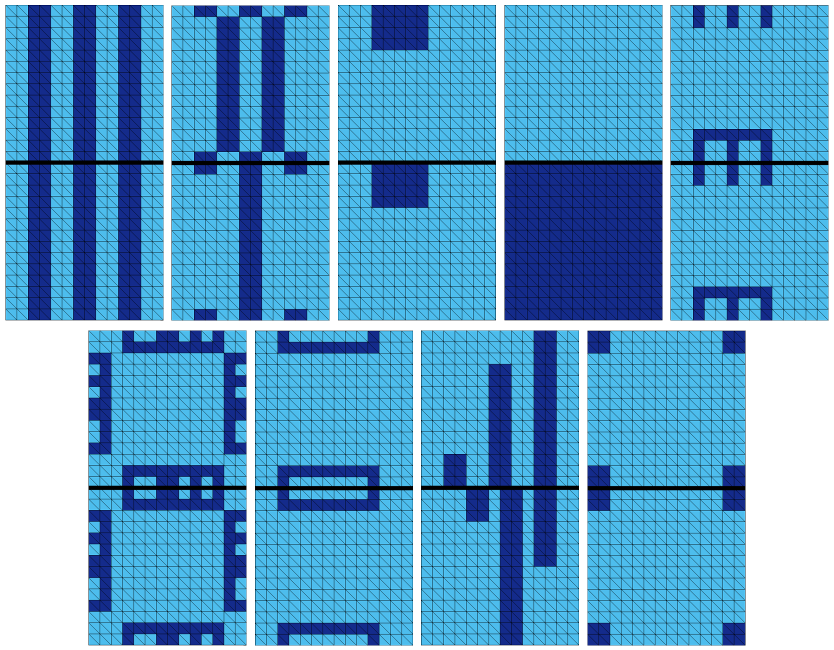

FIG. 7. *Nine different types of coefficient functions used for training and validation of the neural network. The inclusions, channels, boxes, and combs with high coefficient are displaced, modified in sized, and mirrored with respect to the edge in order to generate the complete training data set.*

Here, C is the total number of classes in the classification problem, $o_j(p)$ is the output corresponding to class j in the output vector of data p, and ω_c is the subset of the training data corresponding to class c. Thus, the softmax cross-entropy loss function minimizes the cross-entropy between the deterministic class labels of the training and validation data and the model's prediction for the same data. Minimizing the cross-entropy is equivalent to minimizing the Kullback–Leibler divergence, which is a measure for the difference of two probability distributions from information theory; cf., e.g., [22, sect. 3.13]. Note that the prediction

$$P(p \in \omega_c) = \frac{e^{o_c(p)}}{\sum_{j=1}^{C} e^{o_j(p)}}$$

is the probability that the input with index p belongs to class c.

In this nonlinear optimization problem, we apply a stochastic gradient descent (SGD) method with an adaptive scaling of the learning rate and a batch size of 100. For the adaptive scaling of the learning rates, we consider the AdaGrad (adaptive gradient) [14] and the Adam (adaptive moments) [34] algorithm.

In order to optimize the hyperparameters of the neural network, we apply a grid search algorithm on a discrete search space of parameters. Here, we use as hyperparameters the number of hidden layers, the number of neurons per layer, the dropout rate, the learning rate, and the optimization algorithm. The corresponding hyper parameters, the search space, and the optimal choices for the parameters are given in Table 1. We compare and optimize the generalization properties of the neural network by cross-validation using a random splitting of our data set into 80 % training and 20 % validation data for each iteration of the grid search algorithm. The receiver

TABLE 1
Hyperparameters for the neural network and its training and the optimal choice obtained from a grid search.

Hyperparameter	Range tested by grid search	Optimal choice
# hidden layers	{1, 2, 3, 4}	3
# neurons per layer	{10, 20, 30, 50}	30
dropout rate	{0, 0.2, 0.25, 0.5}	0.2
learning rate	{0, 0.001, 0.005, 0.01, 0.1, 1}	0.01
optimization algorithm	{Adam, AdaGrad}	Adam

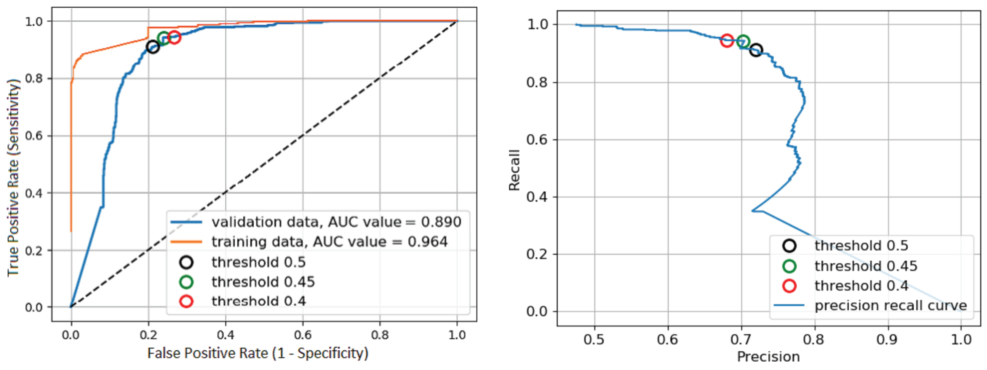

FIG. 8. *ROC curve and precision-recall plot for the optimal model obtained by a grid search; cf. Table* 1. *We define precision as true positives divided by (true positives+false positives), and recall as true positives divided by (true positives+false negatives). The thresholds used in section* 4 *are indicated as circles.*

operating characteristic (ROC) curve and a precision-recall plot of the neural network with optimal hyperparameters are shown in Figure 8. In both plots, the threshold τ for the decision boundary between critical and uncritical edges is varied between zero and one. When increasing τ, the false positive rate, which corresponds to the number of critical edges that are not detected by the algorithm, decreases. Consequently, the robustness of our ML-FETI-DP approach is improved. In Figure 8, we also indicate the thresholds used in the numerical tests in section 4.

3.3. Introducing three-class classification and robust edge constraints.

As described in subsection 2.3.4, for edges which require only one adaptively computed edge constraint, the constraint can be replaced by a manually constructed edge constraint. Consequently, if known a priori, it is not necessary to solve any eigenvalue problems on these edges. Therefore, we also propose an extended approach, which uses a three-class classification. In particular, the neural network distinguishes between edges or classes, respectively, where the eigenvalue problem is unnecessary (class 0), where the eigenvalue problem results in exactly one additional adaptive constraint (class 1), and where the eigenvalue problem selects more than one constraint (class 2). If an edge is assigned to class 0, we will not enforce any edge constraint. If an edge is assigned to class 1, we will enforce a single edge constraint as described in subsection 2.3.4. Otherwise, we solve the eigenvalue problem on the edge and enforce the computed adaptive constraints.

For the three-class classification, we use two different choices for the threshold

Table 2
Results on the complete training data set; the numbers are averages over all 4 500 training configurations. We define the accuracy (acc) as the number of true positives and true negatives divided by the total number of training configurations.

Classification type	Threshold	fp	fn	acc
two-class classification	0.45	8.8%	1.9%	89.2%
	0.5	5.4%	5.1%	89.5%
three-class classification	0.4	5.1%	1.0%	93.9%
	0.5	3.2%	2.3%	94.5%

$\tau = 0.4, 0.5$. Therefore, we first apply the standard multiclass classification rule

$$\arg\max_{c=0,1,2} \{P(p \in \omega_c)\};$$

cf. [65, p. 100]. Then, if p is assigned to any of {class 1, class 2}, we rescale the probabilities for the final classification in class 1 or 2 using the rule

$$\arg\max \left\{ \frac{P(p \in \omega_1)}{(1-\tau)(P(p \in \omega_1) + P(p \in \omega_2))}, \frac{P(p \in \omega_2)}{\tau(P(p \in \omega_1) + P(p \in \omega_2))} \right\}.$$

Consequently, a threshold of 0.5 results in an equal scaling, whereas a threshold of 0.4 results in more edges assigned to class 2. This can improve the robustness of our approach.

Despite that, we do not change our strategy, and using a grid search on the corresponding data set, we obtained the same hyperparameters as for the two-class classification model.

3.4. Results on the training data. On the complete set of training data, we obtain the results listed in Table 2. We observe a significantly better accuracy for the three-class classification compared to the two-class classification. Also, we observe that a classification threshold of 0.5 yields the best accuracy for both types of classification. However, for the cases of irregular subdomains in section 4, we will use a lower threshold τ to improve the robustness of the ML-FETI-DP approach. This will be discussed further in section 4.

4. Numerical results for ML-FETI-DP.

In this section, we compare FETI-DP, adaptive FETI-DP, and ML-FETI-DP. Therefore, we consider different coefficient functions ρ in model problem (2.1), domain decompositions with regular subdomains as well as irregular decompositions obtained with METIS [32], our two- and three-class model, and different ML (machine learning)-thresholds τ.

4.1. Coefficient functions. In the following subsections, we consider two types of discontinuous coefficient functions ρ with large jumps, i.e., a "circle problem" as depicted in Figure 9 (left) and a "microsection problem" as depicted in Figure 9 (middle). For the scalar diffusion case, we consider a coefficient $\rho = 1e6$ in the dark blue circles or, respectively, in the black part of the microsection. In the remaining parts of Ω, we have $\rho = 1$. For linear elasticity, we set the Poisson ratio constantly to $\nu = 0.3$ and only consider jumps in the Young's modulus E. In particular, we set $E = 1e3$ in the black part of the microsection and $E = 1$ elsewhere. The microsection depicted in Figure 9 (middle) is a subsection of the larger microsection from Figure 9 (right), suitable for our MATLAB computations. However, we consider a total of 10 different subsections of Figure 9 (right) that cover the whole structure to prove that

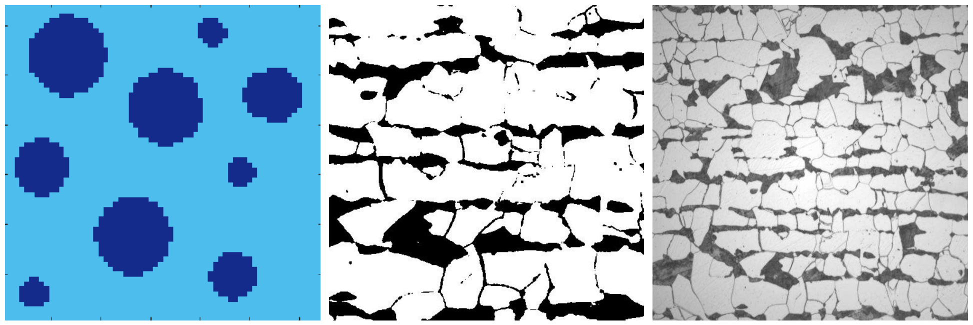

Fig. 9. Left: *Coefficient function ρ with randomly distributed circles of different sizes. We have $\rho = 1e6$ in the dark blue circles and $\rho = 1$ elsewhere. We refer to model problem* (2.1) *equipped with this coefficient distribution by the circle problem.* Middle: *Subsection of a microsection of a dual-phase steel obtained from the image on the right. We consider $\rho = 1e6$ in the black part and $\rho = 1$ elsewhere. We refer to model problem* (2.1) *equipped with this coefficient distribution by "microsection problem."* Right: *Complete microsection of a dual-phase steel. Let us remark that we consider different—randomly chosen—subsections of this microsection in the numerical results but discuss our approach in more details for a single example. Right image: Courtesy of Jörg Schröder, University of Duisburg-Essen, Germany, originating from a cooperation with ThyssenKruppSteel. Color is available online only.*

our algorithm is robust. For a more detailed discussions of the results of ML-FETI-DP, we choose the subsection depicted in Figure 9 (middle) as an example.

In our experiments, we always choose the mesh resolution such that the coefficient function is constant on each finite element. Then, the predictions and the accuracy of our classification algorithm are independent of the mesh resolution of the finite element mesh.

4.2. Two-class model.

Let us first discuss our two-class model. Here, the neural network distinguishes between critical edges, where the eigenvalue problem results in additional adaptive constraints, and edges where the eigenvalue problem is unnecessary. In the following, we will refer to the latter case as "negative" or "negative edge" and to the first one as "positive" or "positive edge." We always use a tolerance of $TOL = 100$ in the adaptive algorithm and, if not stated otherwise, an ML-threshold of 0.5.

4.2.1. Regular domain decompositions.

We consider a regular domain decomposition of the "circle problem" and the "microsection problem" into 64 subdomains. For both the "circle problem" and the "microsection problem," we use a discretization with 8 192 finite elements per subdomain. We depict both in Figure 10 (left) and Figure 11 and mark all edges using the following color code: edges which ML-FETI-DP correctly identifies as positive are marked in green (true positive), edges which ML-FETI-DP incorrectly identifies as positive are marked in yellow (false positive), and edges which ML-FETI-DP incorrectly identifies as negative are marked in red (false negative). All edges which are correctly identified as negative (true negative) are not marked. Let us remark that the yellow edges are not critical for the robustness and convergence of the algorithms and only a single unnecessary eigenvalue problem is solved for each yellow edge. In contrast, red edges might decrease the rate of convergence. Therefore, we also consider the approach of overshooting and lowering the ML-threshold in some cases. In Figure 10 (left), we see that only two yellow edges (false positives) but no critical red edges (false negatives) occur for

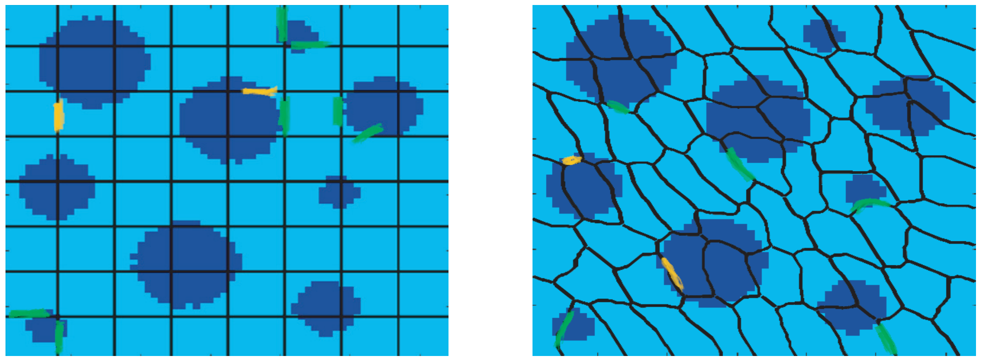

FIG. 10. *Circle problem with marked edges: true positives are marked in green, false positives are marked in yellow, and false negatives are marked in red.* Left: *Regular domain decomposition; cf. also Table* 3. Right: *METIS domain decomposition; cf. also Table* 6.

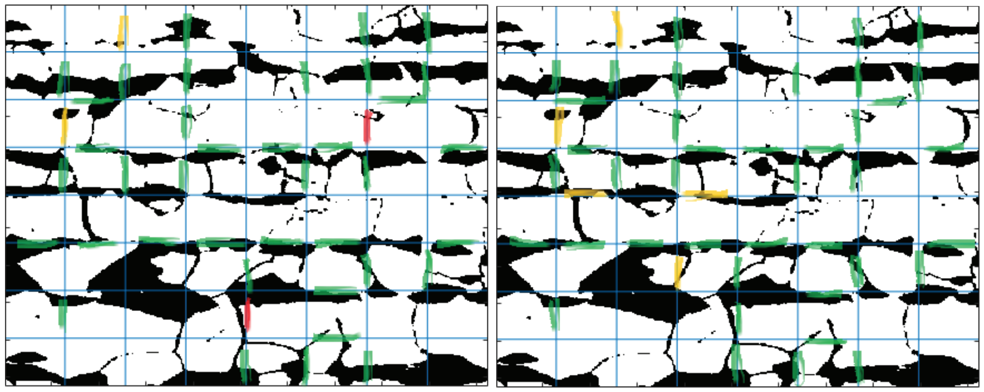

FIG. 11. *Microsection problem with marked edges: true positives are marked in green, false positives are marked in yellow, and false negatives are marked in red; cf. also Table* 3. Left: *ML-threshold of* 0.5*;* Right: *ML-threshold of* 0.45*.*

the “circle problem.” Here, we potentially save 92% of the eigenvalue problems. For the “microsection problem,” we save 65% of the eigenvalue problems using an ML-threshold of 0.5 and 60% using an ML-threshold of 0.45; cf. Table 3. In the latter case, we have no false negatives and two false negatives in the first case. See Figure 11 for graphical representations of both results. Additionally, we provide condition numbers and iteration counts in Table 3 for all discussed examples. The two false negative edges increase the condition number, but the convergence of ML-FETI-DP is still fast. To obtain also a low condition number, overshooting with the ML-threshold of 0.45 works as expected. To prove the robustness of ML-FETI-DP independent of the specific subsection of the microstructure depicted in Figure 9 (right), we summarize numerical results for 10 different subsections. We therefore present averages and maximum values of the condition number and iteration counts in Table 4.

We further provide numerical results for the linear elasticity problem in two dimensions for the same 10 different subsections of the “microsection problem” in Table 5. As for the diffusion case, we again use the same set of training and validation configurations to train the neural network. Here, we consider a domain decomposition into 64 subdomains and use a discretization of 1 800 finite elements per subdomain.

Table 3
Comparison of standard FETI-DP, adaptive FETI-DP, and ML-FETI-DP for regular domain decompositions *for the* two-class model*; cf. also Figure* 10 *(left) and Figure* 11*. We show the ML-threshold (τ), the condition number (cond), the number of CG iterations (it), the number of solved eigenvalue problems (evp), the number of false positives (fp), the number of false negatives (fn), and the accuracy in the classification (acc).*

Model problem	Algorithm	τ	cond	it	evp	fp	fn	acc
circle problem	standard	-	1.04e6	56	0	-	-	-
	adaptive	-	8.82	35	112	-	-	-
	ML	0.5	8.83	35	9	2	0	0.98
microsection problem	standard	-	-	>300	0	-	-	-
	adaptive	-	15.86	36	112	-	-	-
	ML	0.5	9.64e4	45	39	2	2	0.96
	ML	0.45	15.86	36	44	5	0	0.95

Table 4
Results for 10 *different subsections of a microsection of a dual-phase steel for the* two-class model*. Comparison of adaptive FETI-DP and ML-FETI-DP for* regular domain decompositions*. We show the average values as well as the maximum values (in brackets). See Table* 3 *for the column labeling.*

Ten different microsection problems							
Algorithm	τ	cond	it	evp	fp	fn	acc
adaptive	-	11.04	34.6	112.0	-	-	-
		(15.87)	(38)	(112)	-	-	-
ML	0.5	8.61e4	39.5	45.0	1.6	1.9	0.97
		(9.73e4)	(52)	(57)	(2)	(3)	(0.96)
ML	0.45	11.04	34.6	46.9	4.4	0	0.96
		(15.87)	(38)	(59)	(6)	(0)	(0.94)

Table 5
Results for 10 *different subsections of a microsection of a dual-phase steel for the* two-class model *for* linear elasticity*. Comparison of adaptive FETI-DP and ML-FETI-DP for* regular domain decompositions*. We show the average values as well as the maximum values (in brackets). See Table* 3 *for the column labeling.*

Ten different microsection problems							
Algorithm	τ	cond	it	evp	fp	fn	acc
adaptive	-	79.07	87.4	112.0	-	-	-
		(92.83)	(91)	(112)	-	-	-
ML	0.5	9.32e4	92.2	44.0	2.2	2.4	0.96
		(10.32e4)	(95)	(56)	(3)	(3)	(0.95)
ML	0.45	79.07	87.4	48.2	4.8	0	0.95
		(92.83)	(91)	(61)	(7)	(0)	(0.93)

We present averages and maximum values of the condition number and iteration counts in Table 5. Analogously to the diffusion case in Table 4, we are able to eliminate all false negative edges and thus obtain a robust algorithm when using the ML-threshold of 0.45.

4.2.2. METIS domain decompositions. Now, we consider a METIS domain decomposition of the "circle problem" and the "microsection problem" into 64 subdomains. We depict both in Figure 10 (right) and Figure 12 and mark the edges according to their classification using the same color code as before. Again, only red edges are critical for the robustness and we can make an effort to improve the robust-

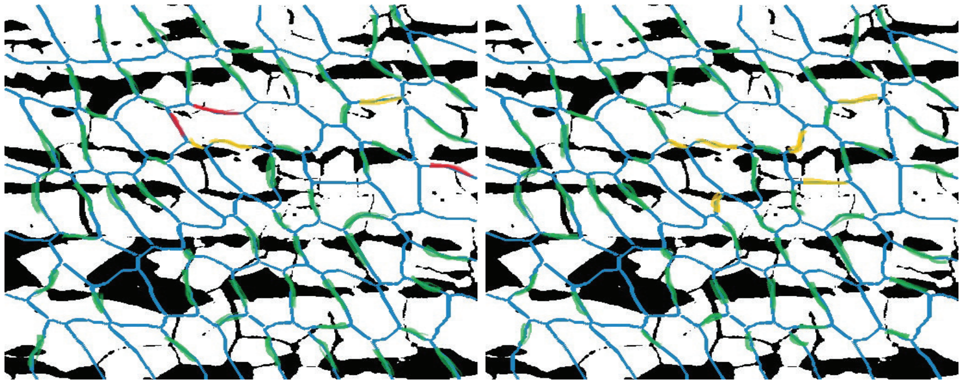

FIG. 12. *Microsection problem with marked edges: true positives are marked in green, false positives are marked in yellow, and false negatives are marked in red; cf. also Table* 6. Left: *ML-threshold of* 0.5*;* Right: *ML-threshold of* 0.45*.*

TABLE 6

Comparison of standard FETI-DP, adaptive FETI-DP, and ML-FETI-DP for a METIS domain decompositions *for the* two-class model*; cf. also Figure* 10 *(right) and Figure* 12*. See Table* 3 *for the column labeling.*

Model problem	Algorithm	τ	cond	it	evp	fp	fn	acc
circle problem	standard	-	9.18e5	75	0	-	-	-
	adaptive	-	13.56	37	160	-	-	-
	ML	0.5	13.56	37	7	2	0	0.99
microsection problem	standard	-	-	>350	-	-	-	-
	adaptive	-	16.52	35	160	-	-	-
	ML	0.5	1.78e4	51	62	3	3	0.96
	ML	0.45	16.52	35	68	6	0	0.96

ness using overshooting and therefore introducing some yellow marked false positive edges. Let us remark that, as described in subsection 3.1, we only use the training set with straight edges and edges with a single jag; see Figure 6. In Figure 10 (right), we observe that, as in the regular case, only two yellow edges (false positives) but no critical red edges (false negatives) occur for the "circle problem." Here, we potentially save 96% of the eigenvalue problems. For the "microsection problem," we save the computation of 61% of the eigenvalue problems considering an ML-threshold of 0.5 as well as 0.45. In the first case, ML-FETI-DP misses three critical edges (red edges), whereas in the second case, we are able to eliminate all red edges. See Figure 12 for both results. We also provide condition numbers and iteration counts in Table 6 for all discussed examples. Although ML-FETI-DP misses three critical edges for an ML-threshold of 0.5, the number of iterations remains moderate. Nonetheless, using a lower threshold of 0.45, we obtain the same robustness as adaptive FETI-DP. For METIS decompositions, we again summarize numerical results for 10 different subsections of the microsection depicted in Figure 9 (right). We present averages and maximum values of the condition number and iteration counts in Table 7.

For linear elasticity, we consider a METIS decomposition into 64 subdomains of the mesh used in subsection 4.2.1. As in the diffusion case, we obtain a robust algorithm with no false negative edges when using the ML-threshold of 0.4; see Table 8.

4.3. Three-class model. Let us now discuss numerical results for our three-class model; cf. subsection 3.3. As before, we always use a tolerance of $TOL = 100$

Table 7
Results for 10 *different subsections of a microsection of a dual-phase steel for the* two-class model. *Comparison of adaptive FETI-DP and ML-FETI-DP for* METIS domain decompositions. *We show the average values as well as the maximum values (in brackets). See Table* 3 *for the column labeling.*

Ten different microsection problems							
Algorithm	τ	cond	it	evp	fp	fn	acc
adaptive	-	14.81	35.6	160.0	-	-	-
		(22.58)	(38)	(160)	-	-	-
ML	0.5	1.38e4	51.0	63.2	2.0	2.0	0.97
		(2.07e4)	(52)	(73)	(3)	(3)	(0.96)
ML	0.45	14.81	35.6	65.4	6.2	0	0.96
		(22.58)	(38)	(75)	(7)	(0)	(0.95)

Table 8
Results for 10 *different subsections of a microsection of a dual-phase steel for the* two-class model *for* linear elasticity. *Comparison of adaptive FETI-DP and ML-FETI-DP for* METIS domain decompositions. *We show the average values as well as the maximum values (in brackets). See Table* 3 *for the column labeling.*

Ten different microsection problems							
Algorithm	τ	cond	it	evp	fp	fn	acc
adaptive	-	85.73	89.4	160.0	-	-	-
		(99.05)	(97)	(160)	-	-	-
ML	0.5	2.74e4	92.8	65.0	2.4	2.6	0.96
		(2.89e4)	(102)	(74)	(3)	(3)	(0.96)
ML	0.45	85.73	89.4	67.2	7.4	0	0.96
		(99.05)	(97)	(77)	(8)	(0)	(0.95)

in the adaptive algorithm and, if not stated otherwise, an ML-threshold of 0.5. Let us remark that we only consider the "microsection problem" in this section, since for the "circle problem" only edges from classes 0 and 1 occur.

4.3.1. Regular domain decompositions. We consider the same discretization and domain decomposition as in subsection 4.2.1. Besides FETI-DP with primal vertices, adaptive FETI-DP, and ML-FETI-DP, we additionally compare FETI-DP with primal vertex constraints and weighted edge averages as described in subsection 2.3.4 on all edges. The latter approach, which we denote as FETI-DP(e), also does not require the solution of any eigenvalue problems.

As depicted in Figure 13, we obtain a single red edge, where ML-FETI-DP assigns this edge falsely to class 1 instead of class 2. Nonetheless, ML-FETI-DP is still robust, since this single edge is not critical for convergence; see Table 9. ML-FETI-DP using the three-class model saves 93% of the eigenvalue problems compared to adaptive FETI-DP. This is a significant improvement over ML-FETI-DP using the two-class model investigated in subsection 4.2, where we saved 65% for exactly the same model problem. Again, considering a lower ML-threshold of $\tau = 0.4$ as described in section 3.3, we can get rid of all false negative edges for the prize of additional false positives; see Figure 13. As already mentioned, the latter ones do not affect the robustness of ML-FETI-DP. Also for the three-class model, we test for the 10 subsections of the complete microsection considered in subsection 4.2.2. We present averages and maximum values of the condition number and iteration counts in Table 10.

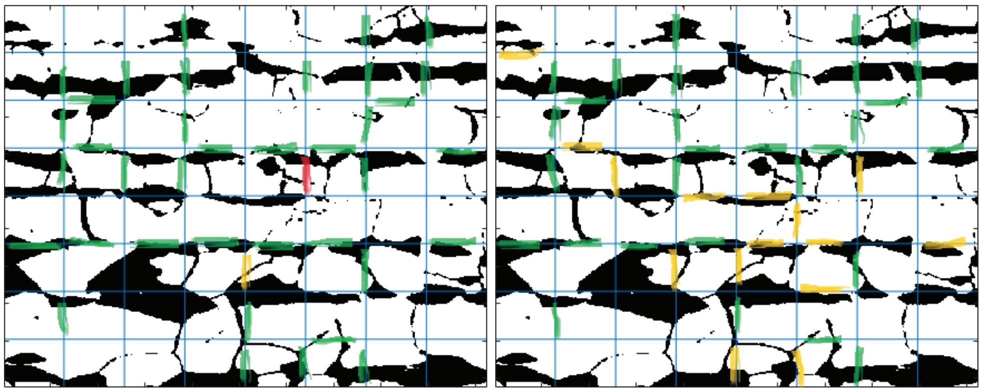

FIG. 13. *Microsection problem with marked edges. Correctly assigned edges from classes* 1 *and* 2 *are marked in green. Edges from class* 1 *which are falsely assigned to class* 0*, and edges from class* 2 *which are falsely assigned to class* 0*, are marked in red. Edges from class* 0*, which are falsely assigned to class* 1*, and edges from class* 1*, which are falsely assigned to class* 2*, are marked in yellow; cf. also Table* 9*. Therefore, again, only red edges are critical for the robustness of ML-FETI-DP.* Left: *ML-threshold of* 0.5*;* Right: *ML-threshold of* 0.4*.*

TABLE 9
Comparison of standard FETI-DP, standard FETI-DP(e), adaptive FETI-DP, and ML-FETI-DP for regular domain decompositions *for the* three-class model*; cf. also Figure* 13*. See Table* 3 *for the column labeling.*

Model problem	Algorithm	τ	cond	it	evp	e-avg	fp	fn	acc
	standard	-	-	>300	0	-	-	-	-
microsection	standard(e)	-	8.21e4	127	0	112	-	-	-
problem	adaptive	-	15.86	36	112	-	-	-	-
	ML	0.5	231.37	56	8	30	1	1	0.98
	ML	0.4	16.21	37	24	18	16	0	0.85

TABLE 10
Results for 10 *different subsections of a microsection of a dual-phase steel for the* three-class model*. Comparison of adaptive FETI-DP and ML-FETI-DP for* regular domain decompositions*. We show the average values as well as the maximum values (in brackets). See Table* 3 *for the column labeling.*

		Ten different microsection problems						
Algorithm	τ	cond	it	evp	e-avg	fp	fn	acc
adaptive	-	11.04	34.6	112.0	-	-	-	-
		(15.87)	(38)	(112)	-	-	-	-
ML	0.5	147.41	48.8	4.1	43.6	1.7	1.3	0.97
		(271.38)	(58)	(10)	(46)	(3)	(3)	(0.95)
ML	0.4	12.37	34.8	16.0	24.2	10.5	0.0	0.90
		(16.41)	(39)	(24)	(28)	(16)	(0)	(0.85)

4.3.2. METIS domain decompositions. Considering the same problem as in subsection 4.3.1, but using irregular domain decompositions obtained by METIS, does not change the picture. We obtain again only one single critical edge using an ML-threshold of $\tau = 0.5$. As for the regular domain decomposition, both can be removed by choosing $\tau = 0.4$; see Figure 14. In both cases, ML-FETI-DP is robust and converges fast, and up to 94% of the eigenvalue problems can be saved; see Table 11. The behavior does not change for the 10 different microsections, and

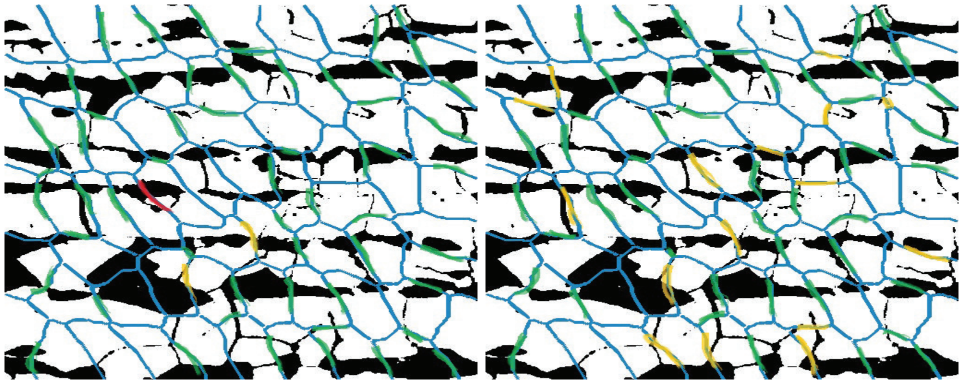

FIG. 14. *Microsection problem with marked edges. Correctly assigned edges from classes* 1 *and* 2 *are marked in green. Edges from class* 1 *which are falsely assigned to class* 0*, and edges from class* 2 *which are falsely assigned to class* 0*, are marked in red. Edges from class* 0 *which are falsely assigned to class* 1*, and edges from class* 1 *which are falsely assigned to class* 2*, are marked in yellow; cf. also Table* 11*. Therefore, again, only red edges are critical for the robustness of ML-FETI-DP.* Left: *ML-threshold of* 0.5*;* Right: *ML-threshold of* 0.4*.*

TABLE 11
Comparison of standard FETI-DP, standard FETI-DP(e), adaptive FETI-DP, and ML-FETI-DP for METIS domain decompositions *for the* three-class *model; cf. also Figure* 14*. See Table* 3 *for the column labeling.*

Model problem	Algorithm	τ	cond	it	evp	e-avg	fp	fn	acc
	standard	-	-	>350	0	-	-	-	-
microsection	standard(e)	-	-	>350	0	160	-	-	-
problem	adaptive	-	16.52	35	160	-	-	-	-
	ML	0.5	245.71	56	9	42	2	1	0.98
	ML	0.4	17.82	36	26	31	16	0	0.90

TABLE 12
Results for 10 *different subsections of a microsection of a dual-phase steel for the* three-class model*. Comparison of adaptive FETI-DP and ML-FETI-DP for* METIS domain decompositions*. We show the average values as well as the maximum values (in brackets). See Table* 3 *for the column labeling.*

Ten different microsection problems								
Algorithm	τ	cond	it	evp	e-avg	fp	fn	acc
adaptive	-	14.81	35.6	160.0	-	-	-	-
		(22.58)	(38)	(160)	-	-	-	-
ML	0.5	233.55	53.2	7.5	46.2	1.6	2.0	0.97
		(256.51)	(57)	(10)	49	(2)	(3)	(0.96)
ML	0.4	15.73	36.8	23.8	28.8	15.8	0.0	0.89
		(24.04)	(40)	(26)	(31)	(21)	(0)	(0.86)

we again provide average and maximum values in Table 12.

5. Conclusion and future work.

We introduced two different machine learning based classification strategies to predict the critical edges where adaptive constraints have to be enforced in adaptive FETI-DP. Both approaches helped to reduce the number of necessary eigenvalue problems significantly. We showed numerically the stability and robustness of the new method ML-FETI-DP and saved up to 94% of the eigenvalue problems for realistic coefficient functions obtained from a microsection of dual phase steel. Although we concentrated on adaptive FETI-DP, this work can be generalized to different domain decomposition methods, such as, e.g., BDDC and GDSW. We also plan to investigate the case of three spatial dimensions. Another approach, which we plan to develop, is the prediction of the adaptive constraints themselves using machine learning techniques.

REFERENCES

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*, https://www.tensorflow.org/, 2015; software available from https://tensorflow.org/.

[2] 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.

[3] S. Badia, A. F. Martín, and J. Principe, *Multilevel balancing domain decomposition at extreme scales*, SIAM J. Sci. Comput., 38 (2016), pp. C22–C52, https://doi.org/10.1137/15M1013511.

[4] 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, https://doi.org/10.1137/15M1054675.

[5] 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 Comput. Sci. 1947, Springer-Verlag, Berlin, Heidelberg, 2000, pp. 373–383.

[6] D. Braess, *Finite Elements: Theory, Fast Solvers, and Applications in Solid Mechanics*, 2nd ed., Cambridge University Press, Cambridge, UK, 2001.

[7] S. Brenner and L. R. Scott, *The Mathematical Theory of Finite Element Methods*, 2nd ed., Springer-Verlag, Berlin, 2002.

[8] 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.

[9] C. R. Dohrmann, *A preconditioner for substructuring based on constrained energy minimization*, SIAM J. Sci. Comput., 25 (2003), pp. 246–258, https://doi.org/10.1137/S1064827502412887.

[10] C. R. Dohrmann, A. Klawonn, and O. B. Widlund, *Domain decomposition for less regular subdomains: Overlapping Schwarz in two dimensions*, SIAM J. Numer. Anal., 46 (2008), pp. 2153–2168, https://doi.org/10.1137/070685841.

[11] C. R. Dohrmann, A. Klawonn, and O. B. Widlund, *A family of energy minimizing coarse spaces for overlapping Schwarz preconditioners*, in Domain Decomposition Methods in Science and Engineering XVII, Lect. Notes Comput. Sci. Eng. 60, Springer, Berlin, 2008, pp. 247–254.

[12] C. R. Dohrmann and O. B. Widlund, *An overlapping Schwarz algorithm for almost incompressible elasticity*, SIAM J. Numer. Anal., 47 (2009), pp. 2897–2923, https://doi.org/10.1137/080724320.

[13] 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.

[14] J. Duchi, E. Hazan, and Y. Singer, *Adaptive subgradient methods for online learning and stochastic optimization*, J. Mach. Learn. Res., 12 (2011), pp. 2121–2159.

[15] 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.

[16] 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.

[17] C. Farhat, M. Lesoinne, and K. Pierson, *A scalable dual-primal domain decomposition*

method, Numer. Linear Algebra Appl., 7 (2000), pp. 687–714.

[18] J. Galvis and Y. Efendiev, *Domain decomposition preconditioners for multiscale flows in high-contrast media*, Multiscale Model. Simul., 8 (2010), pp. 1461–1483, https://doi.org/10.1137/090751190.

[19] J. Galvis and Y. Efendiev, *Domain decomposition preconditioners for multiscale flows in high contrast media: Reduced dimension coarse spaces*, Multiscale Model. Simul., 8 (2010), pp. 1621–1644, https://doi.org/10.1137/100790112.

[20] M. J. Gander, A. Loneland, and T. Rahman, *Analysis of a New Harmonically Enriched Multiscale Coarse Space for Domain Decomposition Methods*, preprint, https://arxiv.org/abs/1512.05285, 2015.

[21] X. Glorot, A. Bordes, and Y. Bengio, *Deep sparse rectifier neural networks*, in Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, 2011, pp. 315–323.

[22] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep Learning*, Vol. 1, MIT Press, Cambridge, MA, 2016.

[23] A. Heinlein, *Parallel Overlapping Schwarz Preconditioners and Multiscale Discretizations with Applications to Fluid-Structure Interaction and Highly Heterogeneous Problems*, PhD thesis, Universität zu Köln, Cologne, Germany, 2016.

[24] 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, Springer, Cham, pp. 373–382.

[25] A. Heinlein, A. Klawonn, J. Knepper, and O. Rheinbach, *Adaptive GDSW coarse spaces for overlapping Schwarz methods in three dimensions*, SIAM J. Sci. Comput., 41 (2019), pp. A3045–A3072, https://doi.org/10.1137/18M1220613.

[26] A. Heinlein, A. Klawonn, J. Knepper, and O. Rheinbach, *Multiscale coarse spaces for overlapping Schwarz methods based on the ACMS space in* 2*D*, Electron. Trans. Numer. Anal., 48 (2018), pp. 156–182.

[27] A. Heinlein, A. Klawonn, M. Lanser, and J. Weber, *A Generalized Robust FETI-DP Coarse Space for Heterogeneous Problems and Arbitrary Scalings*, manuscript.

[28] 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 J. Sci. Comput., 38 (2016), pp. C713–C747, https://doi.org/10.1137/16M1062843.

[29] A. Heinlein, A. Klawonn, O. Rheinbach, and O. 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, Springer, Cham, 2018, pp. 383–392.

[30] M. Jarošová, A. Klawonn, and O. Rheinbach, *Projector preconditioning and transformation of basis in FETI-DP algorithms for contact problems*, Math. Comput. Simulation, 82 (2012), pp. 1894–1907.

[31] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun, *What is the best multi-stage architecture for object recognition?*, in Proceedings of the 12th International IEEE Conference on Computer Vision, 2009, pp. 2146–2153.

[32] G. Karypis and V. Kumar, *MeTis: Unstructured Graph Partitioning and Sparse Matrix Ordering System, Version* 4.0.

[33] 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, https://doi.org/10.1137/140970598.

[34] D. P. Kingma and J. Ba, *Adam: A Method for Stochastic Optimization*, preprint, https://arxiv.org/abs/1412.6980, 2014.

[35] 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, https://doi.org/10.1137/15M1049610.

[36] 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, Lect. Notes Comput. Sci. Eng. 116, Springer, Cham, 2017, pp. 187–196.

[37] 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.

[38] 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, https://doi.org/10.1137/140997907.

[39] 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, Lect. Notes Comput. Sci. Eng. 112, Springer, Cham, 2016, pp. 255–264.

[40] A. Klawonn, M. Lanser, and O. Rheinbach, *Nonlinear BDDC methods with approximate solvers*, Electron. Trans. Numer. Anal., 49 (2018), pp. 244–273, https://doi.org/10.1553/etna_vol49s244.

[41] 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, https://doi.org/10.1137/16M1102495.

[42] A. Klawonn, P. Radtke, and O. Rheinbach, *FETI-DP methods with an adaptive coarse space*, SIAM J. Numer. Anal., 53 (2015), pp. 297–320, https://doi.org/10.1137/130939675.

[43] 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.

[44] 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.

[45] 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.

[46] 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, https://doi.org/10.1137/100811118.

[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, https://doi.org/10.1137/070688675.

[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, https://doi.org/10.1137/S0036142901388081.

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

[51] 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.

[52] 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.

[53] 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.

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

[55] J. Mandel and R. Tezaur, *On the convergence of a dual-primal substructuring method*, Numer. Math., 88 (2001), pp. 543–558.

[56] A. Müller and S. Guido, *Introduction to Machine Learning with Python: A Guide for Data Scientists*, O'Reilly Media, Sebastopol, CA, 2016.

[57] V. Nair and G. E. Hinton, *Rectified linear units improve restricted Boltzmann machines*, in Proceedings of the 27th International Conference on Machine Learning (ICML-10), 2010, pp. 807–814.

[58] 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.

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

[60] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, *Scikit-learn: Machine learning in Python*, J. Mach. Learn. Res., 12 (2011), pp. 2825–2830.

[61] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning*, Cambridge University Press, Cambridge, UK, 2014.

[62] 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.

[63] 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.

[64] A. Toselli and O. Widlund, *Domain decomposition methods—algorithms and theory*, Springer Ser. Comput. Math. 34, Springer-Verlag, Berlin, 2005.

[65] J. Watt, R. Borhani, and A. K. Katsaggelos, *Machine Learning Refined: Foundations, Algorithms, and Applications*, Cambridge University Press, Cambridge, UK, 2016.

[66] S. Zampini, *PCBDDC: A class of robust dual-primal methods in PETSc*, SIAM J. Sci. Comput., 38 (2016), pp. S282–S306, https://doi.org/10.1137/15M1025785.