Robust and scalable adaptive BDDC preconditioners for virtual element discretizations of elliptic partial differential equations in mixed form

Franco Dassi[*,a,1], Stefano Zampini[b,1], S. Scacchi[c,1]

[a]*Dipartimento di Matematica e Applicazioni, Università degli Studi di Milano Bicocca, Via Roberto Cozzi 55 - 20125 Milano, Italy*
[b]*Extreme Computing Research Center, King Abdullah University of Science and Technology, Thuwal 23955, Saudi Arabia*
[c]*Dipartimento di Matematica, Università degli Studi di Milano, Via Saldini 50, 20133 Milano, Italy*

Abstract

The Virtual Element Method (VEM) is a recent numerical technology for the solution of partial differential equations on computational grids constituted by polygonal or polyhedral elements of very general shape. The aim of this work is to develop effective linear solvers for a general order VEM approximation designed to approximate three-dimensional scalar elliptic equations in mixed form. The proposed Balancing Domain Decomposition by Constraints (BDDC) preconditioner allows to use conjugate gradient iterations, albeit the algebraic linear systems arising from the discretization of the problem are indefinite, ill-conditioned, and of saddle point nature. The condition number of the resulting positive definite preconditioned system is adaptively controlled by means of deluxe scaling operators and suitable local generalized eigenvalue problems for the selection of optimal primal constraints. Numerical results confirm the theoretical estimates and the reliability of the adaptive procedure, with the experimental condition numbers always very close to the prescribed adaptive tolerance parameter. The scalability and quasi-optimality of the preconditioner are demonstrated, and the performances of the proposed solver are compared with state-of-the-art parallel direct solvers and block preconditioning techniques in a distributed memory setting.

Key words: Virtual Element Method, Saddle-point linear systems, Parallel computing, Domain Decomposition, BDDC method.

AMS subject classifications: 65F08, 65N30, 65N35, 65N55

1. Introduction

Several physical phenomena in computational mechanics and fluid-dynamics are described by systems of Partial Differential Equations (PDE) that, upon time and space discretization, yield the solution of saddle point algebraic linear systems, which are indefinite and ill-conditioned. Consequently, solving such linear systems with iterative methods (see e.g. [49, 63]) requires the development of robust and effective preconditioners, usually based on approximate block factorization, see [4, 32, 50, 17, 40, 5].

In recent years, the interest of an increasing number of researchers has focused on the development of numerical methods for the approximation of PDEs on polygonal or polyhedral grids, see e.g. [14]. Among the different methodologies, the Virtual Element Method (VEM), introduced in the pioneering paper [10], represents a generalization of the Finite Element Method that can easily handle general polytopal meshes. So far, VEM has been analyzed for mixed formulation of elliptic problems [22, 13, 28], Stokes [15], parabolic equations [54], discrete fracture networks [16] and several further applications. Different variants of the VEM have been proposed and analysed: H(div) and H(curl)-conforming [12], serendipity [11] and non-conforming [6] VEM.

In the VEM literature only a few studies have focused on the conditioning of the stiffness matrix resulting from VEM discretizations (see [41, 26]) and on the construction and analysis of preconditioners for VEM approximations of PDEs (see [18, 3, 23, 24, 19]). In our previous works [28, 27], we have developed parallel block preconditioners for three-dimensional VEM approximations of several saddle point PDEs. However, while such block solvers are very effective in case of low order approximations, their performance deteriorates for high order discretizations. Several Domain Decomposition preconditioners have also been proposed for finite element discretizations of saddle point problems. In the context of finite element approximations of saddle point PDEs,

[*]Corresponding author
Email addresses: `franco.dassi@unimib.it` (Franco Dassi), `stefano.zampini@kaust.edu.sa` (Stefano Zampini), `simone.scacchi@unimi.it` (S. Scacchi)
[1]Member of INdAM-GNCS research group

overlapping preconditioners have been developed for mixed formulation of linear elasticity and Stokes equations [35, 36, 45, 33, 34] and for scalar elliptic equations in mixed form [42, 43], while non-overlapping preconditioners have been developed for Stokes equations [47], almost incompressible elasticity [48] and scalar elliptic equations in mixed form [61]. Within non-overlapping methods, the choices of scaling functions and primal constraints define the coarse approximation, which is crucial to obtain the quasi-optimality (sublinear dependence on the number of degrees of freedom) and scalability (independence of the number of subdomains) of the algorithm.

In this work, we will consider the Balancing Domain Decomposition by Constraints (BDDC) preconditioners [30] that belong to the non-overlapping class. Recently, a considerable effort has been put into developing multilevel [53] and adaptive [25] methods for the selection of primal constraints for BDDC algorithms, in order to improve their effectiveness by adaptively constructing the coarse primal problem. In [61], an effective adaptive BDDC algorithm has been proposed for finite element discretizations of scalar elliptic equations in mixed form; in this work, we extend the solver proposed in [61] to three-dimensional VEM approximations of scalar elliptic equations in mixed form. Several parallel numerical tests show the optimality, scalability and robustness of the adaptive BDDC preconditioner with respect to the VEM approximation order. The efficiency of the BDDC preconditioner is compared with that of block preconditioners as well as with the parallel direct solvers MUMPS [1, 2] and Pardiso [29, 55, 37].

The rest of the paper is organized as follows. In Section 2 we introduce the variational formulation of the saddle point problem considered. Section 3 contains a brief description of the three-dimensional VEM discretizations employed. Section 4, introduces the adaptive BDDC preconditioner, while numerical experiments are reported in Section 5. Conclusions are drawn in Section 6.

2. Model problem: elliptic equation in mixed form

The object of this work is the solution of the variational problem arising from the mixed formulation of a scalar elliptic equation in three dimension. Let Ω be a bounded Lipschitz domain in $\mathbb{R}^3$, whose boundary is denoted by $\partial\Omega$. We define the function spaces

$$\mathbf{V} := \{\mathbf{u} \in H(\mathrm{div}\,,\Omega) \;:\; \mathbf{u}\cdot\mathbf{n} = u_N \quad \text{on } \partial\Omega\},$$

and

$$Q := \left\{ q \in L^2(\Omega) \;:\; \int_\Omega q \;\, \mathrm{d}\Omega = 0 \right\},$$

where $H(\mathrm{div}\,,\Omega)$ is the space of vector-valued functions such that $\mathbf{u}$ and $\mathrm{div}\,(\mathbf{u})$ belong to $[L^2(\Omega)]^3$ and $L^2(\Omega)$, respectively, and u_N is the given Neumann datum.

We are interested in the solution of the following variational problem:

$$\begin{cases} \text{find } (\mathbf{u},p) \in \mathbf{V}\times Q : & & \\ \quad \boldsymbol{a}(\mathbf{u},\mathbf{v}) - \boldsymbol{b}(\mathbf{v},p) & = 0 & \forall \mathbf{v}\in\mathbf{V} \\ \quad \boldsymbol{b}(\mathbf{u},q) & = \displaystyle\int_\Omega f\,q \;\, \mathrm{d}\Omega & \forall q \in Q, \end{cases} \tag{1}$$

where

$$\begin{aligned} \boldsymbol{a}(\mathbf{u},\mathbf{v}) &:= \int_\Omega \nu\,\mathbf{u}\cdot\mathbf{v} \;\, \mathrm{d}\Omega, \\ \boldsymbol{b}(\mathbf{u},q) &:= \int_\Omega \mathrm{div}\,(\mathbf{u})\,q \;\, \mathrm{d}\Omega, \end{aligned} \tag{2}$$

with f a given function in $L^2(\Omega)$ satisfying the compatibility condition $\int_\Omega f \;\mathrm{d}\Omega = 0$, and ν a positive, piece-wise constant, scalar function in $L^\infty(\Omega)$. Symmetric positive definite diffusivity tensors can be considered as well.

Problem (1) arises in the context of multiphase incompressible flow through porous media, see e.g. [20]. Functions $\mathbf{u}$ and p are usually called *velocity* and *pressure*. We refer to [20] for the mathematical analysis of such problem. We note here that, in order to keep the presentation simple and concise, we only consider essential boundary conditions on the velocity variable. The techniques outlined in this paper easily adapt to the case of mixed boundary conditions.

3. Virtual element discretization

Let Ω_h be a polyhedral discretization of a domain $\Omega \subset \mathbb{R}^3$. In order to solve the problem defined in Equation (1) we follow a standard VEM approach; we first define local spaces on each polyhedron $P \in \Omega_h$ and then a global one by gluing together the local spaces. In what follows we give a brief description of the local discretization spaces. For further theoretical details and convergence properties, we refer the reader to [22]; for implementation details on the discretization technique we suggest [28].

3.1. Virtual element spaces

Given a polyhedron P, the discrete velocity space is defined as

$$\mathbf{V}_h^k(P) := \Big\{\mathbf{v}_h \in H(\mathrm{div}\,;P)\cap H(\mathbf{curl};P) \quad:\quad \mathbf{v}_h\cdot\mathbf{n}_F \in \mathbb{P}_k(F) \quad \forall F\in\partial P,$$
$$\mathrm{div}\,(\mathbf{v}_h)\in\mathbb{P}_{k-1}(P), \quad \mathbf{curl}(\mathbf{v}_h)\in[\mathbb{P}_{k-1}(P)]^3\Big\}. \tag{3}$$

The following degrees of freedom uniquely identify a function in $\mathbf{V}_h^k(P)$:

- *normal face moments*
$$\int_f (\mathbf{v}_h\cdot\mathbf{n})\,p_k\,\mathrm{d}f \quad \forall p_k\in\mathbb{P}_k(F) \quad\text{and}\quad \forall f\in\partial P\,,$$
- *internal gradient moments*
$$\int_P \mathbf{v}_h\cdot\nabla p_{k-1}\,\mathrm{d}P \quad \forall p_k\in\mathbb{P}_{k-1}(P)\,,$$
- *internal cross moments*
$$\int_P \mathbf{v}_h\cdot(\boldsymbol{x}\wedge\boldsymbol{p}_{k-1})\,\mathrm{d}P \quad \forall\boldsymbol{p}_{k-1}\in[\mathbb{P}_{k-1}(P)]^3\,.$$
where $\boldsymbol{x}=(x,\,y,\,z)^t$.

Inside each polyhedron, we approximate the pressure variable via a polynomial of degree $k-1$

$$Q_h(P) := \Big\{q_h\in L^2(P)\ :\ q_h\in\mathbb{P}_{k-1}(P)\Big\}\,,$$

that is uniquely determined by the following degrees of freedom

- *internal pressure moments*
$$\int_P q_h\,p_{k-1}\,\mathrm{d}P \quad \forall p_{k-1}\in\mathbb{P}_{k-1}(P)\,. \tag{4}$$

We then proceed in a standard way and we glue these local spaces together in order to build the global velocity and pressure spaces, that will be denoted in what follows as $\mathbf{V}_h^k(\Omega_h)$ and $Q_h(\Omega_h)$, respectively, i.e.

$$\mathbf{V}_h^k(\Omega_h) = \{v_h\in\mathbf{V}:\ v_{h|P}\in\mathbf{V}_h^k(P)\}$$

and

$$Q_h^k(\Omega_h) = \{q_h\in L^2(\Omega):\ q_{h|P}\in Q_h(P)\}.$$

Starting from the definition of the local spaces, we observe that the pressure is a piece-wise discontinuous polynomial of degree $k-1$, while the vector fields in $\mathbf{V}_h^k(\Omega_h)$ are only continuous in their normal component across each face of the polyhedral cells of the mesh. Moreover, the pressure is a polynomial that is computable from its degrees of freedom while the velocity field is not. In other words, the velocity is a *virtual function* that we do not know how, and do not want, to compute explicitly. Although $\mathbf{v}_h$ is virtual, there are some useful quantities that we can compute starting from its degrees of freedom. In particular, we can compute the polynomial representation of the virtual function on each mesh face and $\mathrm{div}\,(\mathbf{v}_h)$ in the interior of each polyhedron [28]; this is crucial for the well-posedness of our preconditioning strategy.

3.2. Discrete bilinear forms

Here we discuss the discretization of the bi-linear forms defined in Equation (2). The operator $\boldsymbol{b}(\mathbf{u}_h,q_h)$ can be computed on each polyhedron because both q_h and $\mathrm{div}\,(\mathbf{u}_h)$ are piece-wise polynomials. The bi-linear form $\boldsymbol{a}(\mathbf{u},\mathbf{v})$ is not computable since it requires knowledge of the velocity field inside each polyhedron, and we thus substitute it with a suitable approximation $\boldsymbol{a}_{h,P}$ that is computable from the degrees of freedom

$$\boldsymbol{a}(\mathbf{u},\mathbf{v}) \approx \sum_{P\in\Omega_h}\boldsymbol{a}_{h,P}(\mathbf{u}_h,\mathbf{v}_h)\,,$$

where

$$\boldsymbol{a}_{h,P}(\mathbf{v}_h,\boldsymbol{w}_h) := \underbrace{\int_P \nu(\boldsymbol{x})\,\mathbf{\Pi}_k^{0,P}\mathbf{u}_h\cdot\mathbf{\Pi}_k^{0,P}\mathbf{v}_h\,\mathrm{d}P}_{(i)} + \underbrace{s_P(\mathbf{u}_h-\mathbf{\Pi}_k^{0,P}\mathbf{u}_h,\mathbf{v}_h-\mathbf{\Pi}_k^{0,P}\mathbf{v}_h)}_{(ii)}\,, \tag{5}$$

In the above definition we distinguish two parts:

i) The **consistency part** that is based on the L^2-projection operator $\boldsymbol{\Pi}_k^{0,P} : \mathbf{V}_h^k(P) \to [\mathbb{P}_k(P)]^3$ defined from the following identity:

$$\int_P \boldsymbol{\Pi}_k^{0,P}\mathbf{v}_h \cdot \boldsymbol{p}_k \,\mathrm{d}P = \int_P \mathbf{v}_h \cdot \boldsymbol{p}_k \,\mathrm{d}P\,. \tag{6}$$

The left-hand side of such equation involves polynomials, while the right-hand side involves virtual functions; the computability of the latter expression follows by using the polynomial identity [28]

$$\boldsymbol{p}_k = \nabla p_{k+1} + \boldsymbol{x} \wedge \boldsymbol{p}_{k-1}\,,$$

and integration by parts.

ii) The **stabilization part** where s_P is any symmetric and positive definite bi-linear form which scales as the continuous bi-linear form $\boldsymbol{a}(\cdot,\cdot)$. In other words, we require that there exists two positive constants α_* and α^*, independent on the diameter of the polyhedron P, such that

$$\alpha_* \boldsymbol{a}(\mathbf{v}_h, \mathbf{v}_h) \le s_P(\mathbf{v}_h, \mathbf{v}_h) \le \alpha^* \boldsymbol{a}(\mathbf{v}_h, \mathbf{v}_h)\,, \quad \forall \mathbf{v}_h \in \mathbf{V}_h^k(P)\,.$$

While there are many choices in literature, see e.g., [9, 26, 41, 10, 8, 13], in this paper we use the so-called "`dofi-dofi`" stabilization [8, 13] and consider the Euclidean scalar product associated with the degrees of freedom of $\mathbf{V}_h^k(P)$ multiplied by the volume of P and the value of $\nu(\boldsymbol{x})$ at its barycenter

$$s_P(\mathbf{u}_h,\, \mathbf{v}_h) = \tau\, \nu(\boldsymbol{x}_P)\, |P| \sum_{i=1}^{\#dof_P} \mathtt{dof}_i(\mathbf{u}_h)\, \mathtt{dof}_i(\mathbf{v}_h)\,, \tag{7}$$

where $\#dof_P$ is the number of degrees of freedom associated with $\mathbf{V}_h^k(P)$, $\mathtt{dof}_i : \mathbf{V}_h^k(P) \to \mathbb{R}$ the linear functional associating functions in $\mathbf{V}_h^k(P)$ with the value of their i-th degree of freedom and $\tau > 0$ is a suitable scalar parameter.

3.3. Discrete mixed problem

We can then define the discrete VEM problem as

$$\begin{cases} \text{find } (\mathbf{u}_h, p_h) \in \mathbf{V}_h^k(\Omega_h) \times Q_h(\Omega_h) : & & \\ \qquad \boldsymbol{a}_h(\mathbf{u}_h, \mathbf{v}_h) - \boldsymbol{b}(\mathbf{v}_h, p_h) & = 0 & \forall \mathbf{v} \in \mathbf{V}_h^k(\Omega_h) \\ \qquad \boldsymbol{b}(\mathbf{u}_h, q_h) & = \displaystyle\int_{\Omega_h} f\, q_h \;\; \mathrm{d}\Omega_h & \forall q_h \in Q_h(\Omega_h)\,, \end{cases} \tag{8}$$

or equivalently in matrix form

$$\begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u}_h \\ \mathbf{p}_h \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{f}_h \end{bmatrix}. \tag{9}$$

Note that the left-hand side of the above equation is singular, with a null-space spanned by the constant functions on the pressure. If we consider a polyhedral mesh Ω_h where all polyhedrons are uniformly star shaped with respect to a ball and where the edge/face diameters are comparable with respect to the polyhedron diameters, Problem (8) has a unique solution $(\mathbf{u}_h, p_h) \in \mathbf{V}_h^k(\Omega_h) \times Q_h(\Omega_h)$ and, assuming enough regularity on the solution, the following error estimates hold

$$\begin{aligned} ||\mathbf{u} - \mathbf{u}_h||_0 &\le C\, h^{k+1} ||\mathbf{u}||_{k+1}\,, \\ ||p - p_h||_0 &\le C\, h^k (||p||_k + ||\mathbf{u}||_k)\,, \\ ||\mathrm{div}(\mathbf{u} - \mathbf{u}_h)||_0 &\le C\, h^k ||f||_k\,, \end{aligned}$$

where C is a constant independent on the mesh size, $||\cdot||_0$ is the standard $L^2(\Omega)$ norm and $||\cdot||_k$ is the norm associated with $H^k(\Omega)$. We refer the reader to [54, 13] for the theoretical proof of such estimate and to [28] for numerical verification.

4. Balancing Domain Decomposition by Constraints solver

Balancing Domain Decomposition by Constraints methods [30] are powerful methods to construct preconditioners for discrete systems arising from the discretization of partial differential equations; for a variational description of the method, see e.g. [21]. The robustness of the methods extends to div-conforming [44] and curl-conforming [31, 59, 62] discretization spaces, as well as to IsoGeometric Analysis [46]. For a recent review, see [56]; for algorithmic details see [58, 60].

In this work, we will consider the extension of these methods to saddle point problems with discontinuous pressure spaces first presented in [38, 52], and later extended to unstructured finite element computations and arbitrary number of levels in [61]. In what follows, we will present a concise description of their theoretical aspects by focusing on the VEM discretization; for further algorithmic details, see [61]. For solvers designed for continuous discretizations of the pressure space, the interested reader is referred to [57] and the references therein.

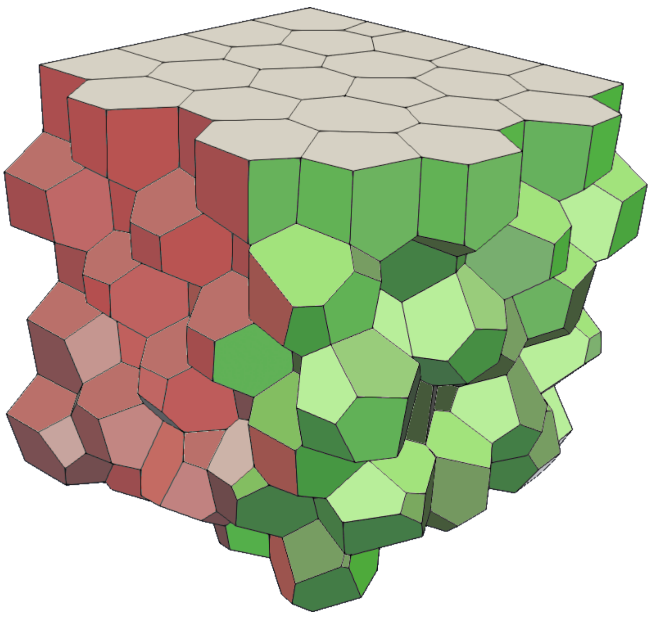


Figure 1: One subdomain with subdomain faces high-lighted in green and red; subdomain boundary face is in gray.

4.1. Non-overlapping domain decomposition

Within the framework of iterative sub-structuring [51] we decompose Ω_h into N non-overlapping connected subdomains

$$\overline{\Omega}_h = \bigcup_{i=1}^{N} \overline{\Omega}_i, \qquad \Gamma = \bigcup_{i\neq j} \partial\Omega_j \cap \partial\Omega_i,$$

where each Ω_i is a collection of polyhedrons, and where Γ denotes the interface among subdomains.

We then consider the splitting of $\mathbf{V}_h(\Omega_h)$ into a direct sum of two subspaces $\mathbf{V}_I \oplus \mathbf{V}_\Gamma$. $\mathbf{V}_I$ contains virtual functions identified by degrees of freedom that are associated to only one of the Ω_i, whereas the virtual functions in $\mathbf{V}_\Gamma$ are those associated with degrees of freedom shared by two subdomains, see Section 3.1. We will use the term subdomain face to identify the union of the polyhedral faces of the mesh that are shared by two given subdomains. See e.g. Figure 1 for an example.

Considering a change of basis of the pressure approximation space

$$Q_h = Q_I \oplus Q_0, \quad Q_0 = \prod_{i=1}^{N} \{q_h \in Q_h(\Omega_i) \mid q_h \text{ constant in } \Omega_i\},$$

and the computability of $\boldsymbol{b}(\mathbf{u}_h, q_h)$, we can use the divergence theorem and rewrite Equation (9) as

$$\begin{bmatrix} A_{II} & B_{II}^T & A_{I\Gamma} & 0 \\ B_{II} & 0 & B_{I\Gamma} & 0 \\ A_{I\Gamma}^T & B_{I\Gamma}^T & A_{\Gamma\Gamma} & B_{0\Gamma}^T \\ 0 & 0 & B_{0\Gamma} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u}_I \\ \mathbf{p}_I \\ \mathbf{u}_\Gamma \\ \mathbf{p}_0 \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{f}_\Gamma \\ 0 \\ \mathbf{f}_0 \end{bmatrix}. \tag{10}$$

Our preconditioning strategy considers a block factorization of the above linear system

$$\begin{bmatrix} I & -K_{II}^{-1}K_{I\Gamma} \\ 0 & I \end{bmatrix} \begin{bmatrix} K_{II}^{-1} & 0 \\ 0 & S^\dagger \end{bmatrix} \begin{bmatrix} I & 0 \\ -K_{I\Gamma}^T K_{II}^{-1} & I \end{bmatrix}, \tag{11}$$

where

$$K_{I\Gamma} = \begin{bmatrix} A_{I\Gamma} & 0 \\ B_{I\Gamma} & 0 \end{bmatrix}, \quad K_{II} = \begin{bmatrix} A_{II} & B_{II}^T \\ B_{II} & 0 \end{bmatrix}, \quad S_\Gamma = A_{\Gamma\Gamma} - \left[\begin{array}{c} A_{I\Gamma} \\ B_{I\Gamma} \end{array} \right]^T \left[\begin{array}{cc} A_{II} & B_{II}^T \\ B_{II} & 0 \end{array} \right] \left[\begin{array}{c} A_{I\Gamma} \\ B_{I\Gamma} \end{array} \right], \quad S = \begin{bmatrix} S_\Gamma & B_{0\Gamma}^T \\ B_{0\Gamma} & 0 \end{bmatrix}. \tag{12}$$

We note here that, recalling that A_{II} and

$$\left[\begin{array}{cc} A_{II} & A_{I\Gamma} \\ A_{I\Gamma}^T & A_{\Gamma\Gamma} \end{array} \right]$$

are positive definite and due to their saddle point structure, the matrices

$$\begin{bmatrix} A_{II} & B_{II}^T & A_{I\Gamma} \\ B_{II} & 0 & B_{I\Gamma} \\ A_{I\Gamma}^T & B_{I\Gamma}^T & A_{\Gamma\Gamma} \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} A_{II} & B_{II}^T \\ B_{II} & 0 \end{bmatrix},$$

have the same number of negative eigenvalues. Then, by Sylvester's law of inertia, the matrices

$$\begin{bmatrix} A_{II} & B_{II}^T & A_{I\Gamma} \\ B_{II} & 0 & B_{I\Gamma} \\ A_{I\Gamma}^T & B_{I\Gamma}^T & A_{\Gamma\Gamma} \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} A_{II} & B_{II}^T & 0 \\ B_{II} & 0 & 0 \\ 0 & 0 & S_\Gamma \end{bmatrix},$$

have the same number of positive and negative eigenvalues, from which it follows that the velocity Schur complement S_Γ has only positive eigenvalues, thus it is positive definite. The operator K_{II}^{-1} (which represents the solution of decoupled subdomain problems) is well defined since the space Q_I does not contain any subdomain constant function. On the other hand, the saddle point Schur complement S is singular, with a kernel spanned by the representation of the pressure constant function in Q_0. For the definition of its pseudo-inverse, we require that the application of $S^\dagger$ will have a null component in such a kernel.

Within the preconditioner, the pseudo-inverse of the saddle point Schur complement $S^\dagger$ is replaced by the action of a suitable preconditioner, resulting in the preconditioner

$$M^{-1} = \begin{bmatrix} I & -K_{II}^{-1}K_{I\Gamma} \\ 0 & I \end{bmatrix} \begin{bmatrix} K_{II}^{-1} & 0 \\ 0 & M_\Gamma^{-1} \end{bmatrix} \begin{bmatrix} I & 0 \\ -K_{I\Gamma}^T K_{II}^{-1} & I \end{bmatrix}. \tag{13}$$

Since subdomain interior problems are solved exactly, the spectrum of the global operator preconditioned by M^{-1} and the spectrum of $M_\Gamma^{-1}S$ are the same.

4.2. BDDC methods for saddle point Schur complement

BDDC methods construct preconditioners for the interface Schur complement operator S by relaxing continuity requirements of the operator; specifically, these methods are completely characterized by (cf. [39]):

- the definition of a partially assembled space $\widetilde{\mathbf{V}}_\Gamma$, direct sum of a *primal*, continuous, space $\mathbf{V}_\Pi$ and a *dual*, discontinuous space $\mathbf{V}_\Delta$, such that $\mathbf{V}_\Gamma \subset \widetilde{\mathbf{V}}_\Gamma = \mathbf{V}_\Delta \oplus \mathbf{V}_\Pi$
- a *scaling* operator $D : \mathbf{V}_\Gamma \to \widetilde{\mathbf{V}}_\Gamma$ such that $D'D = I$; here D' denotes the adjoint of D, and it will be referred to as *averaging* operator.

In this work we use the so-called *deluxe* variant of the scaling operator, which has proven robust for vector field problems in $H(\mathbf{curl})$ [31, 59, 62], $H(\text{div})$ [44], as well as in the context of IsoGeometric Analysis [46].

A crucial observation for the construction of our preconditioner is that the saddle point problem S is symmetric positive definite on $\{\mathbf{V}_\Gamma \cap \ker(B_{0\Gamma})\} \times Q_0$. Following [52], we then require that the virtual functions in the dual space satisfy the so-called *no-net-flux* condition

$$\int_{\partial\Omega_i} \mathbf{v}_\Delta \cdot \mathbf{n} = 0, \quad \forall\, \mathbf{v}_\Delta \in \mathbf{V}_\Delta. \tag{14}$$

Since we have assumed that each Ω_i is connected, in our algorithm we will consider one constraint for each subdomain face F

$$\int_F \mathbf{v}_\Delta \cdot \mathbf{n} = 0.$$

The quadrature weights associated with the above constraint can be computed by considering the linear functional $\boldsymbol{b}(\cdot, q_0)$ restricted to those virtual functions in $\mathbf{V}_h$ represented by nonzero degrees of freedom on F only, and where $q_0 \in Q_0$ is defined as $q_0 = 1$ in Ω_i and zero otherwise. In turn, the virtual functions in the primal space will be characterized by the continuity of

$$\int_F \mathbf{v}_\Pi \cdot \mathbf{n}, \quad \forall\, \mathbf{v}_\Pi \in \mathbf{V}_\Pi,\ \forall F \subset \Gamma. \tag{15}$$

for each subdomain face F.

Using the divergence theorem, it is easy to see that the VEM problem on $\{\mathbf{V}_I \oplus \widetilde{\mathbf{V}}_\Gamma\} \times \{Q_I \oplus Q_0\}$ can be recast as

$$\begin{bmatrix} A_{II} & B_{II}^T & A_{I\Delta} & A_{I\Pi} & 0 \\ B_{II} & 0 & B_{I\Delta} & B_{I\Pi} & 0 \\ A_{I\Delta}^T & B_{I\Delta}^T & A_{\Delta\Delta} & A_{\Delta\Pi} & 0 \\ A_{I\Pi}^T & B_{I\Pi}^T & A_{\Delta\Pi}^T & A_{\Pi\Pi} & B_{0\Pi}^T \\ 0 & 0 & 0 & B_{0\Pi} & 0 \end{bmatrix}. \tag{16}$$

Eliminating the degrees of freedom of $\mathbf{V}_I \times Q_I$ we can define a partially assembled Schur complement on $\widetilde{\mathbf{V}}_\Gamma \times Q_0$

$$\begin{bmatrix} S_{\Delta\Delta} & S_{\Delta\Pi} & 0 \\ S_{\Delta\Pi}^T & S_{\Pi\Pi} & B_{0\Pi}^T \\ 0 & B_{0\Pi} & 0 \end{bmatrix}, \quad \widetilde{S}_\Gamma = \begin{bmatrix} S_{\Delta\Delta} & S_{\Delta\Pi} \\ S_{\Delta\Pi}^T & S_{\Pi\Pi} \end{bmatrix}$$

and define the BDDC preconditioner as

$$M_\Gamma^{-1} = D' \widetilde{S}^\dagger D. \tag{17}$$

By using block factorization arguments, the action of $\widetilde{S}^\dagger$ can be implemented as additive combination of uncoupled subdomain solvers defined on $\{\mathbf{V}_I \oplus \mathbf{V}_\Delta\} \times Q_I$ and the solution of a coarse saddle point problem on $\mathbf{V}_\Pi \times Q_0$ which inherits the kernel from S.

It is easy to see that the preconditioned operator $M_\Gamma^{-1} S$ is positive definite on $\{\mathbf{V}_\Gamma \cap \ker(B_{0\Gamma})\} \times Q_0$ by requiring that the average operator preserves fluxes (15) of the virtual functions in $\widetilde{\mathbf{V}}_\Gamma$ [61, Lemma 3.5]. We can then solve the VEM linear system (9) by splitting the velocity solution as $\mathbf{u}_h = \mathbf{u}_h^* + \overline{\mathbf{u}}_h$, where $\mathbf{u}_h^*$ is defined by its components in $\mathbf{V}_\Gamma$ and $\mathbf{V}_I$ as

$$\begin{bmatrix} \mathbf{u}_{h,\Gamma}^* \\ * \end{bmatrix} = M_\Gamma^{-1} \begin{bmatrix} 0 \\ \mathbf{f}_{h,0} \end{bmatrix}, \quad (\mathbf{u}_h^*)_I = -K_{II}^{-1} K_{I\Gamma} \mathbf{u}_{h,\Gamma}^*,$$

and use conjugate gradient iterations to solve the deflated system

$$\begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \overline{\mathbf{u}}_h \\ \mathbf{p}_h \end{bmatrix} = \begin{bmatrix} -A\mathbf{u}_h^* \\ \mathbf{f}_h - B\mathbf{u}_h^* \end{bmatrix}. \tag{18}$$

Convergence rate estimates for BDDC preconditioned conjugate gradient methods are completely characterized by the norm of the average operator $||D'||_{\widetilde{S}_\Gamma}$; for the finite element method, the latter is usually bounded from above by $C(1 + \log(H/h)^2$ provided the proper primal space has been designed; here, C is a constant independent of the characteristic mesh size h, the diameter of the subdomains H, and the number of subdomains N. Proving such result for the virtual element method requires two-grid estimates and, more importantly, a face lemma, see e.g. [61, Section 4]; we do not provide such theoretical bound in this work. On the other hand, unravelling the expression for $||D'||_{\widetilde{S}_\Gamma}$, we can devise generalized eigenvalue problems, defined on each subdomain face F, to enrich the minimal primal space $\mathbf{V}_\Pi$ characterized by Eq. (15). In particular, given any $\nu_{tol} \in [1, \infty)$, we can algebraically construct an enriched primal space such that condition number of the preconditioned system will be bounded from above by ν_{tol} times a constant independent on h, H and N.

Specifically, given a subdomain face F shared by subdomains i and j, and the two subdomain Schur complement matrices $S^{(i)}$ and $S^{(j)}$ associated to the subdomain interfaces Γ_i and Γ_j, we solve the following generalized eigenvalue problem (separately on each subdomain face)

$$\widetilde{S}^{(i)}_{F_\Delta F_\Delta} : \widetilde{S}^{(j)}_{F_\Delta F_\Delta} \psi = \nu \; S^{(i)}_{F_\Delta F_\Delta} : S^{(j)}_{F_\Delta F_\Delta} \psi.$$

where $A : B = (A^{-1} + B^{-1})^{-1}$. Here F_Δ denotes the set of dual degrees of freedom on the face and

$$\widetilde{S}^{(i)}_{F_\Delta F_\Delta} = S^{(i)}_{F_\Delta F_\Delta} - S^{(i)T}_{F' F_\Delta} S^{(i)-1}_{F' F'} S^{(i)}_{F' F_\Delta},$$

with $F' = \Gamma_i \setminus F_\Delta$. The elements of the primal space are obtained as $S^{(i)}_{F_\Delta F_\Delta} : S^{(j)}_{F_\Delta F_\Delta} \Psi$, where Ψ is the matrix formed column-wise by those eigenvectors associated with eigenvalues smaller than a fixed tolerance $1/\nu_{tol}$. We note here that the choice of deluxe scaling is crucial for the construction of efficient primal spaces: for a comparison of primal spaces obtained with deluxe and diagonal scaling techniques in the context of a mixed discretization, see [61].

To summarize, we can state the following theorem; for technical details and theoretical proofs, see [61, Sections 3-4] and the references therein.

Theorem 1. *Let the dual space satisfy the no-net-flux condition given in* (14) *and let the average operator preserve subdomain normal fluxes as in* (15). *Then, $M_\Gamma^{-1} S$ is symmetric positive definite on the subspace $\{\mathbf{V}_\Gamma \cap \ker(B_{0\Gamma})\} \times Q_0$; the minimum eigenvalue is* 1 *and the maximum eigenvalue is bounded from above by $||D'||_{\widetilde{S}_\Gamma}$, with the norm taken on $\mathbf{V}_\Delta$. In addition, we can algebraically construct a primal space $\mathbf{V}_\Pi$ such that*

$$\kappa(M_\Gamma^{-1} S) \leq C \nu_{tol}, \; \forall \nu_{tol} \in [1, \infty),$$

where C is independent of N, h, and H.

5. Numerical results

In this Section we report numerical results to validate the adapted BDDC algorithm for solving the model problem (1) using the VEM discretization described in Section 2. We will consider three types of polyhedral meshes: hexahedral (**Cube**), octahedral (**Octa**) or Voronoi (**CVT**), see Figure 2 for an example. In what follows, we will denote by k the degree of approximation of the VEM discretization. Except otherwise stated, in all numerical tests we choose $\tau = 1$ as stabilitazion parameter in the bilinear form $s_P(\cdot,\cdot)$ defined in (7).

Our parallel solver is based on the Portable and Extensible Toolkit for Scientific computing (PETSc) library from Argonne National Laboratory [7], which is built on top of the MPI standard. PETSc offers advanced data structures and routines for the parallel solution of PDEs, from basic vector and matrix operations to more complex linear and nonlinear equation solvers. In our code, each subdomain is assigned to a different MPI process. For details on the implementation of the BDDC method in PETSc, see [58, 61].

The BDDC algorithms are employed as preconditioners for the deflated system given in (18), which is solved by the conjugate gradient (CG) method, using as a stopping criterion a 10^{-8} reduction of the l^2 norm of the relative residual. We will use $\nu_{tol} = \infty$ to denote the BDDC algorithm with minimal primal space consisting only of functions with continous normal fluxes across the subdomain interface.

All the numerical tests presented in the following have been performed on the Linux cluster INDACO (www.indaco.unimi.it) of the University of Milan, constituted by 16 nodes, each carrying 2 INTEL XEON E5-2683 V4 processors at 2.1 GHz, with 16 cores each. In the final section, the BDDC algorithms are compared against our previous block-diagonal preconditioner Block-Schur [28], and the parallel direct solvers MUMPS [1, 2] and Pardiso [29, 55, 37].

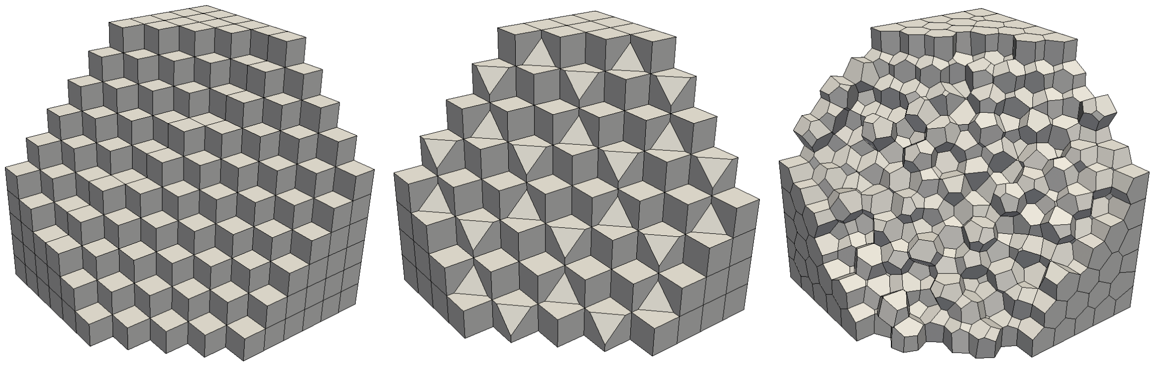


Figure 2: Sections of hexahedral (**Cube**, left), octahedral (**Octa**, middle) and Voronoi (**CVT**, right) meshes of the unit cube.

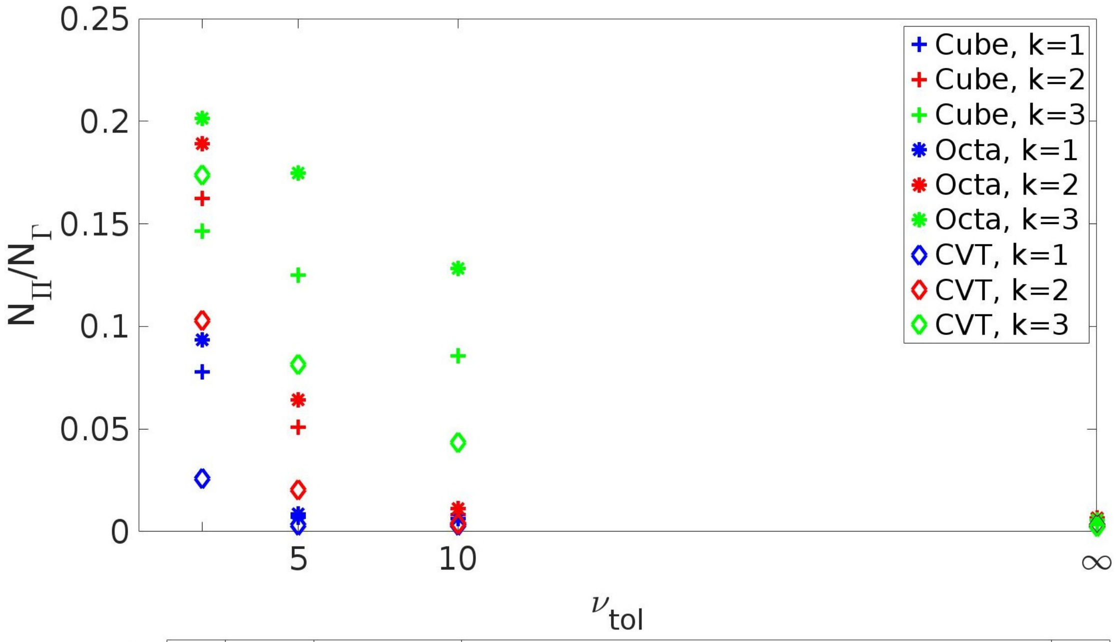
The main computational costs associated with the adaptive procedure consist in the explicit computation of the subdomain Schur complement matrices and in their explicit inversion. Local Schur complement matrices, which are also needed to construct the operators involved in the deluxe scaling procedure, are obtained as a by-product of the factorization of the subdomain problems; their explicit computation thus adds very little on the computational requirements of the overall algorithm. The complexity of the inversion task is independent on the type of discretization used, and it only depends cubically on the size of these dense local matrices. These Cholesky based inversions are the backbone of many numerical workflows, and high-performant implementations are available for barely all computing architectures. The solution of the eigenvalue problems, with dense left- and right-hand sides, also posses a cubical complexity, but it usually requires a fraction of the time needed by the subdomain inversion since the sizes of the submatrices involved are smaller. For additional details, including computational timings, see [58]. Besides these considerations, the analysis of the computational impact of using the adaptive procedure within the overall BDDC algorithm can only be assessed in the context of a real application. In particular, how the choice of ν_{tol} will affect the approximation properties of the primal space will strongly depend on the mesh, on its decomposition, and on the distribution of the material coefficients; moreover, the effect of choosing a tighter value (i.e. close to 1) on computational timings will depend on how many times the preconditioner can be reused between successive applications. This kind of analysis is outside the scope of this study.

5.1. Test 1: varying the adaptive tolerance

With the first set of numerical results we report about the reliability of the adaptive procedure for the selection of primal constraints. We will consider $\nu_{tol} = 2, 5, 10, \infty$ as tolerances and test the algorithm against different meshes with different number of elements and approximation degree, as reported in Table 1. Figure 3 summarizes the results.

	Cube		**Octa**		**CVT**	
k	nel	dofs	nel	dofs	nel	dofs
1	32768	435200	30375	453600	16000	388042
2	13824	508032	9000	361200	8000	465720
3	8000	612000	4608	378880	4000	445950

Table 1: Details of the meshes used in Section 5.1 and 5.2.



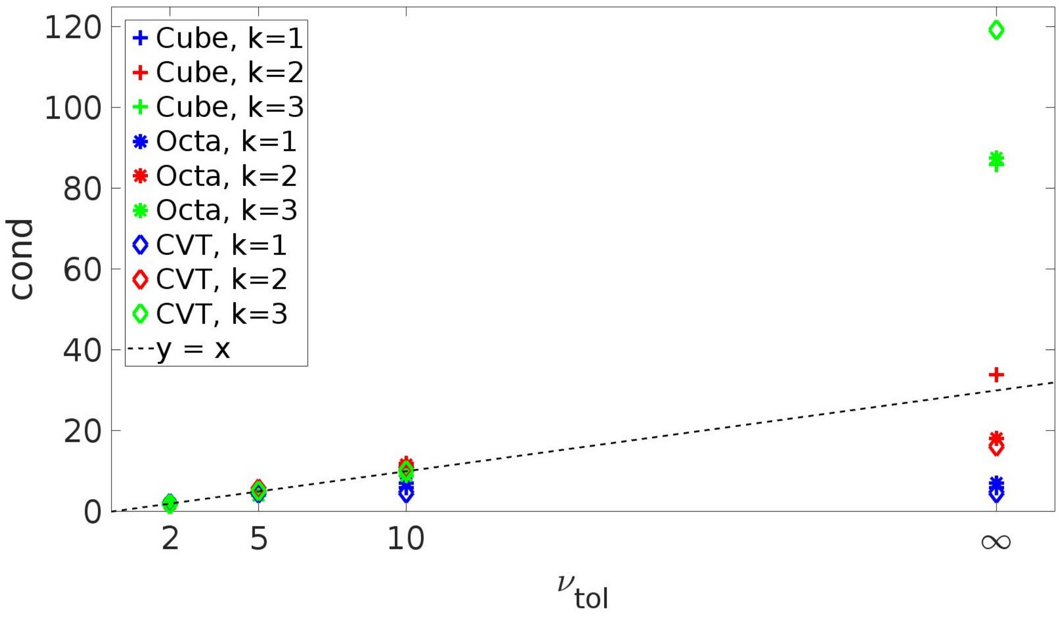


Figure 3: Test 1. Ratio (top panel) between the number of primal constraints (N_Π) and the total number of interface dofs (N_Γ) and spectral condition number of the preconditioned system (cond, bottom panel) as a function of the tolerance ν_{tol} in the adaptive BDDC algorithm. The number of processors is kept fixed at 32.

We first observe that the adaptive procedure works as expected, because the spectral condition number (cond) of the preconditioned system is always close to the prescribed tolerance ν_{tol}. The number of selected primal constraints N_Π increases as ν_{tol} is decreased, with a corresponding decrease in the number of CG iterations (not shown) and condition number. Increasing the VEM degree yields a strong increase of primal constraints, even by a factor of ten in some cases, but both CG iterations and condition number remain bounded. These considerations hold for the three types of polyhedral meshes considered (**Cube**, **Octa** and **CVT**), confirming the robustness of the algorithm with respect to the shape of the elements.

5.2. Test 2: strong scalability

We study the parallel performance of the solvers with a strong scaling test, by increasing the number of processors from 4 to 128, while keeping fixed the global number of degrees of freedom. We recall that, denoting by p the number of processors, the *parallel speedup* S_p is defined as

$$S_p := \frac{\text{CPU time with 4 processes}}{\text{CPU time with } p \text{ processes}}.$$

For details on the meshes used in the tests, see Table 1.

Cube mesh with 32768 elements
k = 1, dofs = 435200

procs	S_p^{id}	T_{ass}	S_p	$\nu_{tol} = 2$				$\nu_{tol} = \infty$			
				N_Π	it	T_{sol}	S_p	N_Π	it	T_{sol}	S_p
4	-	30	-	104	10	198	-	8	17	83	-
8	2	15	2.0	400	11	70	2.8	26	21	34	2.4
16	4	9	3.3	860	11	48	4.1	62	22	21	3.9
32	8	4	7.5	1823	11	16	12.4	147	24	7	11.8
64	16	2	15.0	3479	11	6	33.0	336	24	3	27.7
128	32	1	30.0	6944	11	3	66.0	768	25	2	42.0

Cube mesh with 13824 elements
k = 2, dofs = 508032

procs	S_p^{id}	T_{ass}	S_p	$\nu_{tol} = 2$				$\nu_{tol} = \infty$			
				N_Π	it	T_{sol}	S_p	N_Π	it	T_{sol}	S_p
4	-	117	-	203	9	239	-	6	18	119	-
8	2	64	1.8	797	9	92	2.6	24	28	51	2.3
16	4	34	3.4	2071	10	66	3.6	63	31	30	4.0
32	8	17	6.9	4291	11	32	7.5	146	39	11	10.8
64	16	9	13.0	8095	10	11	21.7	337	44	6	19.8
128	32	5	23.4	15398	11	6	39.8	730	55	4	29.7

Cube mesh with 8000 elements
k = 3, dofs = 612000

procs	S_p^{id}	T_{ass}	S_p	$\nu_{tol} = 2$				$\nu_{tol} = \infty$			
				N_Π	it	T_{sol}	S_p	N_Π	it	T_{sol}	S_p
4	-	491	-	375	9	332	-	8	58	211	-
8	2	254	1.9	748	11	121	2.7	16	62	81	2.6
16	4	138	3.6	1873	11	82	4.0	43	84	42	5.0
32	8	66	7.4	4103	11	33	10.1	99	89	18	11.7
64	16	33	14.9	6848	12	11	30.2	192	85	7	30.1
128	32	18	27.3	19312	11	9	36.9	509	109	7	30.1

Table 2: Test 2. Strong scalability, **Cube** meshes. procs:=number of processors; S_p^{id}:=ideal speedup; T_{ass}:=assembling time in seconds; N_Π:=number of primal constraints; it:=CG iterations; T_{sol}:=solution time in seconds; S_p:=parallel speedup computed with respect to the 4 processors run.

The results on **Cube** meshes are reported in Table 2 and plotted in the first row of Figure 4. We first observe that the CPU times needed to assemble the stiffness matrix and right hand side (T_{ass}) are scalable, with very good speedup values close to the ideal ones. The same holds true for the other meshes (data not shown). The adaptive BDDC solver ($\nu_{tol} = 2$, continuous lines) results scalable, since CG iterations remain bounded and CPU times are reduced super-linearly as the number of processors is increased. This is expected when performing a strong scaling analysis of preconditioners based on exact local factorizations. In addition, we observe that the scalability of the adaptive BDDC solver is independent of the degree of the VEM approximation. The minimal primal space solver ($\nu_{tol} = \infty$, dashed lines) is also scalable for $k = 1$, but its parallel performance deteriorates when increasing the degree of approximation. We also remark that the minimal primal space BDDC is always faster in terms of CPU times considering together preconditioner assembling and solution phases. However, adaptive BDDC is faster in the solution phase, making it an attractive option when we have to solve several linear systems with the same matrix.

The results using **Octa** and **CVT** meshes are reported in the second and third row of Figure 4, respectively. Experiments confirm the scalability of the adaptive BDDC solver, both in terms of CG iterations and CPU times, irrespective of the degree of approximation. The minimal primal space solver exhibits a good scalability for $k = 1$, however for $k = 2$ and $k = 3$ the CG iterations slightly increase with the number of processors. In case

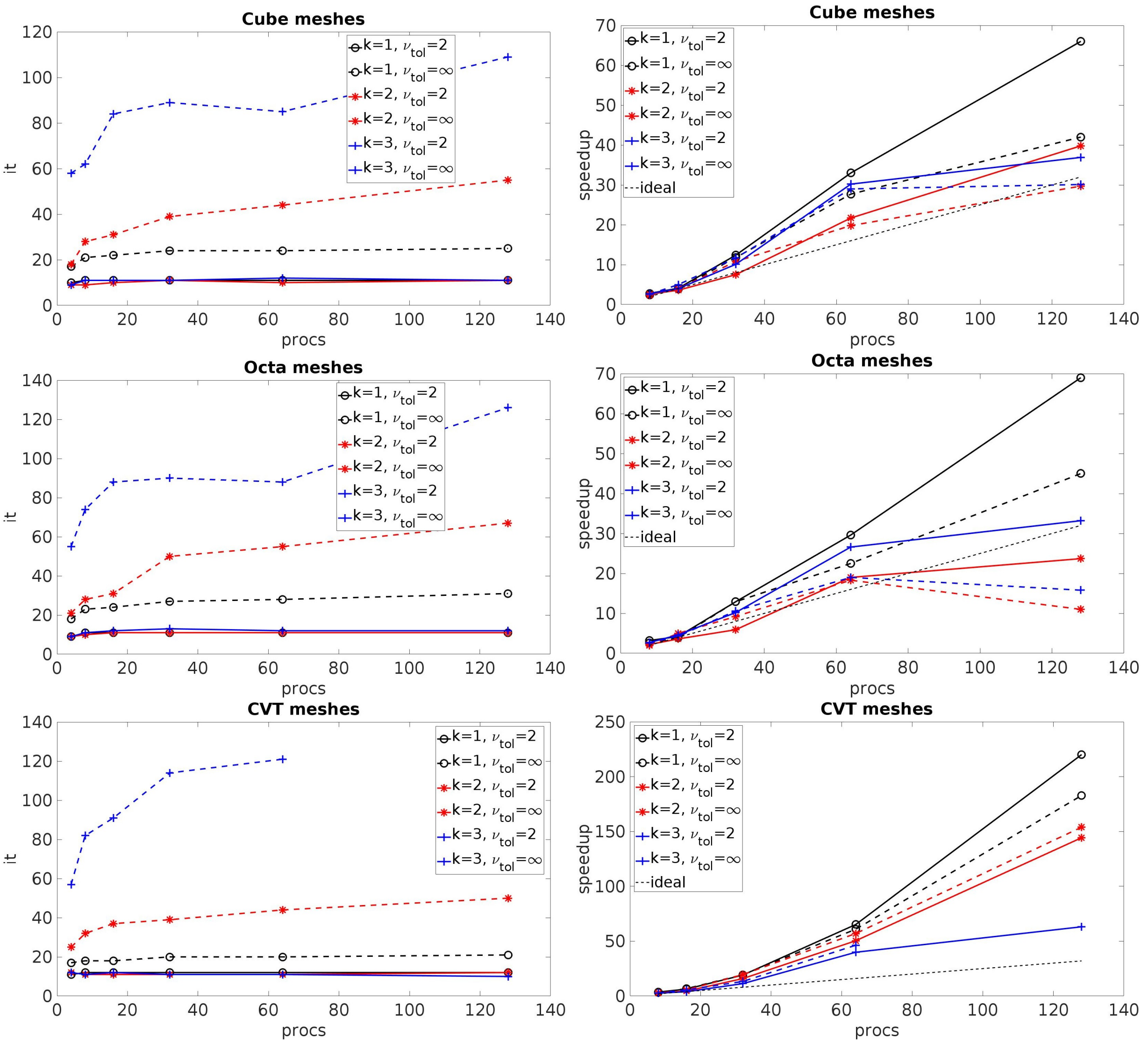


Figure 4: Test 2. Strong scalability. Plots of conjugate gradient iterations (left) and parallel speedup (right) with respect to the number of processors on **Cube** (first row), **Octa** (second row) and **CVT** (third row) meshes for adaptive ($\nu_{tol} = 2$, solid lines) and minimal primal space ($\nu_{tol} = \infty$, dashed lines) BDDC preconditioners.

of **CVT** mesh, $k = 3$, minimal primal space, 128 procs, the method did not converge, thus the corresponding data are missing.

5.3. Test 3: Optimality with respect to the mesh size

We investigate here the behavior of the solver when refining the mesh size, thus increasing the number of dofs. The number of processors is kept fixed at 32 and the orders of the VEM discretization considered are $k = 1, 2, 3$. Results are reported in Tables 3, 4, 5, for **Cube**, **Octa** and **CVT** meshes, respectively. The adaptive solver exhibits an optimal behavior irrespective of the polyhedral mesh considered, since the number of CG iterations and the condition number remain almost constant when increasing the number of dofs. The minimal primal space solver is optimal for $k = 1$, while in the higher order cases it shows an erratic behavior, with large iteration counts and condition number, especially for $k = 3$.

5.4. Test 4: Optimality with respect to the polynomial degree

We then numerically study the solver when the VEM approximation degree is increased, while the mesh is kept fixed. The poyhedral meshes considered are **Hexa** with 4096 elements, **Octa** with 576 elements and **CVT** with 1000 elements, with the degree k of the VEM discretization varying between 1 and 5. As a consequence, the number of dofs varies from 55552 to 904960 in case of **Hexa** meshes, from 9024 to 135744 in case of **Octa** meshes and from 23509 to 290563 in case of **CVT** meshes. The number of processors is kept fixed at 32.

The results reported in Table 6 show that the adaptive solver (with $\nu_{tol} = 2$) is robust with respect to k, since the number of CG iterations and the condition number remain almost constant when k increases. On the

Cube mesh, k = 1, procs = 32							
nel	dofs	$\nu_{tol} = 2$			$\nu_{tol} = \infty$		
		N_Π	it	cond	N_Π	it	cond
21952	292432	1460	11	1.88	124	23	6.42
32768	435200	1823	11	1.87	147	22	5.95
46656	618192	1951	11	1.75	149	21	5.60
64000	846400	2229	11	1.84	148	22	6.26
85184	1124816	2445	11	1.88	151	22	6.84

Cube mesh, k = 2, procs = 32							
nel	dofs	$\nu_{tol} = 2$			$\nu_{tol} = \infty$		
		N_Π	it	cond	N_Π	it	cond
4096	152064	2258	11	1.76	106	31	11.99
8000	295200	2694	11	1.77	99	30	11.22
13824	508032	4291	11	1.87	146	37	33.90
21952	804384	4577	11	1.76	124	31	11.17
32768	1198080	5981	10	1.69	147	33	15.67

Cube mesh, k = 3, procs = 32							
nel	dofs	$\nu_{tol} = 2$			$\nu_{tol} = \infty$		
		N_Π	it	cond	N_Π	it	cond
512	40320	2148	11	1.88	107	115	171.63
1728	133920	3178	12	2.00	107	93	97.00
4096	314880	3623	12	1.99	106	89	93.50
8000	612000	4103	11	1.95	99	86	85.86
13824	1054080	7028	12	2.01	146	131	648.41

Table 3: Test 3. Optimality with respect to the mesh size, **Hexa** meshes. *nel*:=number of polyhedra; dofs:=degrees of freedom; N_Π:=number of primal constraints; it:=CG iterations; cond:=experimental condition number.

Octa mesh, k = 1, procs = 32							
nel	dofs	$\nu_{tol} = 2$			$\nu_{tol} = \infty$		
		N_Π	it	cond	N_Π	it	cond
576	9024	528	11	1.87	106	20	4.71
4608	69888	920	11	1.88	104	21	5.53
9000	135600	1366	11	1.84	130	26	7.14
15552	233280	1544	11	1.86	134	26	6.93
30375	453600	2133	11	1.96	144	27	7.03

Octa mesh, k = 2, procs = 32							
nel	dofs	$\nu_{tol} = 2$			$\nu_{tol} = \infty$		
		N_Π	it	cond	N_Π	it	cond
576	23808	1224	11	1.85	106	38	17.31
4608	185856	2342	12	1.82	104	32	11.32
9000	361200	3978	11	1.79	130	44	33.12
15552	622080	4572	11	1.73	134	33	19.50
30375	1210950	6507	11	1.91	144	32	13.87

Octa mesh, k = 3, procs = 32							
nel	dofs	$\nu_{tol} = 2$			$\nu_{tol} = \infty$		
		N_Π	it	cond	N_Π	it	cond
72	6280	1309	10	1.73	109	228	1210.27
576	48320	2089	12	1.95	106	123	204.54
4608	378880	3609	13	2.00	104	90	87.50
9000	737000	6608	12	2.19	130	159	552.21
15552	1270080	7175	12	2.04	134	112	255.88

Table 4: Test 3. Optimality with respect to the mesh size, **Octa** meshes. *nel*:=number of polyhedra; dofs:=degrees of freedom; N_Π:=number of primal constraints; it:=CG iterations; cond:=experimental condition number.

other hand, the performance of the minimal primal space solver deteriorates significantly when the degree k is greater than 2. As a final remark, we also observe that, for **Cube** and **Octa** meshes, the ratio between the

CVT mesh, k = 1, procs = 32							
nel	dofs	$\nu_{tol} = 2$			$\nu_{tol} = \infty$		
		N_Π	it	cond	N_Π	it	cond
1000	23509	812	12	2.05	163	18	4.18
2000	47513	938	12	2.03	157	19	4.97
4000	95785	1018	12	2.04	158	18	4.27
8000	192860	1170	12	2.06	160	19	4.59
16000	388042	1207	12	2.00	147	19	4.60
CVT mesh, k = 2, procs = 32							
nel	dofs	$\nu_{tol} = 2$			$\nu_{tol} = \infty$		
		N_Π	it	cond	N_Π	it	cond
1000	57018	3230	10	2.00	163	40	22.99
2000	115026	3978	10	1.89	157	38	19.70
4000	231570	4966	10	1.96	158	35	14.18
8000	465720	5998	11	1.92	159	34	16.12
16000	936084	7100	11	1.89	147	32	12.91
CVT mesh, k = 3, procs = 32							
nel	dofs	$\nu_{tol} = 2$			$\nu_{tol} = \infty$		
		N_Π	it	cond	N_Π	it	cond
125	13275	2811	12	2.15	165	146	303.36
1000	110030	6810	12	2.03	163	101	141.87
2000	221710	8540	11	1.92	157	97	150.03
4000	445950	10729	11	1.90	158	91	119.28
8000	896200	13091	11	1.98	157	103	118.69

Table 5: Test 3. Optimality with respect to the mesh size, **CVT** meshes. *nel*:=number of polyhedra; dofs:=degrees of freedom; N_Π:=number of primal constraints; it:=CG iterations; cond:=experimental condition number.

Cube mesh, nel = 4096, procs = 32							
k	dofs	$\nu_{tol} = 2$			$\nu_{tol} = \infty$		
		N_Π	it	cond	N_Π	it	cond
1	55552	866	11	1.92	106	20	5.02
2	152064	2258	11	1.76	106	31	11.98
3	314880	3623	12	1.99	106	89	93.50
4	560384	7686	11	1.92	106	382	2052.33
5	904960	9103	12	2.13	106	1353	28663.80
Octa mesh, nel = 576, procs = 32							
k	dofs	$\nu_{tol} = 2$			$\nu_{tol} = \infty$		
		N_Π	it	cond	N_Π	it	cond
1	9024	528	11	1.87	106	20	4.71
2	23808	1224	11	1.85	106	38	17.31
3	48320	2089	12	1.95	106	123	204.54
4	84864	3133	11	1.86	106	510	4913.25
5	135744	4164	11	1.88	106	1754	62878.43
CVT mesh, nel = 1000, procs = 32							
k	dofs	$\nu_{tol} = 2$			$\nu_{tol} = \infty$		
		N_Π	it	cond	N_Π	it	cond
1	23509	812	12	2.05	163	18	4.18
2	57018	3230	12	2.00	163	40	22.99
3	110030	6810	12	2.03	163	101	141.87
4	186545	12629	12	2.03	163	213	741.88
5	290563	21142	11	2.00	163	540	7304.78

Table 6: Test 4. Optimality with respect to the polynomial degree k. nel:=number of polyhedra; dofs:=degrees of freedom; N_Π:=number of primal constraints; it:=CG iterations; cond:=experimental condition number.

number of primal constraints (N_Π) and the total dofs decreases with the polynomial degree k, as it occurs with standard finite elements, while for **CVT** meshes, this ratio increases with k; see Fig. 5.

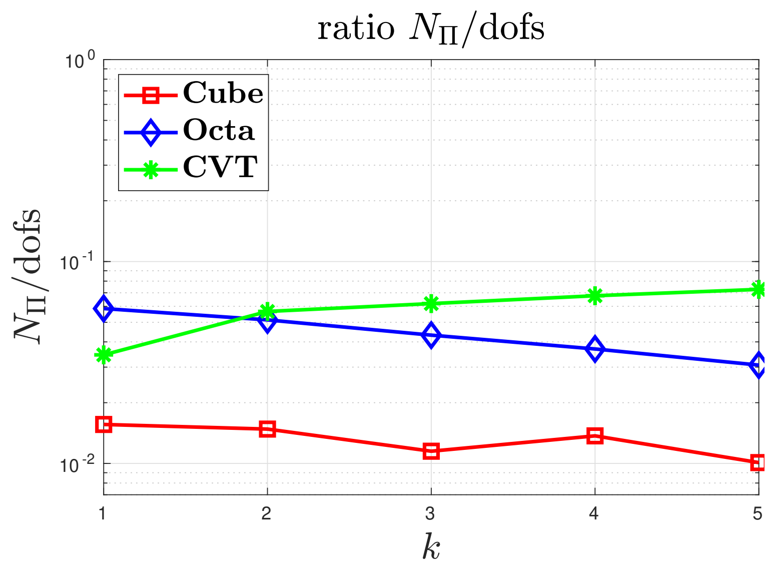


Figure 5: Test 4. Optimality with respect to the polynomial degree k. Plot of the ratio between the number of primal constraints (N_Π) and the total number of dofs for the adaptive BDDC preconditioner as a function of k.

CVT mesh, nel = 16000, dofs = 388042 k = 1, procs = 32						
τ	$\nu_{tol} = 5$			$\nu_{tol} = \infty$		
	N_Π	it	cond	N_Π	it	cond
0.01	3128	19	4.52	147	64	53.68
1	147	19	4.60	147	19	4.60
100	4732	19	4.39	147	76	87.16
CVT mesh, nel = 8000, dofs = 465720 k = 2, procs = 32						
τ	$\nu_{tol} = 5$			$\nu_{tol} = \infty$		
	N_Π	it	cond	N_Π	it	cond
0.01	7452	19	4.96	159	103	104.06
1	1184	21	5.56	159	34	16.12
100	1844	22	5.62	159	53	75.04

Table 7: Test 5. Robustness with respect to the stabilization parameter τ. N_Π:=number of primal constraints; it:=CG iterations; cond:=experimental condition number.

5.5. Test 5: robustness with respect to the stabilization parameter τ

In this test, we investigate how the stabilization parameter τ in the bilinear form $s_P(\cdot,\cdot)$ defined in (7) affects the performance of the BDDC solver. We report in Table 7 only the results obtained on **CVT** meshes, for $k = 1$ and $k = 2$, being the behavior of the solver in case of **Cube** and **Octa** meshes analogous. We let τ assuming the following values: $0.01, 1, 100$. The number of processors is fixed to 32. The results show that the adaptive solver (with $\nu_{tol} = 5$) is robust with respect to τ, since it is able to enrich the primal space in order to maintain the condition number controlled by the adaptive tolerance, resulting in a number of CG iterations independent of τ. When employing the minimal primal space solver ($\nu_{tol} = \infty$), instead, the condition number and iterations counts are significantly affected from the value of τ, being the choice $\tau = 1$ the most effective.

5.6. Test 6: robustness with respect to jumping coefficients

We investigate here the robustness of the BDDC solver with respect to jumps in the parameter ν of the bilinear form $\boldsymbol{a}(\cdot,\cdot)$ in (2). We divide the unit cube domain into $32 = 4 \times 4 \times 2$ subdomains, coloured with two colors (black and red), according to a checkerboard configuration. In the black subdomains, we set $\nu = 1$, whereas in the red subdomains we set $\nu = \nu_{red}$ as given in Table 8. For sake of conciseness, we report only the results obtained on **CVT** meshes, for $k = 1$ and $k = 2$, since the behavior of the solver in case of **Cube** and **Octa** meshes is analogous. The number of processors is fixed to 32. We note here that even if the jump. configuration is structured, the mesh partitioning is instead unstructured, and it is not aligned with the coefficient's jumps. The results show that the adaptive solver (with $\nu_{tol} = 5$) is robust with respect to jumps in ν, since both the

CVT mesh, nel = 16000, dofs = 388042						
k = 1, procs = 32						
ν_{red}	$\nu_{tol} = 5$			$\nu_{tol} = \infty$		
	N_Π	it	cond	N_Π	it	cond
1e-4	235	27	5.10	147	117	2.77e+3
1e-2	226	22	5.20	147	39	28.84
1	147	19	5.60	147	19	4.60
1e+2	214	20	4.91	147	25	9.77
1e+4	237	20	4.91	147	27	13.32
CVT mesh, nel = 8000, dofs = 465720						
k = 2, procs = 32						
ν_{red}	$\nu_{tol} = 5$			$\nu_{tol} = \infty$		
	N_Π	it	cond	N_Π	it	cond
1e-4	1614	26	5.72	159	194	3.37e+3
1e-2	1582	25	5.72	159	47	44.28
1	1184	21	5.56	159	34	16.12
1e+2	1579	22	5.74	159	50	95.12
1e+4	1619	22	5.76	159	128	6.52e+3

Table 8: Test 6. Robustness with respect to jumping coefficients. N_Π:=number of primal constraints; it:=CG iterations; cond:=experimental condition number.

condition number and the CG iteration counts remain bounded. Moreover, the growth of the primal space with respect to the jump magnitude is not large, because the dimension of the primal space remains of the same order when varying ν_{red}. The minimal primal space solver ($\nu_{tol} = \infty$), instead, is not robust with respect to the jumps, since both the condition number and iterations counts increase significantly when ν_{red} is far from 1.

Cube meshes, procs = 64								
nel	k	dofs	Pardiso	MUMPS	Block-Schur		BDDC $\nu_{tol} = 5$	
			T_{sol}	T_{sol}	it	T_{sol}	it	T_{sol}
85184	1	1124816	1397	479	79	64	20	25
32768	2	1198080	1708	621	155	32	20	32
13824	3	1054080	1358	325	745	85	22	43
Octa meshes, procs = 64								
nel	k	dofs	Pardiso	MUMPS	Block-Schur		BDDC $\nu_{tol} = 5$	
			T_{sol}	T_{sol}	it	T_{sol}	it	T_{sol}
30375	1	453600	167	63	80	15	21	5
30375	2	1210950	1166	892	577	96	20	37
15552	3	1270080	1665	785	1468	201	18	60
CVT meshes, procs = 64								
nel	k	dofs	Pardiso	MUMPS	Block-Schur		BDDC $\nu_{tol} = 5$	
			T_{sol}	T_{sol}	it	T_{sol}	it	T_{sol}
16000	1	388042	1916	742	112	11	18	27
16000	2	936084	14992	5926	690	70	22	201
8000	3	896200	15735	5028	3510	318	21	265

Table 9: Test 7. Performance comparison among different parallel solvers. nel:=number of polyhedra; k:=degree of VEM approximation; dofs:=degrees of freedom; T_{sol}:=solution time in seconds; it:=GMRES (Blcok-Schur) and CG (BDDC) iterations.

5.7. Test 7: performance comparison with other parallel solvers

Here we compare the performance in terms of CPU times of the adaptive BDDC solver against the parallel direct solvers Pardiso and MUMPS, as well as against the parallel Block-Schur iterative solver proposed in [28]. The Block-Schur solver is a block-diagonal preconditioner of the form

$$\mathcal{B}_D = \begin{bmatrix} \mathrm{B}_1 & 0 \\ 0 & \mathrm{B}_2 \end{bmatrix}, \qquad (19)$$

where

$$\begin{aligned} B_1^{-1} &= \text{diagonal preconditioner for } A, \text{ i.e. } B_1 = diag(A) \\ B_2^{-1} &= \text{exact solution of the approximate Schur complement } S_{app} \end{aligned} \tag{20}$$

with $S_{app} = -B\ diag(A)^{-1}B^T$. The inversion of S_{app} at each preconditioning step is obtained by MUMPS.

With the adaptive BDDC preconditioner, we use CG as iterative solver, whereas we use the GMRES method with the Block-Schur preconditioner. We consider **Cube**, **Octa** and **CVT** meshes, degree of the VEM approximation $k = 1, 2, 3$ and 64 processors. With the adaptive BDDC solver, the adaptive tolerance is set to $\nu_{tol} = 5$. The results are reported in Table 9.

For all meshes, adaptive BDDC is always significantly faster than Pardiso and MUMPS. For **Cube** and **Octa** meshes, adaptive BDDC is also faster than Block-Schur. Only for **CVT** meshes, with $k = 1$ and $k = 2$, adaptive BDDC results slower than Block-Schur, while for $k = 3$ adaptive BDDC is again about 25% faster than Block-Schur.

6. Conclusions

In this work, we have constructed and numerically studied adaptive BDDC preconditioners for the saddle point linear systems arising from general order three-dimensional VEM approximations of scalar elliptic equations in mixed form. The results show the reliability of the adaptive procedure, which yields a condition number of the preconditioned system close to the prescribed adaptive tolerance parameter ν_{tol}. Several parallel tests demonstrates the scalability and optimality of the proposed solver. The comparison with a BDDC preconditioner with minimal coarse space shows that the adaptive solver is more robust with respect to high-order VEM approximations and jumps in the coefficients. In terms of CPU times, the adaptive BDDC solver outperforms the parallel direct solvers Pardiso and MUMPS, and our Block-Schur iterative solvers, especially in the case of high order VEM approximations.

Acknowledgments

The authors would like to acknowledge INDAM-GNCS for the support. Moreover they would like to thank Lourenco Beirão da Veiga and Alessandro Russo for many helpful discussions and suggestions.

References

[1] P. R. Amestoy, I. S. Duff, J.-Y. L'Excellent, and J. Koster. A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM J. Matr. Anal. Appl.*, 23(1):15–41, 2001.

[2] P. R. Amestoy, A. Guermouche, J.-Y. L'Excellent, and S. Pralet. Hybrid scheduling for the parallel solution of linear systems. *Paral. Comput.*, 32(2):136–156, 2006.

[3] P. F. Antonietti, L. Mascotto, and M. Verani. A multigrid algorithm for the p-version of the virtual element method. *ESAIM: Math. Model. Numer. Anal.*, 52(1):337–364, 2018.

[4] O. Axelsson. *Iterative Solution Methods.* Cambridge University Press, 1994.

[5] O. Axelsson, R. Blaheta, P. Byczanski, J. Karátson, and B. Ahmad. Preconditioners for regularized saddle point problems with an application for heterogeneous Darcy flow problems. *J. Comput. Appl. Math.*, 280:141–157, 2015.

[6] B. Ayuso de Dios, K. Lipnikov, and G. Manzini. The nonconforming virtual element method. *ESAIM: Math. Model. Numer. Anal.*, 50(3):879–904, 2016.

[7] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, D. A. May, L. C. McInnes, R. T. Mills, T. Munson, K. Rupp, P. Sanan, B. F. Smith, S. Zampini, H. Zhang, and H. Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 3.14, Argonne National Laboratory, 2020.

[8] L. Beirão da Veiga, F. Brezzi, L. D. Marini, and A. Russo. The hitchhiker's guide to the virtual element method. *Math. Mod. Meth. Appl. Sci.*, 24(08):1541–1573, 2014.

[9] L. Beirão da Veiga, F. Dassi, and A. Russo. High-order virtual element method on polyhedral meshes. *Comput. Math. Appl.*, 74(5):1110–1122, 2017.

[10] L. Beirão da Veiga, F. Brezzi, A. Cangiani, G. Manzini, L. D. Marini, and A. Russo. Basic principles of virtual element methods. *Math. Mod. Meth. Appl. Sci.*, 23(1):199–214, 2013.

[11] L. Beirão Da Veiga, F. Brezzi, F. Dassi, L. D. Marini, and A. Russo. Serendipity virtual elements for general elliptic equations in three dimensions. *Chinese Ann. Math., Series B*, 39(2):315–334, Mar 2018.

[12] L. Beirão da Veiga, F. Brezzi, L. D. Marini, and A. Russo. H(div) and h(curl)-conforming virtual element methods. *Numer. Math.*, 133:303–332, 2016.

[13] L. Beirão da Veiga, F. Brezzi, L. D. Marini, and A. Russo. Mixed virtual element methods for general second order elliptic problems on polygonal meshes. *ESAIM: Math. Model. Numer. Anal.*, 50(3):727–747, 2016.

[14] L. Beirão da Veiga and A. Ern. Preface. *ESAIM: Math. Model. Numer. Anal.*, 50(3):633–634, 2016.

[15] L. Beirão da Veiga, C. Lovadina, and G. Vacca. Divergence free virtual elements for the stokes problem on polygonal meshes. *ESAIM: Math. Model. Numer. Anal.*, 51(2):509–535, 2017.

[16] M. F. Benedetto, S. Berrone, S. Pieraccini, and S. Scialò. The virtual element method for discrete fracture network simulations. *Comput. Meth. Appl. Mech. Eng.*, 280:135–156, 2014.

[17] M. Benzi, G. H. Golub, and J. Liesen. Numerical solution of saddle point problems. *Acta numerica*, 14:1–137, 2005.

[18] S. Bertoluzza, M. Pennacchio, and D. Prada. BDDC and FETI-DP for the virtual element method. *Calcolo*, 54(4):1565–1593, 2017.

[19] S. Bertoluzza, M. Pennacchio, and D. Prada. FETI-DP for the three dimensional virtual element method. *SIAM J. Numer. Anal.*, 58(3):1556–1591, January 2020.

[20] D. Boffi, F. Brezzi, and M. Fortin. *Mixed finite element methods and applications*, volume 44 of *Springer Series in Computational Mathematics*. Springer, Heidelberg, 2013.

[21] S. C. Brenner and L. R. Scott. *The mathematical theory of finite element methods*, volume 15 of *Texts in Applied Mathematics*. Springer, New York, third edition, 2008.

[22] F. Brezzi, R. S. Falk, and L. D. Marini. Basic principles of mixed virtual element methods. *ESAIM: Math. Model. Numer. Anal.*, 48(4):1227–1240, 2014.

[23] J. G. Calvo. On the approximation of a virtual coarse space for domain decomposition methods in two dimensions. *Math. Mod. Meth. Appl. Sci.*, 28(07):1267–1289, 2018.

[24] J. G. Calvo. An overlapping schwarz method for virtual element discretizations in two dimensions. *Comput. Math. Appl.*, 77(4):1163–1177, 2019.

[25] J. G. Calvo and O. B. Widlund. An adaptive choice of primal constraints for bddc domain decomposition algorithms. *Electron. Trans. Numer. Anal.*, 45:524–544, 2016.

[26] F. Dassi and L. Mascotto. Exploring high-order three dimensional virtual elements: Bases and stabilizations. *Comput. Math. Appl.*, 75(9):3379–3401, 2018.

[27] F. Dassi and S. Scacchi. Parallel block preconditioners for three-dimensional virtual element discretizations of saddle-point problems. *Comput. Meth. Appl. Mech. Eng.*, 372:113424, 2020.

[28] F. Dassi and S. Scacchi. Parallel solvers for virtual element discretizations of elliptic equations in mixed form. *Comput. Math. Appl.*, 79(7):1972–1989, April 2020.

[29] A. De Coninck, B. De Baets, D. Kourounis, F. Verbosio, O. Schenk, S. Maenhout, and J. Fostier. Needles: Toward large-scale genomic prediction with marker-by-environment interaction. 203(1):543–555, 2016.

[30] C. R Dohrmann. A preconditioner for substructuring based on constrained energy minimization. *SIAM J. Sci. Comput.*, 25(1):246–258, 2003.

[31] C. R. Dohrmann and O. B. Widlund. A BDDC algorithm with deluxe scaling for three-dimensional H (curl) problems. *Comm. Pure Appl. Math.*, 69(4):745–770, 2016.

[32] G. H. Golub and C. Greif. On solving block-structured indefinite linear systems. *SIAM J. Sci. Comput.*, 24(6):2076–2092, 2003.

[33] A. Heinlein, C. Hochmuth, and A. Klawonn. Monolithic overlapping schwarz domain decomposition methods with GDSW coarse spaces for incompressible fluid flow problems. *SIAM J. Sci. Comput.*, 41(4):C291–C316, 2019.

[34] A. Heinlein, C. Hochmuth, and A. Klawonn. Reduced dimension GDSW coarse spaces for monolithic schwarz domain decomposition methods for incompressible fluid flow problems. *Int. J. Numer. Meth. Eng.*, 121(6):1101–1119, 2019.

[35] A. Klawonn. Block-triangular preconditioners for saddle point problems with a penalty term. *SIAM J. Sci. Comput.*, 19(1):172–184, 1998.

[36] A. Klawonn and L. F. Pavarino. A comparison of overlapping schwarz methods and block preconditioners for saddle point problems. *Numer. Lin. Alg. Appl.*, 7(1):1–25, 2000.

[37] D. Kourounis, A. Fuchs, and O. Schenk. Towards the next generation of multiperiod optimal power flow solvers. *IEEE Transactions on Power Systems*, PP(99):1–10, 2018.

[38] J. Li and O. B. Widlund. BDDC algorithms for incompressible Stokes equations. *SIAM J. Numer. Anal.*, 44(6):2432–2455, 2006.

[39] J. Li and O. B Widlund. FETI-DP, BDDC, and block Cholesky methods. *Int. J. Numer. Meth. Eng.*, 66(2):250–271, 2006.

[40] K.-A. Mardal and R. Winther. Preconditioning discretizations of systems of partial differential equations. *Numer. Lin. Alg. Appl.*, 18(1):1–40, 2011.

[41] L. Mascotto. Ill-conditioning in the virtual element method: Stabilizations and bases. *Numer. Meth. Part. Diff. Eq.*, 34(4):1258–1281, 2018.

[42] T. P. Mathew. Schwarz alternating and iterative refinement methods for mixed formulations of elliptic problems, part i: Algorithms and numerical results. *Numer. Math.*, 65(1):445–468, 1993.

[43] T. P. Mathew. Schwarz alternating and iterative refinement methods for mixed formulations of elliptic problems, part II: Convergence theory. *Numer. Math.*, 65(1):469–492, 1993.

[44] 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. Comput.*, 87(310):659–692, 2017.

[45] L. F. Pavarino. Indefinite overlapping schwarz methods for time-dependent stokes problems. *Comput. Meth. Appl. Mech. Eng.*, 187(1–2):35–51, 2000.

[46] L. F. Pavarino, S. Scacchi, O. B. Widlund, and S. Zampini. Isogeometric BDDC deluxe preconditioners for linear elasticity. *Math. Mod. Meth. Appl. Sci.*, 28(07):1337–1370, 2018.

[47] L. F. Pavarino and O. B. Widlund. Balancing neumann-neumann methods for incompressible stokes equations. *Comm. Pure Appl. Math.*, 55(3):302–335, 2001.

[48] L. F. Pavarino, O. B. Widlund, and S. Zampini. BDDC preconditioners for spectral element discretizations of almost incompressible elasticity in three dimensions. *SIAM J. Sci. Comput.*, 32(6):3604–3626, 2010.

[49] Y. Saad. *Iterative Methods for Sparse Linear Systems.* SIAM, 2003.

[50] V. Simoncini. Block triangular preconditioners for symmetric saddle-point problems. *Appl. Numer. Math.*, 49(1):63–80, 2004.

[51] A. Toselli and O. B. Widlund. *Domain decomposition methods - Algorithms and theory*, volume 34. Springer Science & Business Media, 2006.

[52] X. Tu. A BDDC algorithm for a mixed formulation of flow in porous media. *Electron. Trans. Numer. Anal*, 20:164–179, 2005.

[53] X. Tu. A three-level BDDC algorithm for a saddle point problem. *Numer. Math.*, 119(1):189–217, 2011.

[54] G. Vacca and L. Beirão da Veiga. Virtual element methods for parabolic problems on polygonal meshes. *Numer. Meth. Part. Diff. Eq.*, 31(6):2110–2134, 2015.

[55] F. Verbosio, A. De Coninck, D. Kourounis, and O. Schenk. Enhancing the scalability of selected inversion factorization algorithms in genomic prediction. *J. Comput. Sci.*, 22(Supplement C):99–108, 2017.

[56] O. B. Widlund. BDDC domain decomposition algorithms. In *75 Years of Mathematics of Computation: Symposium on Celebrating 75 Years of Mathematics of Computation, November 1-3, 2018, the Institute for Computational and Experimental Research in Mathematics (ICERM), Providence, Rhode Island*, volume 754. American Mathematical Soc., 2020.

[57] O. B. Widlund, S. Zampini, L. F. Pavarino, and S. Scacchi. Block FETI–DP/BDDC preconditioners for mixed isogeometric discretizations of three-dimensional almost incompressible elasticity. *Math. Comput.*, page 1, 2021.

[58] S. Zampini. PCBDDC: a class of robust dual-primal methods in PETSc. *SIAM J. Sci. Comput.*, 38(5):S282–S306, 2016.

[59] S. Zampini. Adaptive BDDC deluxe methods for H (curl). In *Domain Decomposition Methods in Science and Engineering XXIII*, pages 285–292. Springer, 2017.

[60] S. Zampini and D. E Keyes. On the robustness and prospects of adaptive BDDC methods for finite element discretizations of elliptic PDEs with high-contrast coefficients. In *Proceedings of the Platform for Advanced Scientific Computing Conference*, pages 1–13, 2016.

[61] S. Zampini and X. Tu. Multilevel balancing domain decomposition by constraints deluxe algorithms with adaptive coarse spaces for flow in porous media. *SIAM J. Sci. Comput.*, 39(4):A1389–A1415, 2017.

[62] S. Zampini, P. Vassilevski, V. Dobrev, and T. Kolev. Balancing domain decomposition by constraints algorithms for curl-conforming spaces of arbitrary order. In *International Conference on Domain Decomposition Methods*, pages 103–116. Springer, 2017.

[63] W. Zulehner. Analysis of iterative methods for saddle point problems: a unified approach. *Math. Comp.*, 71(238):479–506, 2001.