International Journal for Numerical Methods in Fluids

WILEY

RESEARCH ARTICLE OPEN ACCESS

Coupled Clustering in Hierarchical Matrices for the Oseen Problem

Jonas Grams | Sabine Le Borne

Hamburg University of Technology, Hamburg, Germany

Correspondence: Sabine Le Borne (leborne@tuhh.de)

Received: 18 September 2025 | **Revised:** 28 November 2025 | **Accepted:** 15 January 2026

Keywords: clustering | hierarchical matrix | Oseen problem | preconditioning

ABSTRACT

Fluid flow problems can be modelled by the Navier-Stokes or, after linearization, by the Oseen equations. Their discretization results in discrete saddle point problems. These systems of equations are typically very large and need to be solved iteratively. Standard (block-) preconditioning techniques for saddle point problems rely on an approximation of the Schur complement. Such an approximation can be obtained by a hierarchical ($\mathcal{H}$-) matrix LU-decomposition which first approximates the Schur complement explicitly. The computational complexity of this computation depends, among other things, on the hierarchical block structure of the involved hierarchical matrices. However, widely used techniques do not consider the connection between the discretization grids for the velocity field and the pressure, respectively. Here we present a hierarchical block structure for the finite element discretization of the gradient operator which is improved by considering the connection between the two involved grids. We prove a logarithmic depth estimate for cluster trees generated with the coupled clustering. Numerical results will show that the improved block structure allows for a faster computation of the Schur complement which is the bottleneck for the set-up of the $\mathcal{H}$-matrix LU-decomposition. The presented coupled clustering is also applicable to other types of mixed (finite element) problems.

MSC2020 Classification: MSC 65F08, MSC 65N22

1 | Introduction

The simulation of (incompressible) fluids or gases is often based on the Navier-Stokes equations, a non-linear system of partial differential equations. Solving this system requires first a linearization of the problem and second a discretization of the linearized problem. The linearization with, for example, the Picard or the Newton iteration, results in a so-called Oseen problem while the discretization, for example, with the finite element method (FEM), results in a linear system with a natural 2×2 block structure (see, e.g., [1]), a so-called saddle point system of the form

$$\mathcal{M}\begin{pmatrix}\mathbf{x}\\ \mathbf{y}\end{pmatrix} = \begin{pmatrix}\mathbf{F} & \mathbf{B}^{\mathsf{T}}\\ \mathbf{B} & \mathbf{0}\end{pmatrix}\begin{pmatrix}\mathbf{x}\\ \mathbf{y}\end{pmatrix} = \begin{pmatrix}\mathbf{r}\\ \mathbf{s}\end{pmatrix}.$$

These systems are typically large, sparse, non-symmetric, and indefinite. Hence, Krylov subspace methods such as the BiCGStab method or the GMRes method are often used to solve them. Fast convergence of these methods can often only be achieved with preconditioning. Here, we pursue the popular block triangular preconditioning technique [2] which requires easily invertible approximations to the saddle point matrix block $\mathbf{F}$ and to the Schur complement $\mathbf{S} := -\mathbf{B}\mathbf{F}^{-1}\mathbf{B}^{\mathsf{T}}$. We will consider hierarchical matrix ($\mathcal{H}$-matrix) LU factorizations for these approximations (cf. [3, 4]).

$\mathcal{H}$-matrices have been introduced in 1999 as data-sparse matrix format for approximations to (typically dense) integral operator discretizations. They offer an almost linear memory and computational complexity (see, e.g., [5, 6]). Since then, they have also

been considered for other applications such as approximations to the inverse or LU factorization of FEM matrices (cf. [7–9]).

Some key properties of the complexity estimates highly depend on the block structure of the involved $\mathcal{H}$-matrices. Their block structure typically depends on tree structures, so-called cluster and block cluster trees.

A (problem dependent) admissibility condition determines whether matrix blocks can be represented in a low rank format (and hence data sparse) [4, 6, 10]. For our model problem, we require two different cluster trees, one for the index set $\mathcal{I}$ obtained from the velocity discretization and one for the index set $\mathcal{J}$ from the pressure discretization. In [11], we proposed a novel coupled clustering strategy generating the cluster tree $\mathcal{T}_{\mathcal{I}}$ for the index set $\mathcal{I}$ based on a previously generated cluster tree $\mathcal{T}_{\mathcal{J}}$ for the index set $\mathcal{J}$. This approach improved the block structure of the off-diagonal saddle point matrix block $\mathbf{B}$ (or rather sub-blocks of $\mathbf{B}$) and resulted in a significant speed-up for the preconditioner set-up compared to previously used uncoupled clustering strategies as, for example, used in [4]. We extend the results from [11] by a proof of a logarithmic depth estimate for cluster trees generated with the coupled clustering and present additional numerical results. Furthermore, we present an adaptation of the coupled clustering which aims to tackle some disadvantages of the coupled clustering by introducing an additional decomposition of so-called interface clusters.

The remainder of this paper is structured as follows. First, in Section 2, we describe our Oseen model problem yielding a saddle point problem through the finite element discretization. This is followed in Section 3 by a short discussion of the block preconditioning of saddle point matrices with $\mathcal{H}$-LU factorizations. In Section 4, we will give a brief description of hierarchical matrices and the tree structures used in their construction. Then, in Section 5, we give a brief description of the uncoupled clustering [4] and the novel coupled clustering [11] and present the proof for the logarithmic depth estimate. Additionally, we introduce the coupled clustering with interface decomposition. Finally, we present numerical results in Section 6.

2 | Model Problem

We consider a d-dimensional Oseen problem with Dirichlet boundary condition: Find functions $\mathbf{u} : \Omega \to \mathbb{R}^d$ and $p : \Omega \to \mathbb{R}$ with

$$\begin{aligned} -\nu\Delta\mathbf{u} + \mathbf{w}\cdot\nabla\mathbf{u} + \nabla p &= \mathbf{f} \quad \text{on } \Omega := (-1,1)^d, \\ \nabla\cdot\mathbf{u} &= \mathbf{0} \quad \text{on } \Omega, \\ \mathbf{u} &= \mathbf{g} \quad \text{on } \partial\Omega =: \Gamma. \end{aligned}$$

for a right hand side $\mathbf{f} : \mathbb{R}^d \to \mathbb{R}^d$, boundary values $\mathbf{g} : \Gamma \to \mathbb{R}^d$, a convection $\mathbf{w} : \mathbb{R}^d \to \mathbb{R}^d$ and viscosity parameter $\nu > 0$.

A finite element discretization of this problem with modified Taylor-Hood elements and piecewise-linear basis functions is based on two triangulations, that is, sets of simplices covering Ω: A coarse triangulation $\mathcal{T}^h$ with $m+1$ ($m \in \mathbb{N}$) vertices (cf. [12, remark 4.70]) in the interior of Ω and on the boundary

$$\chi_1, \ldots, \chi_{m+1} \in \mathbb{R}^d$$

with mesh width $h = \min\{\|\chi_i - \chi_j\| : i, j \in \{1, \ldots, m+1\}, i \neq j\}$ for the pressure discretization, and a refined triangulation $\mathcal{T}^{h/2}$ for the velocity field discretization. In case of tetrahedra, the refined triangulation may be obtained by splitting each tetrahedron into eight tetrahedra whose vertices either coincide with vertices of the original tetrahedron or its edge midpoints. We denote the $n \in \mathbb{N}$ inner vertices of the refined triangulation $\mathcal{T}^{h/2}$ by

$$\xi_1, \ldots, \xi_n \in \mathbb{R}^d$$

and its $n_D \in \mathbb{N}$ boundary vertices by

$$\xi_{n+1}, \ldots, \xi_{n+n_D} \in \Gamma \subseteq \mathbb{R}^d.$$

The refined triangulation has mesh width $\min\{\|\xi_i - \xi_j\|, i, j \in \{1, \ldots, n+n_D\}, i \neq j\} = h/2$. We denote the set of vertices of $\mathcal{T}^h$ and $\mathcal{T}^{h/2}$ by

$$\mathcal{V}(\mathcal{T}^h) := \{\chi_1, \ldots, \chi_{m+1}\} \quad \text{and} \quad \mathcal{V}(\mathcal{T}^{h/2}) := \{\xi_1, \ldots, \xi_{n+n_D}\},$$

respectively. These triangulations yield the two scalar-valued finite-dimensional spaces of continuous, piecewise linear polynomials

$$Q^h := \mathrm{P}^1(\mathcal{T}^h) \text{ and } V^h := \mathrm{P}^1(\mathcal{T}^{h/2})$$

with respective bases $(\Psi_1, \ldots, \Psi_m)$ and $(\varphi_1, \ldots, \varphi_n, \varphi_{n+1}, \ldots, \varphi_{n+n_D})$ where

$$\operatorname{span}(\varphi_1, \ldots, \varphi_n) = V_0^h := V^h \cap \mathrm{H}_0^1(\Omega;\Gamma).$$

For the discretization of the velocity, we use $(V^h)^d$. The discretization (and elimination of Dirichlet boundary nodes) leads to a saddle point problem of the form

$$\mathcal{M}\begin{pmatrix}\mathbf{x}\\ \mathbf{y}\end{pmatrix} = \begin{pmatrix}\mathbf{F} & \mathbf{B}^{\mathsf{T}}\\ \mathbf{B} & \mathbf{0}\end{pmatrix}\begin{pmatrix}\mathbf{x}\\ \mathbf{y}\end{pmatrix} = \begin{pmatrix}\check{\mathbf{F}} & & & (\mathbf{B}^1)^{\mathsf{T}}\\ & \ddots & & \vdots\\ & & \check{\mathbf{F}} & (\mathbf{B}^d)^{\mathsf{T}}\\ \mathbf{B}^1 & \cdots & \mathbf{B}^d & \mathbf{0}\end{pmatrix}\begin{pmatrix}\mathbf{x}_1\\ \vdots\\ \mathbf{x}_d\\ \mathbf{y}\end{pmatrix} = \begin{pmatrix}\mathbf{r}_1\\ \vdots\\ \mathbf{r}_d\\ \mathbf{s}\end{pmatrix} = \begin{pmatrix}\mathbf{r}\\ \mathbf{s}\end{pmatrix}, \tag{1}$$

with $\check{\mathbf{F}} \in \mathbb{R}^{\mathcal{I}\times\mathcal{I}}$ and $\mathbf{B}_1, \mathbf{B}_2, \mathbf{B}_3 \in \mathbb{R}^{\mathcal{J}\times\mathcal{I}}$ where $\mathcal{I} = \{1, \ldots, n\}$ and $\mathcal{J} = \{1, \ldots, m\}$ denote the velocity and pressure index sets, respectively (see, e.g., [1]).

Remark 1. There exists a wide range of (stable) discretizations for the Oseen problem. A popular alternative to the discretization used by us (piecewise linear elements on two different grids) is given by Taylor-Hood elements which use two different polynomial spaces on the same grid, for example, $Q^h := \mathrm{P}^k(\mathcal{T}^h)$ and $V^h := \mathrm{P}^{k+1}(\mathcal{T}^h)$ for some $k \in \mathbb{N}$. Our subsequent considerations generalize to this as well as other classes of mixed discretizations. In short, our work deals with reorderings of index sets for velocity and pressure unknowns resulting from mixed finite element discretizations. The main idea will be to determine reorderings in a coupled way and not independently of each other based on resulting sparsity patterns in the off-diagonal block $\mathbf{B}$ in (1).

3 | ($\mathcal{H}$-LU) Block Preconditioning of Saddle Point Problems

A popular approach for preconditioning of saddle point problems as in (1) is the so-called block triangular preconditioner (see, e.g., in [1]) which is derived from the block factorization

$$\begin{pmatrix} \mathbf{F} & \mathbf{B}^\mathsf{T} \\ \mathbf{B} & \mathbf{0} \end{pmatrix} = \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{B}\mathbf{F}^{-1} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{F} & \mathbf{B}^\mathsf{T} \\ \mathbf{0} & \mathbf{S} \end{pmatrix}$$

with the Schur complement $\mathbf{S} = -\mathbf{B}\mathbf{F}^{-1}\mathbf{B}^\mathsf{T} \in \mathbb{R}^{\mathcal{J}\times\mathcal{J}}$. The preconditioner uses only the upper block triangular factor and replaces its diagonal blocks by approximations, that is,

$$\mathcal{P} := \begin{pmatrix} \widetilde{\mathbf{F}} & \mathbf{B}^\mathsf{T} \\ \mathbf{0} & \widetilde{\mathbf{S}} \end{pmatrix} \quad \text{with } \widetilde{\mathbf{F}} \approx \mathbf{F},\ \widetilde{\mathbf{S}} \approx \mathbf{S}. \tag{2}$$

We will have $\widetilde{\mathbf{F}}$ and $\widetilde{\mathbf{S}}$ in the form of LU factorizations such that $\mathcal{P}^{-1}$ can be easily applied to vectors. Since our work focuses on the computation of the approximations $\widetilde{\mathbf{F}}$ and $\widetilde{\mathbf{S}}$, we will outline the involved computational steps explicitly. In the following, we denote approximate (formatted) matrix addition by $\oplus$ and approximate (formatted) matrix multiplication by $\odot$. These approximations result from the use of hierarchical matrix arithmetic which leads to almost optimal complexity for these operations.

① Compute an approximate LUfactorization $\check{\mathbf{L}}_\mathbf{F}\check{\mathbf{U}}_\mathbf{F} \approx \check{\mathbf{F}}$ which yields

$$\widetilde{\mathbf{F}} := \begin{pmatrix} \check{\mathbf{L}}_\mathbf{F}\check{\mathbf{U}}_\mathbf{F} & & \\ & \ddots & \\ & & \check{\mathbf{L}}_\mathbf{F}\check{\mathbf{U}}_\mathbf{F} \end{pmatrix}.$$

② Compute $\mathbf{V}^k := \mathbf{B}^k\check{\mathbf{U}}_\mathbf{F}^{-1}$ for $k = 1, \dots, d$ (with backward substitution in approximate arithmetic).

③ Compute $\mathbf{W}^k := \check{\mathbf{L}}_\mathbf{F}^{-1}(\mathbf{B}^k)^\mathsf{T}$ for $k = 1, \dots, d$ (with forward substitution in approximate arithmetic).

④ Compute a Schur complement approximation $\check{\mathbf{S}} := \bigoplus_{k=1}^{d} \mathbf{V}^k \odot \mathbf{W}^k$.

⑤ Compute an approximate LU factorization $\widetilde{\mathbf{S}} := \mathbf{L}_\mathbf{S}\mathbf{U}_\mathbf{S} \approx \check{\mathbf{S}}$.

The (prefactors in the) computational complexity of these five steps depends on the complexity of the computations of approximations for matrix sums and products. We will use hierarchical matrix arithmetic which is based on a hierarchical block structure of the matrix. In the following, we will briefly review hierarchical matrices and then introduce a novel, coupled block structure that accelerates the computation of steps ② to ④.

4 | Hierarchical Matrices

Let $\mathcal{I}$ and $\mathcal{J}$ be two arbitrary (non-empty) index sets. Then the block structure of a hierarchical matrix $\mathbf{H} \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$ is based on (hierarchies of) partitions of $\mathcal{I}$ and $\mathcal{J}$ which may be obtained by clustering the index sets via a tree structure as described, for example, in [13].

Definition 1. Let $\mathcal{T} = (V, E)$ be a tree with root $r := \operatorname{root}(\mathcal{T})$.

1. $\mathcal{T}$ is called a labeled tree if there is a set L of labels and a mapping $\hat{\cdot}\colon V \to L$ assigning a label to each node.
2. For $v, w \in V$ with $(v, w) \in E$, we call v *predecessor* of w and w *successor* of v. Furthermore, we denote the set of successors of v by
$$\mathrm{S}(v) := \{w \in V : (v, w) \in E\}.$$
3. We call vertices $v \in V$ without successors, that is, $\mathrm{S}(v) = \emptyset$, *leaves* of $\mathcal{T}$. The set of leaves of $\mathcal{T}$ is denoted by
$$\mathcal{L}(\mathcal{T}) := \{v \in V : \mathrm{S}(v) = \emptyset\}.$$
4. For vertices $w \in V$, we define the *level* recursively through
$$\operatorname{level}(r) := 0,$$
$$\operatorname{level}(w) := \operatorname{level}(v) + 1 \qquad \text{with } (v, w) \in E.$$
5. We define the depth of $\mathcal{T}$ by
$$\operatorname{depth}(\mathcal{T}) := \max_{v\in V} \operatorname{level}(v).$$

In the following, we will identify $\mathcal{T}$ with the set of vertices V, that is, we write $v \in \mathcal{T}$ instead of $v \in V$.

Definition 2. Let $\mathcal{I}$ be an index set and $\mathcal{T} = (V, E)$ a labeled tree with labels $L := \mathcal{P}(\mathcal{I}) \setminus \emptyset$ where $\mathcal{P}(\mathcal{I})$ is the power set of $\mathcal{I}$. Then $\mathcal{T}$ is called a cluster tree for the index set $\mathcal{I}$ if

1. the label of the root $r := \operatorname{root}(\mathcal{T})$ of $\mathcal{T}$ is given by $\hat{r} = \mathcal{I}$,
2. for each node $t \in V$ with successors $\mathrm{S}(t)$, there holds
$$\hat{t} = \bigcup_{t'\in\mathrm{S}(t)} \hat{t}',$$
where the labels of all successors are disjoint.

We call vertices $t \in \mathcal{T}$ *clusters* and denote cluster trees for an index set $\mathcal{I}$ as $\mathcal{T}_\mathcal{I}$ and the set of leaves by $\mathcal{L}_\mathcal{I} := \mathcal{L}(\mathcal{T}_\mathcal{I})$. Furthermore, we will use clusters $t \in \mathcal{T}_\mathcal{I}$ as identifiers for their labels $\hat{t}$, for example, in $x|_t := x_{\hat{t}}$ for $x \in \mathbb{R}^\mathcal{I}$.

The set of leaves $\mathcal{L}_\mathcal{I}$ yields a partition of the index set $\mathcal{I}$. Two cluster trees $\mathcal{T}_\mathcal{I}, \mathcal{T}_\mathcal{J}$ for index sets $\mathcal{I}$ and $\mathcal{J}$, respectively, can be used to construct a block cluster tree (Definition 3) whose leaves in turn provide a partition of $\mathcal{I} \times \mathcal{J}$.

Definition 3. Let $\mathcal{I}, \mathcal{J}$ be index sets with cluster trees $\mathcal{T}_\mathcal{I}$ and $\mathcal{T}_\mathcal{J}$, and let $\mathcal{T} = (V, E)$ be a labeled tree with $V \subseteq \mathcal{T}_\mathcal{I} \times \mathcal{T}_\mathcal{J}$ and labels $L := \mathcal{P}(\mathcal{I} \times \mathcal{J})$. Then $\mathcal{T}$ is called a block tree of $\mathcal{I} \times \mathcal{J}$ based on the cluster trees $\mathcal{T}_\mathcal{I}$ and $\mathcal{T}_\mathcal{J}$ if

1. the root is given by $\operatorname{root}(\mathcal{T}) = (\operatorname{root}(\mathcal{T}_\mathcal{I}), \operatorname{root}(\mathcal{T}_\mathcal{J}))$,
2. for all $b = (t, s) \in \mathcal{T}$, the label is $\hat{b} = \hat{t} \times \hat{s}$,

3. if $b = (t, s) \in \mathcal{T}$ has successors, then there holds

$$\mathrm{S}(b) = \left\{ \left(t', s'\right) \,:\, t' \in \mathrm{S}(t), s' \in \mathrm{S}(s) \right\}.$$

In the following, we denote a block cluster tree for cluster trees $\mathcal{T}_\mathcal{I}$ and $\mathcal{T}_\mathcal{J}$ by $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ and denote the set of leaves by $\mathcal{L}_{\mathcal{I}\times\mathcal{J}} := \mathcal{L}(\mathcal{T}_{\mathcal{I}\times\mathcal{J}})$. Furthermore, we will use blocks $b \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ as identifiers for their labels $\widehat{b}$, for example, in $\mathbf{M}|_b = \mathbf{M}|_{\widehat{b}}$ for $\mathbf{M} \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$.

The set of leaves $\mathcal{L}_{\mathcal{I}\times\mathcal{J}}$ yields a partition of the product $\mathcal{I} \times \mathcal{J}$. Given cluster trees $\mathcal{T}_\mathcal{I}$ and $\mathcal{T}_\mathcal{J}$, a block cluster tree $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ may be constructed using a so-called admissibility condition to distinguish between inadmissible blocks (to be further refined if possible) and admissible blocks as follows.

Definition 4. Let $\mathcal{I}$ and $\mathcal{J}$ be index sets with cluster trees $\mathcal{T}_\mathcal{I}$ and $\mathcal{T}_\mathcal{J}$, respectively. Then an admissibility condition is a mapping

$$\mathrm{adm} \,:\, \mathcal{T}_\mathcal{I} \times \mathcal{T}_\mathcal{J} \to \{\mathrm{true}, \mathrm{false}\}.$$

With such an admissibility condition, we can construct a block cluster tree $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ as follows.

1. Set the root as $\mathrm{root}(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}) = (\mathrm{root}(\mathcal{T}_\mathcal{I}), \mathrm{root}(\mathcal{T}_\mathcal{J}))$.
2. For an already constructed block $b = (t, s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$, we set

$$\mathrm{S}(b) = \begin{cases} \emptyset & \text{if } \mathrm{adm}(t,s) = \mathrm{true} \\ & \text{or } \mathrm{S}(t) = \emptyset \text{ or } \mathrm{S}(s) = \emptyset, \\ \left\{ \left(t', s'\right) \,:\, t' \in \mathrm{S}(t), s' \in \mathrm{S}(s) \right\} & \text{else.} \end{cases}$$

We distinguish between admissible and inadmissible leaves,

$$\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}} := \left\{ b = (t,s) \in \mathcal{L}_{\mathcal{I}\times\mathcal{J}} \,:\, \mathrm{adm}(t,s) = \mathrm{true} \right\},$$
$$\mathcal{L}^-_{\mathcal{I}\times\mathcal{J}} = \left\{ b = (t,s) \in \mathcal{L}_{\mathcal{I}\times\mathcal{J}} \,:\, \mathrm{adm}(t,s) = \mathrm{false} \right\} = \mathcal{L}_{\mathcal{I}\times\mathcal{J}} \setminus \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}.$$

Remark 2. Let $\mathcal{T}_\mathcal{I}$ and $\mathcal{T}_\mathcal{J}$ be cluster trees for (finite) index sets $\mathcal{I}$ and $\mathcal{J}$, respectively. Furthermore, assume that we have additional geometric information about the index sets, that is, that there are $X_i, Y_j \subseteq \mathbb{R}^d$ for all $i \in \mathcal{I}$ and for all $j \in \mathcal{J}$. Then we define $X_s := \bigcup_{i\in s} X_i$ for $s \subseteq \mathcal{I}$ and $Y_t := \bigcup_{j\in t} Y_j$ for $t \subseteq \mathcal{J}$. The standard (or strong, η-) admissibility condition uses this geometric information as follows (cf. [6]):

$$\mathrm{adm}_\eta \,:\, \mathcal{T}_\mathcal{I} \times \mathcal{T}_\mathcal{J} \to \{\mathrm{true}, \mathrm{false}\},$$
$$\mathrm{adm}_\eta(s,t) = \begin{cases} \mathrm{true} & \text{if } \min\left\{\mathrm{diam}_2(X_s), \mathrm{diam}_2(Y_t)\right\} \\ & \quad \le \eta \mathrm{dist}_2\left(X_s, Y_t\right), \\ \mathrm{false} & \text{else} \end{cases}$$

for some $\eta \in \mathbb{R}$. Note that in practice the sets X_s and Y_t are typically replaced by (minimal) bounding boxes containing these sets which facilitates the computation of diameters and distances.

Admissibility conditions should be defined in such a way that matrix blocks corresponding to admissible block clusters allow for accurate low rank approximations.

Definition 5. Let $\mathcal{I}, \mathcal{J}$ be finite index sets and $k \in \mathbb{N}$. Then, we denote the set of matrices with at most rank k by

$$\mathcal{R}(\mathcal{I}, \mathcal{J}, k) := \left\{ \mathbf{M} \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}} \,:\, \mathrm{rank}(\mathbf{M}) \le k \right\}.$$

Note that $\mathbf{M} \in \mathcal{R}(\mathcal{I}, \mathcal{J}, k)$ if and only if there are $\mathbf{A} \in \mathbb{R}^{\mathcal{I}\times k}$ and $\mathbf{B} \in \mathbb{R}^{\mathcal{J}\times k}$ with $\mathbf{M} = \mathbf{A}\mathbf{B}^\mathsf{T}$ (see, for example, [13]), hence

$$\mathcal{R}(\mathcal{I}, \mathcal{J}, k) = \left\{ \mathbf{A}\mathbf{B}^\mathsf{T} \,:\, \mathbf{A} \in \mathbb{R}^{\mathcal{I}\times k}, \mathbf{B} \in \mathbb{R}^{\mathcal{J}\times k} \right\}.$$

We can now formally define hierarchical matrices.

Definition 6 ($\mathcal{H}$-matrix). Let $\mathcal{I}$ and $\mathcal{J}$ be index sets with cluster trees $\mathcal{T}_\mathcal{I}$ and $\mathcal{T}_\mathcal{J}$ and a block cluster tree $\mathcal{T}_{\mathcal{I}\times\mathcal{J}}$ constructed with an admissibility condition adm. Then a matrix $\mathbf{H} \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$ is called an $\mathcal{H}$-matrix with local rank $k \in \mathbb{N}$ if for each admissible block $b = (t, s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}$ there are $\mathbf{A}_b \in \mathbb{R}^{t\times k}$ and $\mathbf{B}_b \in \mathbb{R}^{s\times k}$ with

$$\mathbf{H}|_{t\times s} = \mathbf{A}_b \mathbf{B}_b^\mathsf{T} \in \mathcal{R}(t, s, k).$$

We denote the set of $\mathcal{H}$-matrices with local rank at most k by $\mathcal{H}(\mathcal{T}_{\mathcal{I}\times\mathcal{J}}, k)$.

5 | Clustering Strategies

Our goal is to use hierarchical matrices to compute the approximations $\widetilde{\mathbf{F}}, \widetilde{\mathbf{S}}$ outlined through the computational steps ①–⑤ in the block triangular preconditioner (2). To do so, we require cluster trees $\mathcal{T}_\mathcal{I}$ associated with velocity degrees of freedom and $\mathcal{T}_\mathcal{J}$ associated with pressure degrees of freedom. Once such cluster trees (and suitable admissibility conditions) are available, they can be used to construct block cluster trees as outlined in Definition 4 for the representation of the involved matrix blocks, in particular

1. a block cluster tree $\mathcal{T}_{\mathcal{I}\times\mathcal{I}}$ for $\check{\mathbf{F}}_\mathcal{H}$ and its $\mathcal{H}$-LU factorization,
2. a block cluster tree $\mathcal{T}_{\mathcal{J}\times\mathcal{I}}$ for the off-diagonal, rectangular blocks $\mathbf{B}^1_\mathcal{H}, \ldots, \mathbf{B}^d_\mathcal{H}$ as well as for the intermediate results $\mathbf{V}^1, \ldots, \mathbf{V}^d$ and $\mathbf{W}^1, \ldots, \mathbf{W}^d$,
3. a block cluster tree $\mathcal{T}_{\mathcal{J}\times\mathcal{J}}$ for the Schur complement approximation $\mathbf{S}_\mathcal{H}$ and its $\mathcal{H}$-LU factorization.

We will compare two different approaches for the generation of these cluster trees $\mathcal{T}_\mathcal{I}$ and $\mathcal{T}_\mathcal{J}$.

First, in Section 5.1, we briefly review an *uncoupled clustering*. This approach was originally introduced in [4] and constructs the two cluster trees independently. The cluster tree $\mathcal{T}_\mathcal{I}$ constructed with a nested dissection approach which is favorable for the computation of the $\mathcal{H}$-LU factorization $\check{\mathbf{L}}_\mathbf{F}\check{\mathbf{U}}_\mathbf{F} \approx \check{\mathbf{F}}$. The cluster tree $\mathcal{T}_\mathcal{J}$ is generated using a bisection approach which is favorable for the representation of the (approximate) Schur complement $\mathbf{S}_\mathcal{H}$ which can be expected to be dense when computed with exact arithmetic.

Second, in Section 5.2, we will describe a novel *coupled clustering* strategy which we have introduced in [11]. The main motivation is to also take the sparsity structure of the matrices $\mathbf{B}^1, \ldots, \mathbf{B}^d$ into

consideration for the construction of the cluster tree $\mathcal{T}_{\mathcal{I}}$ which is relevant for steps ②–④ in the computation of the preconditioner. In the coupled clustering, the nested dissection clustering of the velocity index set $\mathcal{I}$ is based on a prior (geometric) bisection clustering of the pressure index set $\mathcal{J}$ such that the hierarchical block structure of the matrices $\mathbf{B}^1, \ldots, \mathbf{B}^d$ contains large zero blocks. This block structure is favorable for the (time consuming) explicit computation of an (approximate) Schur complement and leads to a speed-up of the preconditioner set-up. We extend the results presented in [11] by proving a logarithmic bound for the depth of the new cluster tree. Furthermore, we introduce adapted cluster tree construction, the *coupled clustering with interface decomposition*, which aims to further improve the coupled clustering. The estimate for the tree depth as well as the clustering with interface decomposition have been developed in [14] but not been published in a journal yet.

5.1 | Uncoupled Clustering

In [4], the two cluster trees $\mathcal{T}_{\mathcal{I}}, \mathcal{T}_{\mathcal{J}}$ were generated so that the block cluster trees $\mathcal{T}_{\mathcal{I}\times\mathcal{I}}$ and $\mathcal{T}_{\mathcal{J}\times\mathcal{J}}$ are favorable for the computations of the $\mathcal{H}$-LU factorizations of the matrices $\check{\mathbf{F}}$ and $\mathbf{S}_{\mathcal{H}}$. The matrix $\check{\mathbf{F}}$ is sparse, and a suitable clustering for its $\mathcal{H}$-LU factorization is obtained through a domain decomposition based cluster tree $\mathcal{T}_{\mathcal{I}}$ [10]. This clustering strategy yields a block-arrowhead structure for $\check{\mathbf{F}}$ which is favorable for the computation of an $\mathcal{H}$-LU factorization (since certain off-diagonal zero blocks remain zero) and potential for parallelization. In domain decomposition clustering, the index set $\mathcal{I}$ is split into three subsets, s_1, s_2, s_3, on the first level of the cluster tree, as illustrated in Figure 1 (right).

The matrix $\mathbf{S}_{\mathcal{H}}$, on the other hand, is an approximation of the typically dense Schur complement. Therefore, the cluster tree $\mathcal{T}_{\mathcal{J}}$ is generated with geometric bisection, that is, the index set $\mathcal{J}$ is split into two subsets, r_1 and r_2, on the first level of the cluster tree $\mathcal{T}_{\mathcal{J}}$, as illustrated in Figure 1 (left).

In addition to the partitioning, the clustering induces the following reordering of the index sets. For $\mathcal{I}$, the indices in s_1 are ordered first, those in s_2 second and indices in s_3 last. The clusters corresponding to s_1 and s_2 are called *domain clusters* and are (geometrically) separated by the *interface cluster* associated with s_3. For $\mathcal{J}$, the indices in r_1 are ordered first and the indices in r_2 last.

However, the reordering also affects the block structure of the matrices $\mathbf{B}^1, \ldots, \mathbf{B}^d$ (and therefore $\mathbf{V}^1, \ldots \mathbf{V}^d$ as well as $\mathbf{W}^1, \ldots, \mathbf{W}^d$). Figure 2 depicts the sparsity pattern of the matrix $\mathbf{B}^1$ and the block structure defined by the first level of the block cluster tree $\mathcal{T}_{\mathcal{J}\times\mathcal{I}}$ for the 2D example from Figure 1.

We can observe that there are just a few non-zero entries in the blocks $\mathbf{B}^1|_{r_1\times s_2}$ and $\mathbf{B}^1|_{r_2\times s_2}$. Due to these non-zero entries, we have $\mathrm{adm}(r_1, s_2) = \mathrm{false} = \mathrm{adm}(r_2, s_1)$ for the (standard) admissibility condition presented in Remark 2. Hence, these blocks have to be partitioned further. The underlying idea of the following

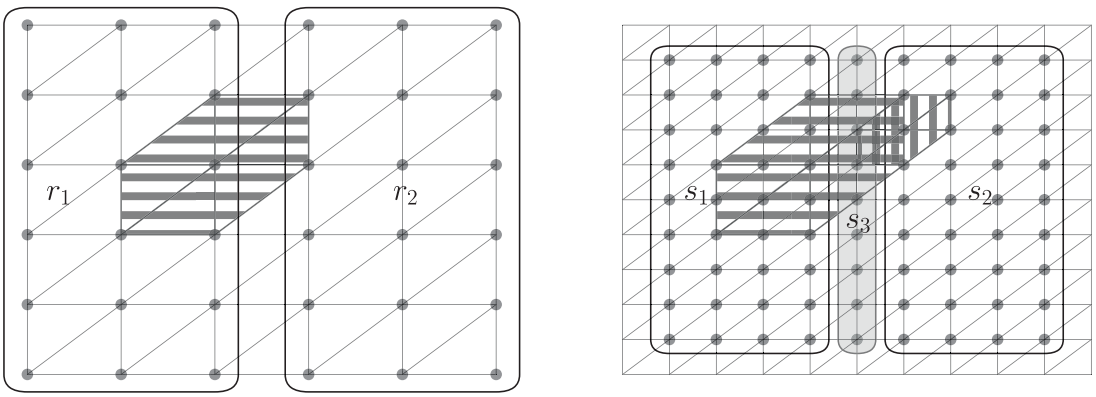

FIGURE 1 | 2D example of the uncoupled clustering and the effect of its induced reordering of the index sets $\mathcal{I}$ and $\mathcal{J}$. Left: the first division of $\mathcal{J}$ into r_1 and r_2, right: the division of $\mathcal{I}$ into the domain clusters s_1 and s_2 and the interface cluster s_3 (marked light gray). These figures also show the support of a pressure basis function Ψ_j with $j \in r_1$ (marked dark gray, hatched horizontally) and the support of a velocity basis function φ_i with $i \in s_2$ (dark gray, hatched vertically).

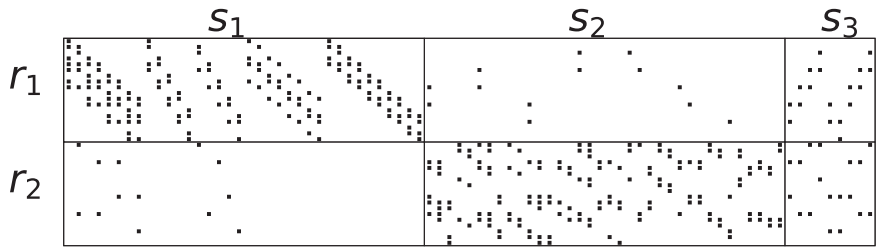

FIGURE 2 | Sparsity pattern of the matrices B^k, $k = 1, \ldots, d$, with reordered rows/columns and the block structure induced by the uncoupled clustering.

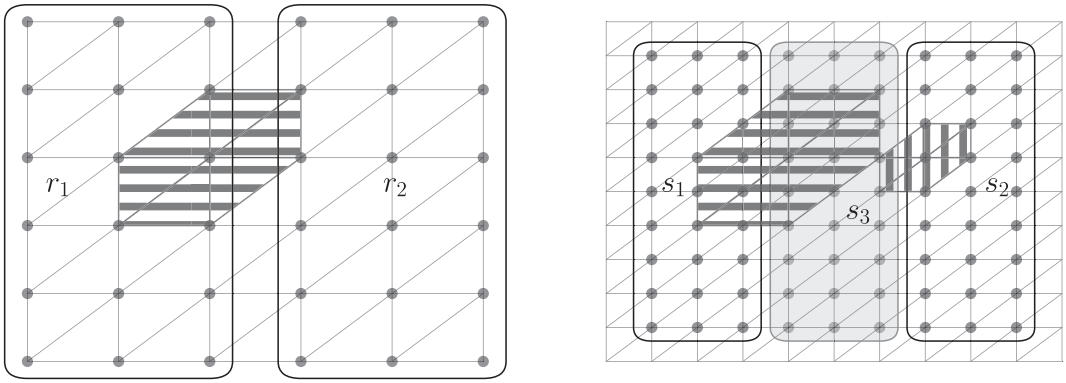

FIGURE 3 | 2D example of the uncoupled clustering and the effect of its induced reordering of the index sets $\mathcal{I}$ and $\mathcal{J}$. Left: the first division of $\mathcal{J}$ into r_1 and r_2, right: the division of $\mathcal{I}$ into the domain clusters s_1 and s_2 and the interface cluster s_3 (marked light gray). These figures also show the support of a pressure basis function Ψ_j with $j \in r_1$ (marked dark gray, hatched horizontally) and the support of a velocity basis function φ_i with $i \in s_2$ (dark gray, hatched vertically).

coupled clustering is to (slightly) increase the interface cluster s_3 such that the two blocks $\mathbf{B}^1|_{r_1 \times s_2}$ and $\mathbf{B}^1|_{r_2 \times s_2}$ become zero blocks.

5.2 | Coupled Clustering

In the following, we will use the geometric information of the mixed finite element discretization described in Section 2. As introduced in Remark 2 for the standard admissibility condition, we define for vertices $\xi_1, \ldots, \xi_n$ and $\chi_1, \ldots, \chi_m$ the supports of the respective (hat) basis functions as

$$X_i := \operatorname{supp}(\varphi_i) \quad \text{and} \quad Y_j := \operatorname{supp}(\Psi_j) \tag{3}$$

for $i \in \mathcal{I}$ and $j \in \mathcal{J}$, respectively. For $s \subseteq \mathcal{I}$ and $t \subseteq \mathcal{J}$, we define

$$X_s := \bigcup_{i \in s} X_i \quad \text{and} \quad Y_t := \bigcup_{j \in t} Y_j.$$

Furthermore, we assume that we have already constructed a cluster tree $\mathcal{T}_\mathcal{J}$ based on (geometric) bisection (as in the uncoupled clustering). Hence, $\mathcal{T}_{\mathcal{J}\times\mathcal{J}}$ will be a suitable cluster tree for the (approximate) Schur complement and its $\mathcal{H}$-LU factorization. Next, we adapt the domain decomposition clustering to generate $\mathcal{T}_\mathcal{I}$ to yield a suitable block structure $\mathcal{T}_{\mathcal{J}\times\mathcal{I}}$ for the (sparse) matrices $\mathbf{B}^1, \ldots, \mathbf{B}^d$. Since these matrices describe a coupling between the pressure and the velocity space, we couple the clustering of $\mathcal{I}$, that is, the generation of $\mathcal{T}_\mathcal{I}$, to the cluster tree $\mathcal{T}_\mathcal{J}$. All domain clusters of $\mathcal{T}_\mathcal{I}$ will be associated with clusters of $\mathcal{T}_\mathcal{J}$. We start with the entire index set $\mathcal{I}$ and associate it with the root $r_\mathcal{J} := \operatorname{root}(\mathcal{T}_\mathcal{J})$. We then decompose $\mathcal{I}$ according to the successors of its associated cluster $r_\mathcal{J}$. In particular, if $r_\mathcal{J}$ has successors, we define

$$\begin{aligned} s_1 &:= \{ i \in \mathcal{I} : \lambda(X_i \cap Y_{r_2}) = 0 \}, \\ s_2 &:= \{ i \in \mathcal{I} : \lambda(X_i \cap Y_{r_1}) = 0 \}, \\ s_3 &:= \mathcal{I} \setminus (s_1 \cup s_2), \end{aligned}$$

where λ is the Lebesgue measure. The subsets s_1 and s_2 are now geometrically separated from r_2 and r_1, respectively, as illustrated in Figure 3. We associate s_1 with r_1 and s_2 with r_2.

ALGORITHM 1 | Coupled clustering algorithm.

Input: Subset $s \subseteq \mathcal{I}$, associated cluster $r \in \mathcal{T}_\mathcal{J}$, leaf size bound n_{leaf} for interface clusters
Output: Cluster tree with root t and $\widehat{t} = s$

```
function BuildCoupledCluster(s, r, n_leaf)
    Create new cluster t with t̂ = s
    S(t) ← ∅
    if S(r) ≠ ∅ then
        {r_1, r_2} ← S(r)

        s_1 ← {i ∈ s : λ(X_i ∩ Y_{r_2}) = 0}
        s_2 ← {i ∈ s : λ(X_i ∩ Y_{r_1}) = 0}
        s_3 ← s \ (s_1 ∪ s_2)

        t_1 ← BuildCoupledCluster(s_1, r_1, n_leaf)
        t_2 ← BuildCoupledCluster(s_2, r_2, n_leaf)
        t_3 ← BuildInterface(s_3, n_leaf, 1)
        S(t) ← {t_1, t_2, t_3}
    end if
    return t
end function
```

As a result, we have

$$\mathbf{B}^k\Big|_{r_1 \times s_2} = 0 \quad \text{and} \quad \mathbf{B}^k\Big|_{r_2 \times s_1} = 0$$

for $k = 1, \ldots, d$, as depicted in Figure 4.

We continue to subdivide the domain clusters s_1 and s_2 by the same approach, that is, with respect to their associated clusters r_1 and r_2, respectively. The interface cluster s_3, on the other hand, is handled as described for interface clusters in the domain decomposition clustering. This is summarized in Algorithm 1.

Remark 3. The proposed coupled clustering easily generalizes to other types of mixed finite element discretizations. For example, in the case of Taylor-Hood elements $Q^h := \mathrm{P}^1(\mathcal{T}^h)$ and $V^h := \mathrm{P}^2(\mathcal{T}^h)$, the subdivision of a (velocity) domain cluster into three successors s_1, s_2, s_3 is still associated to a pressure cluster and based on the supports of the basis functions associated with

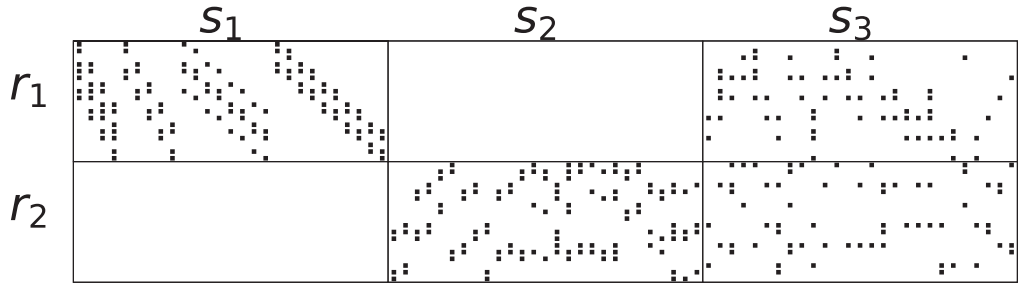

FIGURE 4 | Sparsity pattern of the matrices B^k, $k = 1, \ldots, d$, with reordered rows and columns induced by the coupled clustering.

the indices. Since now the supports of (some of the) velocity basis functions have the same (and not half the) size of the supports of the pressure basis functions, this leads to an enlarged separator s_3. This in turn will affect the constants in the complexity estimates but not the complexity order.

Remark 4.

1. Let $B_{\mathcal{J}} \supseteq \{\chi_1, \ldots, \chi_m\}$ and $B_{\mathcal{I}} \supseteq \{\xi_1, \ldots, \xi_n\}$ be (minimal) bounding boxes, that is, the smallest boxes containing $\{\chi_1, \ldots, \chi_m\}$ and $\{\xi_1, \ldots, \xi_n\}$, respectively. For the generation of $\mathcal{T}_{\mathcal{J}}$ with geometric bisection, bounding box $B_{\mathcal{J}}$ is split along the longest axis: Let $k \in \{1, \ldots, d\}$ be such that for $B_{\mathcal{J}} = \prod_{i=1}^{d} [a_i, b_i]$, there holds

$$k = \arg \max_{i \in \{1, \ldots, d\}} \{b_i - a_i\}.$$

   Since $\mathcal{T}^{h/2}$ was refined from $\mathcal{T}^h$, we have $B_{\mathcal{I}} \subseteq B_{\mathcal{J}}$. The clusters r_1 and r_2 are generated by splitting $B_{\mathcal{J}}$ along the k-th axis with $m_k = \frac{1}{2}(a_k + b_k)$ into

$$\begin{aligned} B_1 &= [a_1, b_1] \times \cdots \times [a_{k-1}, b_{k-1}] \times [a_k, m_k] \\ &\quad \times [a_{k+1}, b_{k+1}] \times \cdots \times [a_d, b_d], \\ B_2 &= [a_1, b_1] \times \cdots \times [a_{k-1}, b_{k-1}] \times [m_k, b_k] \\ &\quad \times [a_{k+1}, b_{k+1}] \times \cdots \times [a_d, b_d]. \end{aligned}$$

   Let B_{s_3} be a (minimal) bounding box for $\{\xi_i : i \in s_3\}$. Then there holds

$$\begin{aligned} B_{s_3} \subseteq\ & [a_1, b_1] \times \cdots \times [a_{k-1}, b_{k-1}] \\ & \times [m_k - h, m_k + h] \times [a_{k+1}, b_{k+1}] \times \cdots \times [a_d, b_d], \end{aligned}$$

   that is, B_{s_3} has at most width $2h$ along the k-th axis (cf. [14, remark 5.4]).
   The decomposition of s_3 is handled with geometric bisection. As suggested for the domain decomposition clustering in [10], the subdivision of interface clusters is delayed every d-th step such that the diameters of the bounding boxes of the interface clusters and the domain clusters are reduced at comparable rates.

2. Since the triangulation $\mathcal{T}^{h/2}$ is constructed by refinement of $\mathcal{T}^h$, the subsets s_1 and s_2 are also (geometrically) separated from each other. Therefore, we obtain a domain decomposition structure for the block cluster tree $\mathcal{T}_{\mathcal{I}\times\mathcal{I}}$.

3. The common sparsity pattern of the matrices $\mathbf{B}^k$, $k = 1, \ldots, d$, is closely related to the sets X_i and Y_j ($i \in \mathcal{I}, j \in \mathcal{J}$) defined in (3). An entry b^k_{ji} can only be non-zero if $\lambda(X_i \cap Y_j) > 0$.

For the further discussion, we introduce the following terminology.

Definition 7. Let $\mathcal{T}_{\mathcal{I}}$ be a cluster tree generated with Algorithm 1.

1. As for the domain decomposition clustering, we call clusters $s \in \mathcal{T}_{\mathcal{I}}$ generated with `BuildInterface`interface clusters. On the other hand, clusters generated with `BuildCoupledCluster` are called domain clusters. We denote the set of domain clusters of $\mathcal{T}_{\mathcal{I}}$ by C_{dom}.

2. As noted before, each domain cluster $s \in C_{\text{dom}}$ is generated by Algorithm 1 with respect to an associated cluster $t_s \in \mathcal{T}_{\mathcal{J}}$. We define the mapping

$$^{\text{a}} : C_{\text{dom}} \to \mathcal{T}_{\mathcal{J}};\ s \mapsto s^a := t_s$$

   to formalize this association.

If a cluster tree $\mathcal{T}_{\mathcal{I}}$ based on domain decomposition clustering (see, e.g., Section 5.1) is used to construct a block cluster tree $\mathcal{T}_{\mathcal{I}\times\mathcal{I}}$, then one typically uses the following *domain decomposition admissibility condition* (cf. [10])

$$\text{adm}_{\text{DD}} : \mathcal{T}_{\mathcal{I}} \times \mathcal{T}_{\mathcal{I}} \to \{\text{true}, \text{false}\}$$

$$\text{adm}_{\text{DD},\eta}(s,t) = \begin{cases} \text{true} & \text{if } s,t \in C_{\text{dom}} \text{ with } s \neq t, \\ \text{adm}_{\eta} & \text{else.} \end{cases}$$

The association of domain clusters from the velocity cluster tree $\mathcal{T}_{\mathcal{I}}$ and clusters from the pressure cluster tree $\mathcal{T}_{\mathcal{J}}$ allows us to introduce the following coupled admissibility condition for the construction of block cluster trees $\mathcal{T}_{\mathcal{J}\times\mathcal{I}}$.

Definition 8 (Coupled admissibility condition). Let $\mathcal{T}_{\mathcal{J}}$ be a cluster tree generated with geometric bisection. Let $\mathcal{T}_{\mathcal{I}}$ be a cluster tree generated with the coupled clustering from Algorithm 1 and with respect to $\mathcal{T}_{\mathcal{J}}$. Then we call the admissibility condition

$$\text{adm}^{\text{c}}_{\eta} : \mathcal{T}_{\mathcal{J}} \times \mathcal{T}_{\mathcal{I}} \to \{\text{true}, \text{false}\}$$

with

$$\mathrm{adm}_\eta^{\mathrm{c}}(r,s) = \begin{cases} \text{true} & \text{if } s \in C_{\mathrm{dom}} \text{ and } r \neq s^{\mathrm{a}}, \\ \mathrm{adm}_\eta(r,s) & \text{else} \end{cases}$$

(strong) coupled admissibility condition.

We will now analyze the depth of the block cluster tree $\mathcal{T}_{\mathcal{J}\times\mathcal{I}}$ generated by the coupled clustering. This quantity is an important part in complexity estimates for hierarchical matrices (see, e.g., [6]) and typically yields the logarithmic part in the log-linearity of the complexity of arithmetic operations. Our goal is to prove a logarithmic bound for the tree depth. Note that this requires an additional assumption about the supports X_i, $i \in \mathcal{I}$ and Y_j, $j \in \mathcal{J}$. Due to the finite element context, a suitable assumption that was used, for example, in [5] and [10], is that the sets Y_j, $j \in \mathcal{J}$ are locally separated, that is, there are constants $C_{\mathrm{sep}} > 0$ and $n_{\min} \in \mathbb{N}$ with

$$\max_{j\in\mathcal{J}} \left| \left\{ k \in \mathcal{J} \,:\, \mathrm{dist}_2(Y_j, Y_k) \leq C_{\mathrm{sep}}^{-1} \mathrm{diam}_2(Y_j) \right\} \right| \leq n_{\min}.$$

These constants depend on the triangulation $\mathcal{T}^h$ (for details see, for example, the discussion in [13, section 6.4.3.2]). Since $\mathcal{T}^{h/2}$ was refined from $\mathcal{T}^h$ into eight subtetrahedra using the edge midpoints, this property is transfered to the sets X_i, $i \in \mathcal{I}$, that is, we have

$$\max_{i\in\mathcal{I}} \left| \left\{ \ell \in \mathcal{J} \,:\, \mathrm{dist}_2(X_i, X_\ell) \leq C_{\mathrm{sep}}^{-1} \mathrm{diam}_2(X_i) \right\} \right| \leq n_{\min}.$$

Additionally, we assume that $n_{\min}$ satisfies

$$n_{\min} > 4^d \left(\frac{h}{h_{\min}} \right)^d \tag{4}$$

where h is the mesh width of the triangulation $\mathcal{T}^h$ and

$$h_{\min} := \min_{\substack{\mathbf{p},\mathbf{q}\in\mathcal{V}(\mathcal{T}^h) \\ \mathbf{p}\neq\mathbf{q}}} \|\mathbf{p}-\mathbf{q}\|_2.$$

This allows the following estimate.

Theorem 1. *Let $\mathcal{T}_{\mathcal{J}}$ be a cluster tree generated with geometric bisection and leaf size bound $n_{\mathrm{leaf}} \geq n_{\min}$. Let $\mathcal{T}_{\mathcal{I}}$ be a cluster tree generated with the coupled clustering, i.e., with Algorithm 1 and with respect to $\mathcal{T}_{\mathcal{J}}$. Let $\underline{h} := \min_{i\in\mathcal{I}} \mathrm{diam}_2(X_i)$ and $\delta := \mathrm{diam}_\infty(B_{\mathcal{J}})$. Then the depth of the cluster tree $\mathcal{T}_{\mathcal{I}}$ can be estimated by*

$$\mathrm{depth}(\mathcal{T}_{\mathcal{I}}) \leq d \log_2 \left(4\sqrt{d} C_{\mathrm{sep}} \delta \underline{h}^{-1} \right) = \mathcal{O}\left(\log_2 \left(\underline{h}^{-d} \right) \right).$$

Proof. We will prove the estimate by defining an upper bound for the level of the clusters in $\mathcal{T}_{\mathcal{I}}$. Therefore, let $s \in \mathcal{T}_{\mathcal{I}}$. Then s is either a domain cluster or an interface cluster (cf. Definition 7).

In the first case, that is, if s is a domain cluster, we can estimate the level of the associated cluster s^{a}, for example, with the result from [5]. Since $\mathcal{T}^{h/2}$ was refined from $\mathcal{T}^h$ into eight subtetrahedra using the edge midpoints, $\min_{j\in\mathcal{J}} Y_j = 2\underline{h}$ holds. Therefore, we obtain

$$\begin{aligned} \mathrm{level}(s) = \mathrm{level}(s^{\mathrm{a}}) &\leq d \log_2 \left(2\sqrt{d} C_{\mathrm{sep}} \delta (2\underline{h})^{-1} \right) \\ &= d \log_2 \left(\sqrt{d} C_{\mathrm{sep}} \delta \underline{h}^{-1} \right) \\ &< d \log_2 \left(4\sqrt{d} C_{\mathrm{sep}} \underline{h}^{-1} \right). \end{aligned}$$

Now, assume that s is an interface cluster. Let r be the nearest predecessor of s to be a domain cluster. Then there is a path

$$p = (r_0, r_1, \ldots, r_\ell)$$

of length $\ell \in \mathbb{N}$ from $r_0 = r$ to $r_\ell = s$. Since r is the nearest predecessor of s to be a domain cluster, the clusters $r_1, \ldots, r_{k-1}$ are all interface clusters. This yields

$$\mathrm{level}(s) = \mathrm{level}(r) + \ell. \tag{5}$$

So, it remains to estimate $\mathrm{level}(r)$ and ℓ. The first can again be estimated with the result in [5]:

$$\mathrm{level}(r) = \mathrm{level}(r^{\mathrm{a}}) < d \log_2 \left(2\delta \frac{1}{\mathrm{diam}_\infty(B_{r^{\mathrm{a}}})} \right). \tag{6}$$

To estimate ℓ, we will use the same techniques used for the proof for this estimate. First note, that since $\mathcal{T}^{h/2}$ was refined from $\mathcal{T}^h$ into eight subtetrahedra using the edge midpoints, we have

$$\min_{\substack{\mathbf{p},\mathbf{q}\in\mathcal{V}(\mathcal{T}^h) \\ \mathbf{p}\neq\mathbf{q}}} \|\mathbf{p}-\mathbf{q}\|_2 = \frac{1}{2} \min_{K\in\mathcal{T}^h} \min_{\substack{\mathbf{p},\mathbf{q}\in K \\ \mathbf{p}\neq\mathbf{q}}} \|\mathbf{p}-\mathbf{q}\| = \frac{1}{2} h_{\min}.$$

This implies that there is at least one axis of B_s with a larger width than $2h$, since otherwise (4) would yield

$$|s| \leq \left(\frac{\mathrm{diam}_\infty(B_s)}{\frac{1}{2}h_{\min}} \right)^d \leq \left(\frac{4h}{h_{\min}} \right)^d \overset{(4)}{<} n_{\min} \leq |s|.$$

Due to the delayed subdivision every d-th step, the diameter of the bounding boxes is at least halved after at most d steps, that is,

$$\mathrm{diam}_\infty(B_{r_{k+d}}) \leq \frac{1}{2} \mathrm{diam}_\infty(B_k)$$

holds for $k = 1, \ldots, \ell - d$. Hence, we have

$$\begin{aligned} \mathrm{diam}_\infty(B_s) = \mathrm{diam}_\infty(B_{r_\ell}) &\leq 2^{-\left\lfloor \frac{\ell}{d} \right\rfloor} \mathrm{diam}_\infty(B_{r_1}) \\ &\leq 2^{-\left\lfloor \frac{\ell}{d} \right\rfloor} \mathrm{diam}_\infty(B_r) \\ &\leq 2^{-\left\lfloor \frac{\ell}{d} \right\rfloor} \mathrm{diam}_\infty(B_{r^{\mathrm{a}}}). \end{aligned} \tag{7}$$

Now, we assume that s is not a leaf, that is, $|s| > n_{\mathrm{leaf}} \geq n_{\min}$. Since the sets X_i are locally separated, there have to be $i, j \in s$ with

$$\mathrm{dist}_2(X_i, X_j) > C_{\mathrm{sep}} \mathrm{diam}_2(X_i).$$

This directly yields

$$\begin{aligned} \mathrm{diam}_2(B_s) \geq \mathrm{dist}_2(\xi_i, \xi_j) &\geq \mathrm{dist}_2(X_i, X_j) \\ &> C_{\mathrm{sep}}^{-1} \mathrm{diam}_2(X_i) \geq C_{\mathrm{sep}}^{-1} \underline{h}. \end{aligned} \tag{8}$$

Together with (7)), we therefore obtain

$$
\begin{aligned}
2^{\left\lfloor \frac{\ell}{d} \right\rfloor} &\overset{(7)}{\leq} \frac{\operatorname{diam}_\infty\left(B_{r^a}\right)}{\operatorname{diam}_\infty^s} \\
&\leq \frac{\operatorname{diam}_\infty B_{r^a}}{\operatorname{diam}_2\left(B_s\right)} \\
&\overset{(8)}{<} C_{\mathrm{sep}} \sqrt{d} \operatorname{diam}_\infty\left(B_{r^a}\right) \underline{h}^{-1}.
\end{aligned} \tag{9}
$$

By taking the logarithm on both sides, we finally obtain

$$
\begin{aligned}
\ell = d\frac{\ell}{d} &\leq d\left(\left\lfloor \frac{\ell}{d} \right\rfloor + 1\right) \\
&\overset{(9)}{<} d \log_2\left(\sqrt{d} C_{\mathrm{sep}} \operatorname{diam}_\infty\left(B_{r^a}\right) \underline{h}^{-1}\right) + d.
\end{aligned} \tag{10}
$$

Combining (10) with (5) and (6) then finally yields

$$
\begin{aligned}
\operatorname{level}(s) &\overset{(5)}{=} \operatorname{level}(r) + \ell \overset{(6)}{<} d \log_2\left(2\delta \frac{1}{\operatorname{diam}_\infty B_{r^a}}\right) + \ell \\
&\overset{(10)}{<} d \log_2\left(2\delta \frac{1}{\operatorname{diam}_\infty B_{r^a}}\right) \\
&\quad + d \log_2\left(\sqrt{d} C_{\mathrm{sep}} \operatorname{diam}_\infty\left(B_{r^a}\right) \underline{h}^{-1}\right) + d \\
&= d \log_2\left(2\sqrt{d} C_{\mathrm{sep}} \delta \underline{h}^{-1}\right) + d \\
&= d \log_2\left(4\sqrt{d} C_{\mathrm{sep}} \delta \underline{h}^{-1}\right).
\end{aligned}
$$

□

5.3 | Coupled Clustering with Interface Decomposition

With the coupled clustering, we introduced a cluster strategy similar to the domain decomposition clustering. One important advantage of the domain decomposition clustering is the resulting block-arrowhead structure for the matrix $\mathbf{F}$. This is advantageous for the computation of an ($\mathcal{H}$-)LU factorization of this matrix. Although the coupled clustering also results in a block-arrowhead structure, the larger interface created by this cluster strategy is less advantageous for the computation of an ($\mathcal{H}$-)LU factorization. In order to tackle this disadvantage of the coupled clustering, we consider the same 2D example as in the previous section. The coupled clustering splits the index set $\mathcal{I}$ into three parts s_1, s_2, and s_3. In addition to the reordering induced by the coupled clustering (s_1 before s_2, s_3 last), we order the interface s_3 as follows: the nodes neighboring neither s_1 nor s_2 are ordered first, then the nodes neighboring s_1 and the nodes neighboring s_2 are ordered last. The result of this ordering is illustrated in Figure 5.

We can observe that the non-zero entries in $\check{\mathbf{F}}|_{s_3 \times s_1}$ and $\check{\mathbf{F}}|_{s_3 \times s_2}$ are in about a third of the rows of the corresponding blocks. In particular, the sparsity plot in Figure 5 suggests that, for example, the nodes neighboring s_2 (which are ordered last in s_3) are separated from the nodes in s_1. The same holds for the nodes neighboring neither s_1 nor s_2, which are ordered first in s_3.

Based on the observations we made, we will now describe a cluster strategy similar to the coupled clustering but with an additional decomposition of the interfaces which we call *coupled clustering with interface decomposition*. As for the coupled clustering, we start with the cluster tree $\mathcal{T}_\mathcal{I}$ already generated by geometric bisection. Then we start the decomposition of $\mathcal{I}$ with respect to the root $r_\mathcal{I} := \operatorname{root}\mathcal{T}_\mathcal{I}$ of $\mathcal{T}_\mathcal{I}$ into five (distinct) subsets. If $r_\mathcal{I}$ has successors, we define

$$
\begin{aligned}
s_1 &:= \left\{i \in \mathcal{I} : \lambda\left(X_i \cap Y_{r_2}\right) = 0\right\}, \\
s_2 &:= \left\{i \in \mathcal{I} : \lambda\left(X_i \cap Y_{r_1}\right) = 0\right\}, \\
s_3 &:= \left\{i \in \mathcal{I} \setminus \left(s_1 \cup s_2\right) : \lambda\left(X_i \cap X_{s_1}\right) = 0 \text{ and } \right. \\
&\qquad \left. \lambda\left(X_i \cap X_{s_2}\right) = 0\right\}, \\
s_4 &:= \left\{i \in \mathcal{I} \setminus \left(s_1 \cup s_2\right) : \lambda\left(X_i \cap X_{s_1}\right) \neq 0 \text{ and } \right. \\
&\qquad \left. \lambda\left(X_i \cap X_{s_2}\right) = 0\right\}, \\
s_5 &:= \left\{i \in \mathcal{I} \setminus \left(s_1 \cup s_2\right) : \lambda\left(X_i \cap X_{s_1}\right) = 0 \text{ and } \right. \\
&\qquad \left. \lambda\left(X_i \cap X_{s_2}\right) \neq 0\right\}.
\end{aligned}
$$

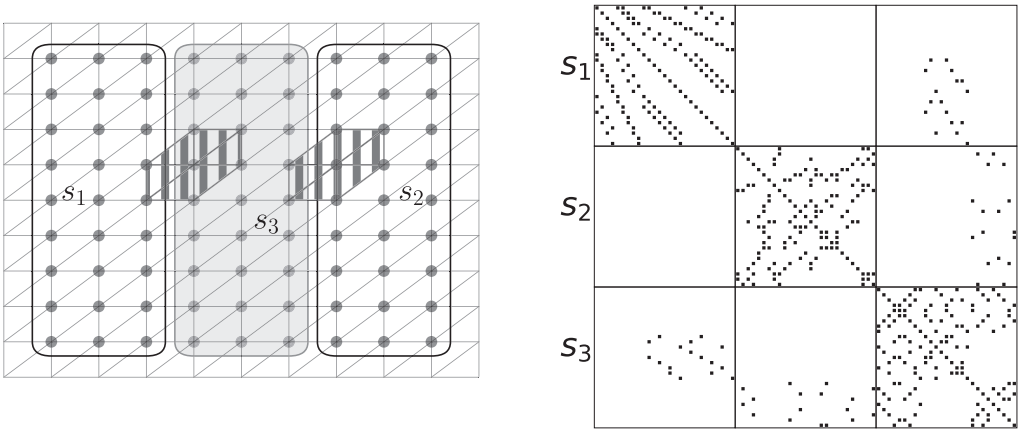

FIGURE 5 | Effect of the reordering induced by the coupled clustering to the structure of $\check{\mathbf{F}}$ for a 2D example. The left figure shows the geometric decomposition of $\mathcal{I}$ into the two domain clusters s_1, s_2 and the interface cluster s_3 (marked light gray). Additionally, it shows the support of a basis function φ_i with $i \in s_2$ and the support of a basis function φ_j with $j \in s_3$ (both dark gray, hatched vertically). The right figure shows the effect of the reordering on the matrix $\check{\mathbf{F}}$. The interface s_3 was ordered internally so that all nodes in the middle are ordered first, then all nodes neighboring s_1 and then all nodes neighboring s_2.

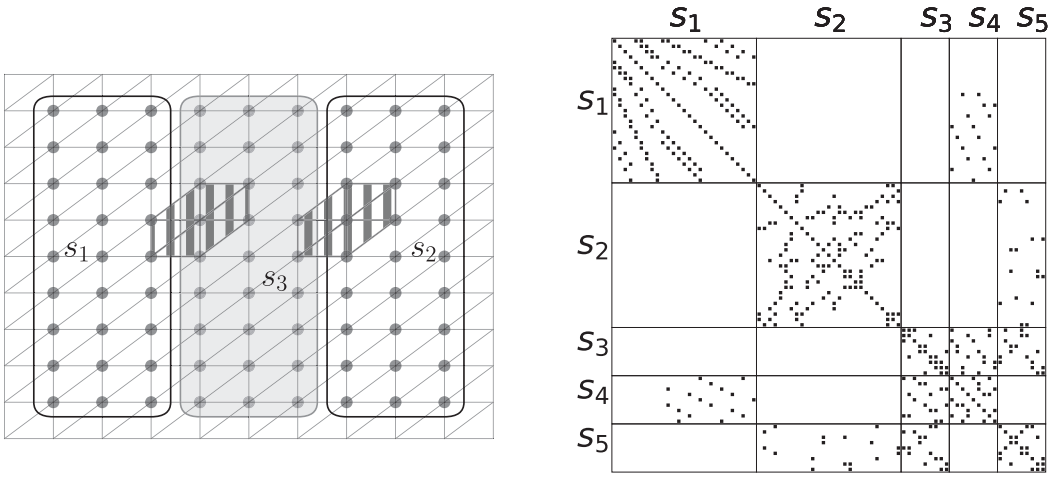

FIGURE 6 | Effect of the reordering induced by the coupled clustering with interface decomposition for a 2D example. The left figure shows the geometric decomposition of $\mathcal{I}$ into the two domain clusters s_1 and s_2 as well as the three interface clusters s_3, s_4, and s_5 (marked light gray). Additionally, it shows the support of a basis function φ_i with $i \in s_2$ and the support of a basis function φ_j with $j \in s_4$ (both dark gray, hatched vertically). The right figure shows the effect of the reordering to the matrix $\check{\mathbf{F}}$.

As before, the sets s_1 and s_2 are (geometrically) separated from r_2 and r_1, respectively. They are subdivided by the same strategy with respect to r_1 and r_2, respectively. The set s_3 is (geometrically) separated from both, s_1 and s_2, while s_4 and s_5 are (geometrically) separated from either s_1 or s_2, respectively. We handle all of them as before the single interface clusters: s_3, s_4, and s_5 are subdivided with geometric bisection. The subdivision is again delayed every d-th step (cf. Remark 4). However, due to the additional decomposition of the interface, the delay is introduced earlier. The clustering is summarized in Algorithm 3.

The induced reordering is the same that was considered in Figure 5 but now with a decomposition of the interface into three parts as depicted in Figure 6.

As in the previous section, we introduce additional terminology allowing us to define an admissibility condition for the construction of the block cluster tree $\mathcal{T}_{\mathcal{I}\times\mathcal{I}}$.

Definition 9. Let $\mathcal{T}_{\mathcal{I}}$ be a cluster tree generated by Algorithm 3, coupled with the cluster tree $\mathcal{T}_{\mathcal{J}}$ generated with geometric bisection.

1. As in Definition 7, we call clusters from $\mathcal{T}_{\mathcal{I}}$ generated by Algorithm 3 domain clusters and clusters from $\mathcal{T}_{\mathcal{I}}$ generated by Algorithm 2 interface clusters. We denote the set of domain clusters of $\mathcal{T}_{\mathcal{I}}$ by $\mathcal{C}_{\text{dom}}(\mathcal{T}_{\mathcal{I}})$ or just $\mathcal{C}_{\text{dom}}$.
2. As in Definition 7, we use the mapping

$$^{\text{a}} : \mathcal{C}_{\text{dom}} \to \mathcal{T}_{\mathcal{J}};\ s \mapsto s^{\text{a}}$$

assigning associated clusters $s^{\text{a}} \in \mathcal{T}_{\mathcal{J}}$ to the corresponding domain clusters $s \in \mathcal{C}_{\text{dom}}$.

3. Let $s \in \mathcal{T}_{\mathcal{I}}$ be a domain cluster with

$$\text{S}(s) = \{s_1, \ldots, s_5\}$$

where $s_1, \ldots, s_5$ are as defined in Algorithm 3. Then s_3 is separated from both domain clusters s_1 and s_2. We call s_3 *separated interface cluster*. The cluster s_4 is separated from s_2 but not from s_1, that is, $X_{s_4} \cap X_{s_2} = \emptyset$ and $X_{s_4} \cap X_{s_1} \neq \emptyset$ holds. The cluster s_5, on the other hand, is separated from s_1 but not from s_2. We say that s_4 is *connected* to s_1 and s_5 is connected to s_2 and call s_4 and s_5 connected interface clusters.
We denote the set of connected interface clusters of $\mathcal{T}_{\mathcal{I}}$ with $\mathcal{C}_{\text{cint}}(\mathcal{T}_{\mathcal{I}})$ or just $\mathcal{C}_{\text{cint}}$ and the set of separated interface clusters of $\mathcal{T}_{\mathcal{I}}$ with $\mathcal{C}_{\text{sint}}(\mathcal{T}_{\mathcal{I}})$ or just $\mathcal{C}_{\text{sint}}$.

4. Let $s \in \mathcal{T}_{\mathcal{I}}$ be a connected interface cluster. Then it is connected to a (unique) domain cluster $t_s \in \mathcal{C}_{\text{dom}}$ with the same

ALGORITHM 2 | Interface clustering algorithm.

Input: Subset $s \subseteq \mathcal{I}$, leaf size bound n_{leaf}, distance ℓ to the nearest domain predecessor
Output: Cluster tree with root t and $\hat{t} = s$

```
function BuildInterface(s, n_leaf, ℓ)
    Create a cluster t with t̂ = s and S(t) = ∅
    if |s|>n_leaf then
        if ℓ mod d>0 then
            Determine bounding box B_s
            Split B_s into B_1 and B_2

            s_1 ← {i ∈ s : x_i ∈ B_1}
            s_2 ← s \ s_1

            t_1 ← BuildInterface(s_1, n_leaf, ℓ + 1)
            t_2 ← BuildInterface(s_2, n_leaf, ℓ + 1)
            S(t) ← {t_1, t_2}
        else
            t' ← BuildInterface(s, n_leaf, ℓ + 1)
            S(t) ← {t'}
        end if
    end if
    return t
end function
```

ALGORITHM 3 | Coupled clustering with interface decomposition.

Input: Subset $s \subseteq \mathcal{I}$, associated cluster $r \in \mathcal{T}_{\mathcal{J}}$, leaf size bound n_{leaf} for interface clusters
Output: Cluster tree with root t and $\hat{t} = s$

function `CoupledIDClustering`(s, r, n_{leaf})
  Create new cluster t with $\hat{t} = s$ and $\mathrm{S}(t) = \emptyset$
  if $\mathrm{S}(r) \neq \emptyset$ **then**
    $\{r_1, r_2\} \leftarrow \mathrm{S}(r)$
    $s_1 \leftarrow \{i \in s : \lambda(X_i \cap Y_{r_2}) = 0\}$
    $s_2 \leftarrow \{i \in s : \lambda(X_i \cap Y_{r_1}) = 0\}$
    $s_3 \leftarrow \{i \in s \setminus (s_1 \cup s_2) : \lambda(X_i \cap X_{s_1}) = 0 \text{ and } \lambda(X_i \cap X_{s_2}) = 0\}$
    $s_4 \leftarrow \{i \in s \setminus (s_1 \cup s_2) : \lambda(X_i \cap X_{s_1}) \neq 0 \text{ and } \lambda(X_i \cap X_{s_2}) = 0\}$
    $s_5 \leftarrow \{i \in s \setminus (s_1 \cup s_2) : \lambda(X_i \cap X_{s_1}) = 0 \text{ and } \lambda(X_i \cap X_{s_2}) \neq 0\}$
    $t_1 \leftarrow$ `CoupledIDClustering`$(s_1, r_1, n_{\text{leaf}})$
    $t_2 \leftarrow$ `CoupledIDClustering`$(s_2, r_2, n_{\text{leaf}})$
    $t_3 \leftarrow$ `BuildInterface`$(s_3, n_{\text{leaf}}, 2)$
    $t_4 \leftarrow$ `BuildInterface`$(s_4, n_{\text{leaf}}, 2)$
    $t_5 \leftarrow$ `BuildInterface`$(s_5, n_{\text{leaf}}, 2)$
    $\mathrm{S}(t) \leftarrow \{t_1, t_2, t_3, t_4, t_5\}$
  end if
  return t
end function

predecessor as s. We define the mapping

$$^{\mathrm{c}} : \mathcal{C}_{\text{cint}} \to \mathcal{C}_{\text{dom}}$$

assigning the corresponding connected domain cluster $s^{\mathrm{c}} = t_s$ to each connected cluster s.

Definition 10. Let $\mathcal{T}_{\mathcal{I}}$ be a cluster tree generated with the coupled clustering with interface decomposition. Then we define the (strong) coupled interface decomposition admissibility condition

$$\mathrm{adm}_{\eta}^{\text{cid}} : \mathcal{T}_{\mathcal{I}} \times \mathcal{T}_{\mathcal{I}} \to \{\text{true}, \text{false}\}$$

with

$$\mathrm{adm}_{\eta}^{\text{cid}}(t,s) = \begin{cases} \text{true} & \text{if } t, s \in \mathcal{C}_{\text{dom}} \text{ and } t \neq s, \\ \text{true} & \text{if } t \in \mathcal{C}_{\text{dom}}, s \in \mathcal{C}_{\text{cint}} \text{ and } t \neq s^{\mathrm{c}}, \\ \text{true} & \text{if } t \in \mathcal{C}_{\text{cint}}, s \in \mathcal{C}_{\text{dom}} \text{ and } t^{\mathrm{c}} \neq s, \\ \text{true} & \text{if } t \in \mathcal{C}_{\text{dom}}, s \in \mathcal{C}_{\text{sint}} \text{ or } t \in \mathcal{C}_{\text{sint}}, s \in \mathcal{C}_{\text{dom}}, \\ \mathrm{adm}_{\eta}(t,s) & \text{else.} \end{cases}$$

6 | Numerical Results

For our numerical tests, we used the Oseen model problem described in Section 2 with viscosity $\nu = 10^{-2}$ and a recirculating convection given by

$$\mathbf{w}_{\text{re}} = \begin{pmatrix} -\sin(\pi x_1)\left(\cos(\pi x_2)\sin(\pi x_3) + \sin(\pi x_2)\cos(\pi x_3)\right) \\ \sin(\pi x_2)\left(\cos(\pi x_1)\sin(\pi x_3) - \sin(\pi x_1)\cos(\pi x_3)\right) \\ \sin(\pi x_3)\left(\cos(\pi x_1)\sin(\pi x_2) + \sin(\pi x_1)\cos(\pi x_2)\right). \end{pmatrix}$$

The notation for the different parameters in the set-up of the involved $\mathcal{H}$-matrices is as follows:

- In all computations involving $\mathcal{H}$-arithmetic, we use adaptive ranks such that the relative truncation errors in admissible matrix blocks are bounded. In particular, we allow for different truncation accuracies in different parts of the $\mathcal{H}$-matrix computations:
  1. $\delta_{\text{vel}}^{\mathcal{H}}$ for the $\mathcal{H}$-LU factorization of $\check{\mathbf{F}}$ (step ①),
  2. $\delta_{\text{mul}}^{\mathcal{H}}$ for the computation of the Schur complement (steps ②–④), and
  3. $\delta_{\text{Schur}}^{\mathcal{H}}$ for the $\mathcal{H}$-LU factorization of the Schur complement (step ⑤).

  If the same truncation accuracy is used for all steps and nothing else is specified, we have used
  $$\delta^{\mathcal{H}} := \delta_{\text{vel}}^{\mathcal{H}} = \delta_{\text{mul}}^{\mathcal{H}} = \delta_{\text{Schur}}^{\mathcal{H}} = 10^{-1}.$$
- The block cluster trees $\mathcal{T}_{\mathcal{I}\times\mathcal{I}}$, $\mathcal{T}_{\mathcal{J}\times\mathcal{I}}$, and $\mathcal{T}_{\mathcal{J}\times\mathcal{J}}$ are constructed from the cluster trees $\mathcal{T}_{\mathcal{I}}$ and $\mathcal{T}_{\mathcal{J}}$ with admissibility conditions
  1. $\mathrm{adm}_{\mathcal{J}\times\mathcal{I}} = \mathrm{adm}_{\eta}^{c}$,
  2. $\mathrm{adm}_{\mathcal{I}\times\mathcal{I}} = \mathrm{adm}_{\text{DD}}$ for the uncoupled and coupled clustering (§6.1) and $\mathrm{adm}_{\mathcal{I}\times\mathcal{I}} = \mathrm{adm}_{\eta}^{\text{cid}}$ for the coupled clustering with interface decomposition (§6.2),
  3. $\mathrm{adm}_{\mathcal{J}\times\mathcal{J}} = \mathrm{adm}_{\eta}$ (since weaker admissibility conditions are not suitable for the representation of the [dense] Schur complement, that is, would result in significantly larger ranks for the [admissible] low-rank blocks).

  We have always used $\eta = 16$.

Furthermore, we use the preconditioned BiCGStab method to solve the linear system described in (1) with the right hand side

$$\begin{pmatrix} \mathbf{r} \\ \mathbf{s} \end{pmatrix} := \mathcal{M} \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}.$$

The BiCGStab method is stoppend when a relative residual of at most 10^{-12} is achieved. We have also performed numerical results using a preconditioned GMRes which led to similar results and is hence not shown here.

The tests were performed sequentially on the TUHH HPC cluster[1] using an AMD Epyc 9354 processor with 32 cores and 3.8 GHz. The implementation of the tests is based on the H2Lib [15], an open source software library for hierarchical matrices written in C.

6.1 | Coupled Clustering

6.1.1 | Varying System Size

First, we compare the computational times needed for uncoupled and coupled clustering strategies. The computation times per degree of freedom are depicted in Figure 7.

We observe that the coupled clustering leads to faster computations of the backward solve and the multiplication for the computation of the (approximate) Schur complement (steps ② and ④).

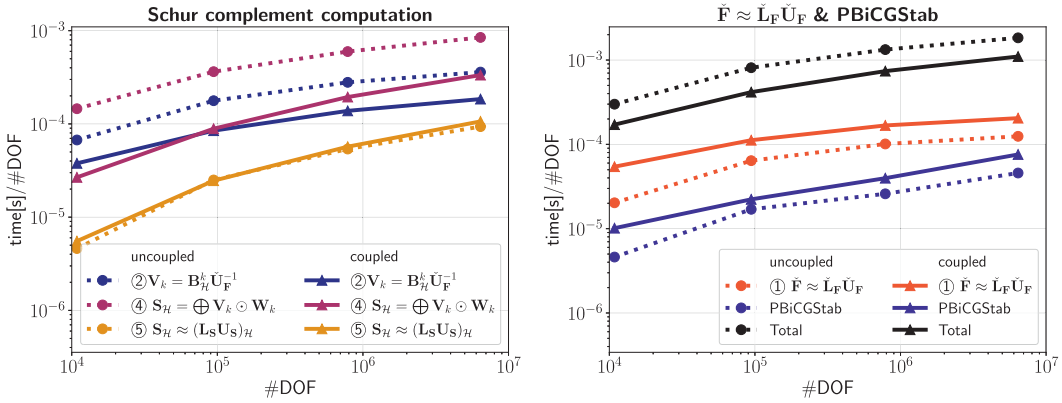

FIGURE 7 | Computation times per degree of freedom for the five steps of the preconditioner set-up and the BiCGStab iteration as well as the total time for the coupled clustering (solid lines) and the uncoupled clustering (dotted lines). [Colour figure can be viewed at wileyonlinelibrary.com]

TABLE 1 | Absolute computation times for the steps of the preconditioner set-up as well as the solve with the BiCGStab method (with iteration count) and the total time for the uncoupled clustering (top) and the coupled clustering (bottom).

#DOF	①	②	④	⑤	PBiCGStab (Iter)	Total
10,853	0.2 s	0.7 s	1.6 s	0.1 s	0.05 s (9)	3.2 s
94,285	6.0 s	16.7 s	34.3 s	2.4 s	1.6 s (13)	76.5 s
786,077	79.5 s	219.6 s	469.4 s	42.3 s	20.4 s (18)	1045.7 s
6,419,773	799.6 s	2308.3 s	5435.8 s	600.6 s	293.6 s (26)	11,775.6 s
#DOF	①	②	④	⑤	**PBiCGStab (Iter)**	**Total**
10,853	0.6 s	0.4 s	0.3 s	0.1 s	0.1 s (9)	1.9 s
94,285	10.6 s	8.0 s	8.4 s	2.3 s	2.1 s (13)	39.2 s
786,077	131.8 s	108.5 s	152.7 s	44.8 s	31.2 s (21)	580.9 s
6,419,773	1313.4 s	1185.6 s	2144.2 s	682.8 s	486.9 s (33)	7066.8 s

The same observation, although not shown here, holds for the forward solve (step ③). The computation of an $\mathcal{H}$-LU factorization of $\check{\mathbf{F}}$, on the other hand, is slower with the coupled clustering (due to the increased size of the interface cluster).

The absolute computation times, also shown in Table 1, indicate that coupled clustering reduces the total computational times to about 60% of the time needed for uncoupled clustering.

Similar observations as for the computation times can be made for the required memory and are depicted in Figure 8. This figure shows the memory required for the matrix $\mathbf{V}^1$ and the factors in the $\mathcal{H}$-LU factorizations $\check{\mathbf{L}}_\mathbf{F}\check{\mathbf{U}}_\mathbf{F} \approx \check{\mathbf{F}}$ and $\mathbf{L}_\mathbf{S}\mathbf{U}_\mathbf{S} \approx \mathbf{S}_\mathcal{H}$. The matrices $\mathbf{V}^k$ and $\mathbf{W}^k$, $k = 1, 2, 3$, all require a similar amount of memory, and the memory of the sparse matrices $\mathbf{B}^k$, $k = 1, 2, 3$, and $\check{\mathbf{F}}_\mathcal{H}$ can be neglected in comparison.

While the $\mathcal{H}$-LU factorization $\check{\mathbf{L}}_\mathbf{F}\check{\mathbf{U}}_\mathbf{F} \approx \check{\mathbf{F}}$ requires only slightly more memory with the coupled clustering, there are significant savings for the (intermediate) matrix $\mathbf{V}^1$ when using the coupled compared to the uncoupled clustering.

6.1.2 | Varying Truncation Accuracy

Next, we consider tests with varying truncation accuracies for the different steps of the preconditioner set-up. The motivation of this is that although a higher truncation accuracy results in slower computations this may result in a speed-up of the BiCGStab solve through an even lower number of iteration steps required to achieve the target (relative) residual norm. First, we fix the truncation accuracies to $\delta^\mathcal{H}_\text{mul} = \delta^\mathcal{H}_\text{Schur} = 10^{-1}$ and vary $\delta^\mathcal{H}_\text{vel}$.

As we can observe, varying $\delta^\mathcal{H}_\text{vel}$ does not affect steps ④ and ⑤ of the preconditioner set-up (as it can be expected). However, a higher truncation accuracy $\delta^\mathcal{H}_\text{vel}$ slows down all computations involving (parts of) the $\mathcal{H}$-LU factorization of $\check{\mathbf{F}}$. This effect is even stronger with the coupled clustering. Additionally, the time required for the BiCGStab solve does decrease only slightly with the uncoupled clustering and even increases with the coupled clustering.

Now, we consider fixed truncation accuracies $\delta^\mathcal{H}_\text{vel} = \delta^\mathcal{H}_\text{Schur} = 10^{-1}$ and a varying truncation accuracy $\delta^\mathcal{H}_\text{mul}$ for the steps ②–④ of the preconditioner set-up (cf. Section 3). The computation times per degree of freedom are presented in Figure 9.

We observe similar results as in Figure 10: As expected, the $\mathcal{H}$-LU factorization of $\check{\mathbf{F}}$ is not affected by changing the accuracy $\delta^\mathcal{H}_\text{mul}$, and the time required for the explicit computation of the (approximate) Schur complement increases with a decreasing $\delta^\mathcal{H}_\text{mul}$. This effect is (again) stronger with the coupled clustering.

6.2 | Coupled Clustering with Interface Decomposition

In this subsection we will show numerical experiments for the coupled clustering with interface decomposition introduced in Section 5.3. For this, we consider a varying system size and use the same truncation accuracy $\delta^{\mathcal{H}} = 10^{-1}$ for all $\mathcal{H}$-matrix operations. Figure 11 depicts the computation times per degree of freedom.

We observe that the computation of an (approximate) Schur complement is also faster with the coupled clustering with interface decomposition. Additionally, the computation time for an $\mathcal{H}$-LU factorization $\mathbf{L}_\mathbf{F}\mathbf{U}_\mathbf{F} \approx \check{\mathbf{F}}$ is similar for the two cluster strategies. However, Figure 11 suggests that the speed-up with the coupled clustering with interface decomposition is smaller than the speed-up with the coupled clustering observed in Figure 7. The absolute computation times in Table 2 indicate a reduction to 80%–90% of the computational time with uncoupled clustering compared to the reduction to 60% observed in Table 1 for the coupled clustering.

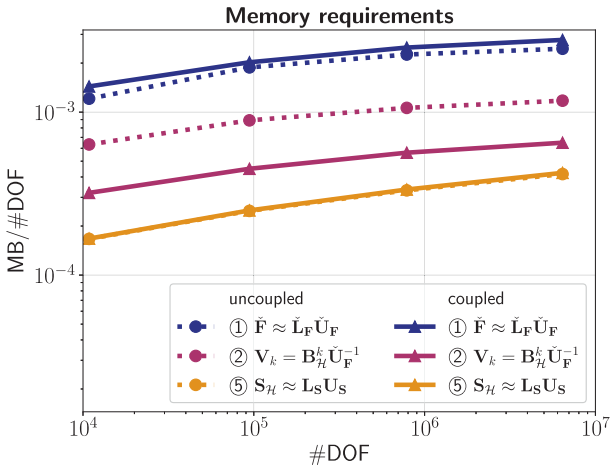

FIGURE 8 | Required memory per degree of freedom for the $\mathcal{H}$-LU factorizations $\check{\mathbf{L}}_\mathbf{F}\check{\mathbf{U}}_\mathbf{F} \approx \check{\mathbf{F}}$ and $\mathbf{L}_\mathbf{S}\mathbf{U}_\mathbf{S} \approx \mathbf{S}_\mathcal{H}$ as well as for $\mathbf{V}^1 = \mathbf{B}_\mathcal{H}^1\check{\mathbf{U}}_\mathbf{F}^{-1}$. The plot shows the memory required with the uncoupled clustering (dotted lines) and with the coupled clustering (solid lines). [Colour figure can be viewed at wileyonlinelibrary.com]

7 | Conclusions

The coupled clustering described in Section 5.2 is developed to yield a block structure with large zero blocks in the $\mathcal{H}$-matrix representations of the off-diagonal blocks $\mathbf{B}^1, \ldots, \mathbf{B}^d$ of the saddle point system (1). We were able to prove that this clustering strategy results in a cluster tree with a logarithmic depth bound in Theorem 1 in the presented finite element scenario. This property is important to obtain a computational complexity of $\mathcal{O}(n\log^\alpha n)$ for $\mathcal{H}$-matrix operations.

Although the coupled clustering approach results in a less favorable $\mathcal{H}$-matrix representation of the saddle point matrix block $\mathbf{F}$ (or rather $\check{\mathbf{F}}$) due to increased interface clusters, our numerical results suggest that, compared to the uncoupled clustering, the coupled clustering reduces the compuational time to 60% compared to uncoupled clusterin. However, the situation changes if we use higher truncation accuracies, that is, smaller values for $\delta_{\text{vel}}^{\mathcal{H}}$ or $\delta_{\text{mul}}^{\mathcal{H}}$. In this case, the coupled clustering looses its advantage, which is discussed in more detail in [14].

The coupled clustering with interface decomposition, on the other hand, achieves the intended improvement of the block structure of $\mathcal{H}$-matrix representation of $\check{\mathbf{F}}$. We observed similar computation times for the $\mathcal{H}$-LU factorization of $\check{\mathbf{F}}$ with the uncoupled clustering and the coupled clustering with interface decomposition. However, although we also observed a small improvement of the computation times for the explicit computation of an (approximate) Schur complement with the coupled clustering with interface decomposition, the observed total reduction leads to 80%–90% compared to uncoupled clustering and is hence inferior to the reduction to 60% we observed for the coupled clustering.

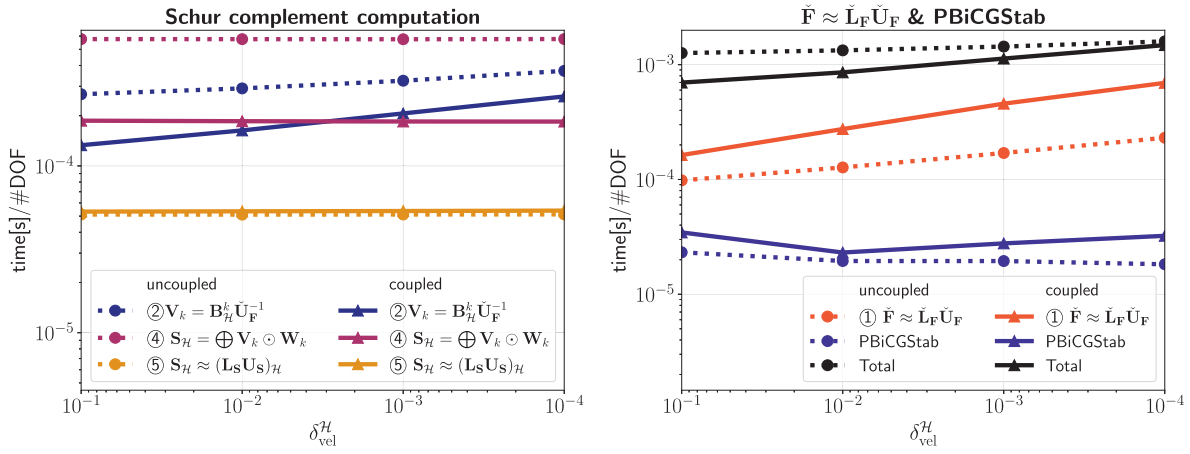

FIGURE 9 | Computation times per degree of freedom for a varying truncation with the uncoupled clustering (dotted lines) and the coupled clustering (solid lines) and a varying truncation accuracy $\delta_{\text{mul}}^{\mathcal{H}}$. [Colour figure can be viewed at wileyonlinelibrary.com]

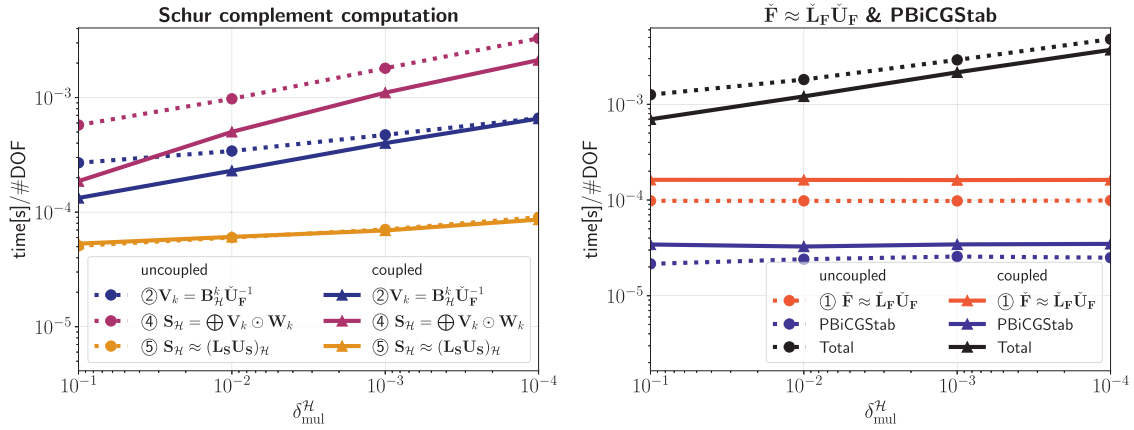

FIGURE 10 | Computation times per degree of freedom for a varying truncation accuracy [$\delta^{\mathcal{H}}_{\text{vel}}$] for the uncoupled clustering (dotted lines) and the coupled clustering (solid lines). [Colour figure can be viewed at wileyonlinelibrary.com]

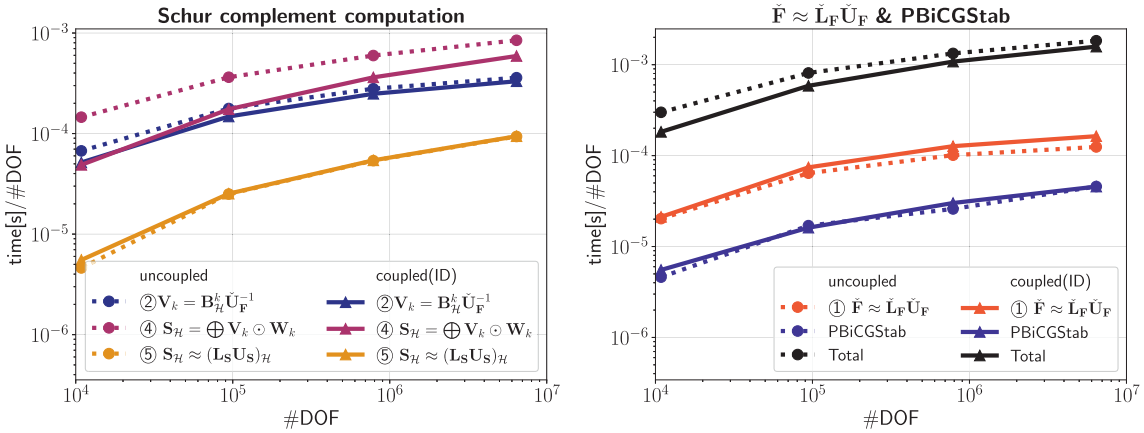

FIGURE 11 | Computation times per degree of freedom for the preconditioner set-up, the solve with the BiCGStab method, and the total time for the uncoupled clustering (dotted lines) and the coupled clustering with interface decomposition (coupled(ID)) (solid lines). [Colour figure can be viewed at wileyonlinelibrary.com]

TABLE 2 | Absolute computation times for the steps of the preconditioner set-up as well as the solve with the BiCGStab method (with iteration count) and the total time for the uncoupled clustering (top) and the coupled clustering with interface decomposition (bottom).

#DOF	①	②	④	⑤	PBiCGStab (Iter)	Total
10,853	0.2 s	0.7 s	1.6 s	0.1 s	0.05 s (9)	3.2 s
94,285	6.0 s	16.7 s	34.3 s	2.4 s	1.6 s (13)	76.5 s
786,077	79.5 s	219.6 s	469.4 s	42.3 s	20.4 s (18)	1045.7 s
6,419,773	799.6 s	2308.3 s	5435.8 s	600.6 s	293.6 s (26)	11,775.6 s
#DOF	①	②	④	⑤	**PBiCGStab (Iter)**	**Total**
10,853	0.2 s	0.6 s	0.5 s	0.1 s	0.06 s (8)	2.0 s
94,285	7.0 s	13.9 s	16.5 s	2.4 s	1.5 s (13)	55.2 s
786,077	99.8 s	195.4 s	284.8 s	42.6 s	23.7 s (19)	849.9 s
6,419,773	1049.9 s	2127.5 s	3793.7 s	603.9 s	292.9 s (24)	10,141.6 s

Author Contributions

Jonas Grams: writing – original draft, software, investigation, visualization, formal analysis. **Sabine Le Borne:** conceptualization, writing – review and editing, project administration, supervision.

Data Availability Statement

Data sharing not applicable to this article as no datasets were generated or analysed during the current study.

Endnotes

[1]https://www.tuhh.de/rzt/services/hpc/hardware (Last visited on March 18th, 2025).

Acknowledgments

Open Access funding enabled and organized by Projekt DEAL.

References

1. H. C. Elman, D. J. Silvester, and A. J. Wathen, "Finite Elements and Fast Iterative Solvers: With Applications in Incompressible Fluid Dynamics," in *Second Numerical Mathematics and Scientific Computation* (Oxford University Press, 2014), xiv+479, https://doi.org/10.1093/acprof:oso/9780199678792.001.0001.

2. M. Benzi, G. H. Golub, and J. Liesen, "Numerical Solution of Saddle Point Problems," *Acta Numerica* 14 (2005): 1–137, https://doi.org/10.1017/S0962492904000212.

3. S. Börm and S. Le Borne, "$\mathscr{H}$-LU Factorization in Preconditioners for Augmented Lagrangian and Grad-Div Stabilized Saddle Point Systems," *International Journal for Numerical Methods in Fluids* 68, no. 1 (2012): 83–98, https://doi.org/10.1002/fld.2495.

4. S. Le Borne, "Hierarchical Matrix Preconditioners for the Oseen Equations," *Computing and Visualization in Science* 11, no. 3 (2008): 147–157, https://doi.org/10.1007/s00791-007-0065-x.

5. L. Grasedyck and W. Hackbusch, "Construction and Arithmetics of $\mathscr{H}$-Matrices," *Computing. Archives for Scientific Computing* 70, no. 4 (2003): 295–334, https://doi.org/10.1007/s00607-003-0019-1.

6. W. Hackbusch, "A Sparse Matrix Arithmetic Based on $\mathscr{H}$-Matrices. I. Introduction to $\mathscr{H}$-Matrices," *Computing. Archives for Scientific Computing* 62, no. 2 (1999): 89–108, https://doi.org/10.1007/s006070050015.

7. M. Bebendorf, "Why Finite Element Discretizations Can be Factored by Triangular Hierarchical Matrices," *SIAM Journal on Numerical Analysis* 45, no. 4 (2007): 1472–1494, https://doi.org/10.1137/060669747.

8. M. Bebendorf and W. Hackbusch, "Existence of $\mathscr{H}$-Matrix Approximants to the Inverse FE -Matrix of Elliptic Operators With L^∞-Coefficients," *Numerische Mathematik* 95, no. 1 (2003): 1–28, https://doi.org/10.1007/s00211-002-0445-6.

9. M. Faustmann, J. M. Melenk, and D. Praetorius, "$\mathscr{H}$ -Matrix Approximability of the Inverses of FEM Matrices," *Numerische Mathematik* 131, no. 4 (2015): 615–642, https://doi.org/10.1007/s00211-015-0706-9.

10. L. Grasedyck, R. Kriemann, and S. Le Borne, "Domain Decomposition Based $\mathscr{H}$-LU Preconditioning," *Numerische Mathematik* 112, no. 4 (2009): 565–600, https://doi.org/10.1007/s00211-009-0218-6.

11. J. D. Grams and S. Le Borne, "Coupled Clustering Strategies for Hierarchical Matrix Preconditioners in Saddle Point Problems," *Proceedings in Applied Mathematics and Mechanics* 23 (2023): e202300077.

12. V. John, *Finite Element Methods for Incompressible Flow Problems, Vol. 51*. Springer Series in Computational Mathematics (Springer, 2016), xiii+812, https://doi.org/10.1007/978-3-319-45750-5.

13. W. Hackbusch, *Hierarchical Matrices: Algorithms and Analysis, Vol. 49*. Springer Series in Computational Mathematics (Springer, 2015), xxv+511, https://doi.org/10.1007/978-3-662-47324-5.

14. J. D. Grams, "Construction of Hierarchical Matrices for the Preconditioning of the Three-Dimensional Navier-Stokes Equations," PhD thesis, Technische Universität Hamburg, 2025, https://doi.org/10.15480/882.14508.

15. S. Börm, "H2Lib," https://www.h2lib.org. Version 3.0 (visited on 18 March, 2025).