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

Franco Dassi*\textsuperscript{a,1}, Stefano Zampini\textsuperscript{b,1}, S. Scacchi\textsuperscript{c,1}

\textsuperscript{a}Dipartimento di Matematica e Applicazioni, Università degli Studi di Milano Bicocca, Via Roberto Cozzi 55 - 20125 Milano, Italy

\textsuperscript{b}Extreme Computing Research Center, King Abdullah University of Science and Technology, Thuwal 23955, Saudi Arabia

\textsuperscript{c}Dipartimento di Matematica, Università degli Studi di Milano, Via Saldini 50, 20133 Milano, Italy

\textsuperscript{*}Corresponding author

Email addresses: franco.dassi@unimib.it (Franco Dassi), stefano.zampini@kaust.edu.sa (Stefano Zampini), simone.scacchi@unimi.it (S. Scacchi)

\textsuperscript{1}Member of INdAM-GNCS research group

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,
overlapping preconditioners have been developed for mixed formulation of linear elasticity and Stokes equations \cite{35, 36, 45, 33, 34} and for scalar elliptic equations in mixed form \cite{42, 43}, while non-overlapping preconditioners have been developed for Stokes equations \cite{47}, almost incompressible elasticity \cite{48} and scalar elliptic equations in mixed form \cite{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 \cite{30} that belong to the non-overlapping class. Recently, a considerable effort has been put into developing multilevel \cite{53} and adaptive \cite{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 \cite{61}, an effective adaptive BDDC algorithm has been proposed for finite element discretizations 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 \cite{1, 2} and Pardiso \cite{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 $\Omega$ be a bounded Lipschitz domain in $\mathbb{R}^3$, whose boundary is denoted by $\partial \Omega$. We define the function spaces

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

and

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

where $H(\text{div}, \Omega)$ is the space of vector-valued functions such that $u$ and $\text{div}(u)$ 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:

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

(1)

where

$$a(u, v) := \int_{\Omega} \nu u \cdot v \, d\Omega,$$

$$b(u, q) := \int_{\Omega} \text{div}(u) q \, d\Omega,$$

(2)

with $f$ a given function in $L^2(\Omega)$ satisfying the compatibility condition $\int_{\Omega} f \, d\Omega = 0$, and $\nu$ 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. \cite{20}. Functions $u$ and $p$ are usually called velocity and pressure. We refer to \cite{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 $\Omega_h$ 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 \cite{22}; for implementation details on the discretization technique we suggest \cite{28}.
3.1. Virtual element spaces

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

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

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

- **normal face moments**
  \[
  \int_f (\mathbf{v}_h \cdot \mathbf{n}) p_k \, df \quad \forall p_k \in \mathbb{P}_k(F) \quad \forall f \in \partial P,
  \]

- **internal gradient moments**
  \[
  \int_P \mathbf{v}_h \cdot \nabla p_{k-1} \, dP \quad \forall p_{k-1} \in \mathbb{P}_{k-1}(P),
  \]

- **internal cross moments**
  \[
  \int_P \mathbf{v}_h \cdot (\mathbf{x} \times p_{k-1}) \, dP \quad \forall p_{k-1} \in [\mathbb{P}_{k-1}(P)]^3.
  \]

where \( \mathbf{x} = (x, y, z)^t \).

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

\[
Q_h(P) := \left\{ q_h \in L^2(P) : q_h \in \mathbb{P}_{k-1}(P) \right\},
\]

that is uniquely determined by the following degrees of freedom

- **internal pressure moments**
  \[
  \int_P q_h \, p_{k-1} \, dP \quad \forall p_{k-1} \in \mathbb{P}_{k-1}(P).
  \]

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.

\[
\mathbf{V}_h^k(\Omega_h) = \{ \mathbf{v}_h \in \mathbf{V} : \mathbf{v}_{h|P} \in \mathbf{V}_h^k(P) \}
\]

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 \( \text{div}(\mathbf{v}_h) \) 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 \( \mathbf{b}(\mathbf{u}_h, q_h) \) can be computed on each polyhedron because both \( q_h \) and \( \text{div}(\mathbf{u}_h) \) are piece-wise polynomials. The bi-linear form \( \mathbf{a}(\mathbf{u}, \mathbf{v}) \) is not computable since it requires knowledge of the velocity field inside each polyhedron, and we thus substitute it with a suitable approximation \( \mathbf{a}_{h,p} \) that is computable from the degrees of freedom

\[
\mathbf{a}(\mathbf{u}, \mathbf{v}) \approx \sum_{P \in \Omega_h} \mathbf{a}_{h,p}(\mathbf{u}_h, \mathbf{v}_h),
\]

where

\[
\mathbf{a}_{h,p}(\mathbf{v}_h, \mathbf{w}_h) := \int_P \nu(\mathbf{x}) (\Pi^0_k \mathbf{u}_h \cdot \Pi^0_k \mathbf{v}_h \, dP + s_p(\mathbf{u}_h - \Pi^0_k \mathbf{u}_h, \mathbf{v}_h - \Pi^0_k \mathbf{v}_h), \quad (i)
\]

(5)

In the above definition we distinguish two parts:
i) The consistency part that is based on the $L^2$-projection operator $\Pi_k^{0, P} : V_h^k(P) \to [P_k(P)]^3$ defined from the following identity:

$$\int_P \Pi_k^{0, P} v_h \cdot p_h \, dP = \int_P v_h \cdot p_h \, dP. \quad (6)$$

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]

$$p_k = \nabla p_{k+1} + \mathbf{x} \wedge p_{k-1},$$

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 $a(\cdot, \cdot)$. In other words, we require that there exists two positive constants $\alpha_*$ and $\alpha^*$, independent on the diameter of the polyhedron $P$, such that

$$\alpha_* a(v_h, v_h) \leq s_P(v_h, v_h) \leq \alpha^* a(v_h, v_h), \quad \forall v_h \in V_h^k(P).$$

While there are many choices in literature, see e.g., [9, 26, 41, 10, 8, 13], in this paper we use the so-called “dof1-dof1” stabilization [8, 13] and consider the Euclidean scalar product associated with the degrees of freedom of $V_h^k(P)$ multiplied by the volume of $P$ and the value of $\nu(x)$ at its barycenter

$$s_P(u_h, v_h) = \tau \nu(x_P) |P| \sum_{i=1}^{\#dof_P} \text{dof}_i(u_h) \text{dof}_i(v_h), \quad (7)$$

where $\#dof_P$ is the number of degrees of freedom associated with $V_h^k(P)$, $\text{dof}_i : V_h^k(P) \to \mathbb{R}$ the linear functional associating functions in $V_h^k(P)$ 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

$$\begin{cases}
\text{find } (u_h, p_h) \in V_h^k(\Omega_h) \times Q_h(\Omega_h) : \\
\alpha_h(u_h, v_h) - b(v_h, p_h) = 0, \quad \forall v \in V_h^k(\Omega_h) \\
b(u_h, q_h) = \int_{\Omega_h} f q_h \, d\Omega_h, \quad \forall q_h \in Q_h(\Omega_h),
\end{cases} \quad (8)$$

or equivalently in matrix form

$$\begin{bmatrix}
A & B^T \\
B & 0
\end{bmatrix}
\begin{bmatrix}
u_h \\
p_h
\end{bmatrix} =
\begin{bmatrix}
0 \\
f_h
\end{bmatrix}. \quad (9)$$

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 $\Omega_h$ 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 $(u_h, p_h) \in V_h^k(\Omega_h) \times Q_h(\Omega_h)$ and, assuming enough regularity on the solution, the following error estimates hold

$$\|u - u_h\|_0 \leq C h^{k+1} \|u\|_{k+1},$$
$$\|p - p_h\|_0 \leq C h^k (\|p\|_k + \|u\|_k),$$
$$\|\text{div}(u - u_h)\|_0 \leq C h^k \|f\|_k,$$

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.
4.1. Non-overlapping domain decomposition

Within the framework of iterative sub-structuring [51] we decompose \( \Omega_h \) into \( N \) non-overlapping connected subdomains

\[
\Pi_h = \bigcup_{i=1}^{N} \Pi_i, \quad \Gamma = \bigcup_{i \neq j} \partial \Omega_j \cap \partial \Omega_i,
\]

where each \( \Omega_i \) is a collection of polyhedrons, and where \( \Gamma \) denotes the interface among subdomains.

We then consider the splitting of \( V_h(\Omega_h) \) into a direct sum of two subspaces \( V_I \oplus V_\Gamma \). \( V_I \) contains virtual functions identified by degrees of freedom that are associated to only one of the \( \Omega_i \), whereas the virtual functions in \( V_\Gamma \) 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 \( b(\mathbf{u}_h, q_h) \), we can use the divergence theorem and rewrite Equation (9) as

\[
\begin{bmatrix}
A_{II} & B^T_{II} & A_{I\Gamma} & 0 \\
B_{II} & 0 & B_{I\Gamma} & 0 \\
A^T_{I\Gamma} & B^T_{I\Gamma} & A_{\Gamma\Gamma} & B^T_{0\Gamma} \\
0 & 0 & B_{0\Gamma} & 0
\end{bmatrix}
\begin{bmatrix}
\mathbf{u}_I \\
\mathbf{p}_I \\
\mathbf{u}_\Gamma \\
\mathbf{p}_0
\end{bmatrix}
= \begin{bmatrix}
\mathbf{0} \\
\mathbf{f}_I \\
\mathbf{0} \\
\mathbf{0}
\end{bmatrix}.
\]

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

\[
\begin{bmatrix}
I & -K^{-1}_{II}K_{I\Gamma} \\
0 & I
\end{bmatrix}
\begin{bmatrix}
K^{-1}_{II} & 0 \\
0 & S_I
\end{bmatrix}
\begin{bmatrix}
I & 0 \\
-K^T_{II}K^{-1}_{II} & I
\end{bmatrix},
\]

where

\[
K_{II} = \begin{bmatrix} A_{II} & 0 \\ B_{II} & 0 \end{bmatrix}, \quad K_{I\Gamma} = \begin{bmatrix} A_{II} & B^T_{II} \\ B_{II} & 0 \end{bmatrix}, \quad S_T = A_{\Gamma\Gamma}^{-1} \begin{bmatrix} A_{I\Gamma} & B^T_{II} \\ A_{II} & B_{II} \end{bmatrix}^T \begin{bmatrix} A_{II} & B^T_{II} \\ B_{II} & 0 \end{bmatrix}, \quad S = \begin{bmatrix} S_T & B^T_{0\Gamma} \\ B_{0\Gamma} & 0 \end{bmatrix}.
\]

We note here that, recalling that \( A_{II} \) and

\[
\begin{bmatrix}
A_{II} & A_{I\Gamma} \\
A^T_{I\Gamma} & A_{\Gamma\Gamma}
\end{bmatrix}
\]
are positive definite and due to their saddle point structure, the matrices
\[
\begin{bmatrix}
A_{II} & B_{II}^T & A_{I\Gamma} \\
B_{II} & 0 & B_{I\Gamma} \\
A_{I\Gamma}^T & B_{I\Gamma}^T & A_{\Gamma\Gamma}
\end{bmatrix}
\text{ and }
\begin{bmatrix}
A_{II} & B_{II}^T \\
B_{II} & 0 \\
0 & 0 & S_{\Gamma}
\end{bmatrix},
\]

have the same number of negative eigenvalues. Then, by Sylvester’s law of inertia, the matrices
\[
\begin{bmatrix}
A_{II} & B_{II}^T & A_{I\Gamma} \\
B_{II} & 0 & B_{I\Gamma} \\
A_{I\Gamma}^T & B_{I\Gamma}^T & A_{\Gamma\Gamma}
\end{bmatrix}
\text{ and }
\begin{bmatrix}
A_{II} & B_{II}^T & 0 \\
B_{II} & 0 & 0 \\
0 & 0 & S_{\Gamma}
\end{bmatrix},
\]

have the same number of positive and negative eigenvalues, from which it follows that the velocity Schur complement \(S\) 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 \(S\) will have a null component in such a kernel.

Within the preconditioner, the pseudo-inverse of the saddle point Schur complement \(S\) is replaced by the action of a suitable preconditioner, resulting in the preconditioner
\[
M^{-1} = \begin{bmatrix}
I & -K_{II}^{-1}K_{I\Gamma} \\
0 & K_{II}^{-1} \end{bmatrix}
\begin{bmatrix}
I & 0 \\
-K_{II}^{-1}K_{I\Gamma} & M_{\Gamma}^{-1}
\end{bmatrix}
\begin{bmatrix}
I & 0 \\
-K_{II}^{-1}K_{I\Gamma} & I
\end{bmatrix}.
\]

(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 \(\tilde{V}_\Gamma\), direct sum of a primal, continuous, space \(V_\Pi\) and a dual, discontinuous space \(V_\Delta\), such that \(V_\Gamma \subset \tilde{V}_\Gamma = V_\Delta \oplus V_\Pi\)

- a scaling operator \(D : V_\Gamma \to \tilde{V}_\Gamma\) 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 \(H(\text{curl})\) [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 \(\{V_\Gamma \cap \ker (B_{\Gamma I})\} \times Q_0\). Following [52], we then require that the virtual functions in the dual space satisfy the so-called no-net-flux condition
\[
\int_{\partial \Omega_i} v_\Delta \cdot n = 0, \quad \forall v_\Delta \in V_\Delta.
\]

(14)

Since we have assumed that each \(\Omega_i\) is connected, in our algorithm we will consider one constraint for each subdomain face \(F\)
\[
\int_F v_\Delta \cdot n = 0.
\]

The quadrature weights associated with the above constraint can be computed by considering the linear functional \(b_{\cdot} (q_0)\) restricted to those virtual functions in \(V_\Delta\) represented by nonzero degrees of freedom on \(F\) only, and where \(q_0 \in Q_0\) is defined as \(q_0 = 1\) in \(\Omega_i\), and zero otherwise. In turn, the virtual functions in the primal space will be characterized by the continuity of
\[
\int_F v_\Pi \cdot n, \quad \forall v_\Pi \in V_\Pi, \forall F \subset \Gamma.
\]

(15)

for each subdomain face \(F\).

Using the divergence theorem, it is easy to see that the VEM problem on \(\{V_\Pi \oplus \tilde{V}_\Gamma\} \times \{Q_I \oplus Q_0\}\) can be recast as
\[
\begin{bmatrix}
A_{II} & B_{II}^T & A_{I\Delta} & A_{I\Pi} & 0 \\
B_{II} & 0 & B_{I\Delta} & B_{I\Pi} & 0 \\
A_{I\Delta}^T & B_{I\Delta}^T & A_{\Delta\Pi} & A_{\Delta\Pi} & 0 \\
A_{I\Pi}^T & B_{I\Pi}^T & A_{\Pi\Pi} & B_{\Pi\Pi}^T & 0 \\
0 & 0 & 0 & B_{0\Pi} & 0
\end{bmatrix}.
\]

(16)
Eliminating the degrees of freedom of \( V_I \times Q_I \) we can define a partially assembled Schur complement on \( \tilde{V}_F \times Q_0 \)

\[
\begin{bmatrix}
S_{\Delta\Delta} & S_{\Delta\Pi} & 0 \\
S_{\Pi\Delta} & S_{\Pi\Pi} & B_{\Pi\Pi}' \\
0 & B_{0\Pi} & 0
\end{bmatrix}, \quad \tilde{S}_F = \begin{bmatrix}
S_{\Delta\Delta} & S_{\Delta\Pi} \\
S_{\Pi\Delta} & S_{\Pi\Pi}
\end{bmatrix}
\]

and define the BDDC preconditioner as

\[
M^{-1}_F = D'\tilde{S}D.
\]

By using block factorization arguments, the action of \( \tilde{S} \) can be implemented as additive combination of uncoupled subdomain solvers defined on \( \{ V_I \oplus V_\Delta \} \times Q_I \) and the solution of a coarse saddle point problem on \( V_\Pi \times Q_0 \) which inherits the kernel from \( S \).

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

\[
\begin{bmatrix}
u \bar{u}_h^* \\
\left[ \begin{array}{c}
\bar{u}_h^* \cr \bar{f}_h
\end{array} \right]
\end{bmatrix} = M^{-1}_F \begin{bmatrix}
u \bar{u}_h^* \\
\left[ \begin{array}{c}
\bar{u}_h^* \\
\bar{f}_h
\end{array} \right]
\end{bmatrix}, \quad (u_h^*)_I = -K^{-1}_I K_F u_h^* ,
\]

and use conjugate gradient iterations to solve the deflated system

\[
\begin{bmatrix}
A & B^T \\
B & 0
\end{bmatrix} \begin{bmatrix}
\bar{u}_h^* \\
\bar{p}_h
\end{bmatrix} = \begin{bmatrix}
-Au_h^* \\
-f_h - Bu_h^*
\end{bmatrix}.
\]

Convergence rate estimates for BDDC preconditioned conjugate gradient methods are completely characterized by the norm of the average operator \( ||D'||_{\bar{\Omega}_F} \); 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 \( ||D'||_{\bar{\Omega}_F} \), we can devise generalized eigenvalue problems, defined on each subdomain face \( F \), to enrich the minimal primal space \( V_\Pi \) 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 \( \nu_{tol} \) times a constant independent on \( h, H, N \).

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 \( \Gamma_i \) and \( \Gamma_j \), we solve the following generalized eigenvalue problem (separately on each subdomain face)

\[
\tilde{S}^{(i)}_{F \Delta F \Delta} : \tilde{S}^{(j)}_{F \Delta F \Delta} \psi = \nu \tilde{S}^{(i)}_{F \Delta F \Delta} : \tilde{S}^{(j)}_{F \Delta F \Delta} \Psi,
\]

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

\[
\tilde{S}^{(i)}_{F \Delta F \Delta} = S^{(i)}_{F \Delta F \Delta} - S^{(i)\top}_{F'F} S^{(i)\top} S^{(i)}_{F'F},
\]

with \( F' = \Gamma_i \backslash F_\Delta \). The elements of the primal space are obtained as \( S^{(i)}_{F \Delta F \Delta} : \kappa^{(i)}_{F \Delta F \Delta} \Psi \), where \( \Psi \) 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, \( M^{-1}_F S \) is symmetric positive definite on the subspace \( \{ V_\Gamma \cap \ker( B_{0\Pi}') \} \times Q_0 \); the minimum eigenvalue is 1 and the maximum eigenvalue is bounded from above by \( ||D'||_{\bar{\Omega}_F} \), with the norm taken on \( V_\Delta \). In addition, we can algebraically construct a primal space \( V_\Pi \) such that

\[
\kappa(M^{-1}_F S) \leq C \nu_{tol}, \quad \forall \nu_{tol} \in [1, \infty),
\]

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 stabilization 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 continuous 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.](image)

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 possess 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 \( \nu_{tol} \) 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.
Table 1: Details of the meshes used in Section 5.1 and 5.2.

<table>
<thead>
<tr>
<th></th>
<th>Cube</th>
<th>Octa</th>
<th>CVT</th>
</tr>
</thead>
<tbody>
<tr>
<td>k</td>
<td>nel</td>
<td>dofs</td>
<td>nel</td>
</tr>
<tr>
<td>1</td>
<td>32768</td>
<td>435200</td>
<td>30375</td>
</tr>
<tr>
<td>2</td>
<td>13824</td>
<td>508032</td>
<td>9000</td>
</tr>
<tr>
<td>3</td>
<td>8000</td>
<td>612000</td>
<td>4608</td>
</tr>
</tbody>
</table>

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 $\nu_{tol}$. The number of selected primal constraints $N_{\Pi}$ increases as $\nu_{tol}$ 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

<table>
<thead>
<tr>
<th>proc</th>
<th>$S^p_{id}$</th>
<th>$T_{ass}$</th>
<th>$S_p$</th>
<th>$\nu_{tol} = 2$</th>
<th>$\nu_{tol} = \infty$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$N_{II}$</td>
<td>$N_{II}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$\it$</td>
<td>$T_{sol}$</td>
<td>$S_p$</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>30</td>
<td>-</td>
<td>104</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>15</td>
<td>2.0</td>
<td>400</td>
<td>26</td>
</tr>
<tr>
<td>16</td>
<td>4</td>
<td>9</td>
<td>3.3</td>
<td>860</td>
<td>62</td>
</tr>
<tr>
<td>32</td>
<td>8</td>
<td>4</td>
<td>7.5</td>
<td>1823</td>
<td>147</td>
</tr>
<tr>
<td>64</td>
<td>16</td>
<td>2</td>
<td>15.0</td>
<td>3479</td>
<td>768</td>
</tr>
<tr>
<td>128</td>
<td>32</td>
<td>1</td>
<td>30.0</td>
<td>6944</td>
<td>104</td>
</tr>
</tbody>
</table>

### Cube mesh with 13824 elements

<table>
<thead>
<tr>
<th>proc</th>
<th>$S^p_{id}$</th>
<th>$T_{ass}$</th>
<th>$S_p$</th>
<th>$\nu_{tol} = 2$</th>
<th>$\nu_{tol} = \infty$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$N_{II}$</td>
<td>$N_{II}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$\it$</td>
<td>$T_{sol}$</td>
<td>$S_p$</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>117</td>
<td>-</td>
<td>203</td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>64</td>
<td>1.8</td>
<td>797</td>
<td>24</td>
</tr>
<tr>
<td>16</td>
<td>4</td>
<td>34</td>
<td>3.4</td>
<td>2071</td>
<td>63</td>
</tr>
<tr>
<td>32</td>
<td>8</td>
<td>17</td>
<td>6.9</td>
<td>4291</td>
<td>146</td>
</tr>
<tr>
<td>64</td>
<td>16</td>
<td>9</td>
<td>13.0</td>
<td>8095</td>
<td>337</td>
</tr>
<tr>
<td>128</td>
<td>32</td>
<td>5</td>
<td>23.4</td>
<td>15398</td>
<td>730</td>
</tr>
</tbody>
</table>

### Cube mesh with 8000 elements

<table>
<thead>
<tr>
<th>proc</th>
<th>$S^p_{id}$</th>
<th>$T_{ass}$</th>
<th>$S_p$</th>
<th>$\nu_{tol} = 2$</th>
<th>$\nu_{tol} = \infty$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$N_{II}$</td>
<td>$N_{II}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$\it$</td>
<td>$T_{sol}$</td>
<td>$S_p$</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>491</td>
<td>-</td>
<td>375</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>254</td>
<td>1.9</td>
<td>748</td>
<td>16</td>
</tr>
<tr>
<td>16</td>
<td>4</td>
<td>138</td>
<td>3.6</td>
<td>1873</td>
<td>43</td>
</tr>
<tr>
<td>32</td>
<td>8</td>
<td>66</td>
<td>7.4</td>
<td>4103</td>
<td>99</td>
</tr>
<tr>
<td>64</td>
<td>16</td>
<td>33</td>
<td>14.9</td>
<td>6848</td>
<td>192</td>
</tr>
<tr>
<td>128</td>
<td>32</td>
<td>18</td>
<td>27.3</td>
<td>19312</td>
<td>509</td>
</tr>
</tbody>
</table>

Table 2: Test 2. Strong scalability, Cube meshes. proc:=number of processors; $S^p_{id}$:=ideal speedup; $T_{ass}$:=assembling time in seconds; $N_{II}$:=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 speedups 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
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 polyhedral 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
<table>
<thead>
<tr>
<th>nel</th>
<th>dofs</th>
<th>( v_{t, \text{tol}} = 2 )</th>
<th>( v_{t, \text{tol}} = \infty )</th>
<th>( N_H ) it</th>
<th>( N_H ) it</th>
<th>( \text{it} )</th>
<th>( \text{cond} )</th>
<th>( \text{it} )</th>
<th>( \text{cond} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>21052</td>
<td>292432</td>
<td>1460 11 1.88</td>
<td>124 23 6.42</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32768</td>
<td>435200</td>
<td>1823 11 1.87</td>
<td>147 22 5.95</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>46656</td>
<td>618192</td>
<td>1951 11 1.75</td>
<td>149 21 5.60</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>64000</td>
<td>846400</td>
<td>2229 11 1.84</td>
<td>148 22 6.26</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>85184</td>
<td>1124816</td>
<td>2445 11 1.88</td>
<td>151 22 6.84</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Cube mesh, \( k = 1 \), procs = 32**

<table>
<thead>
<tr>
<th>nel</th>
<th>dofs</th>
<th>( v_{t, \text{tol}} = 2 )</th>
<th>( v_{t, \text{tol}} = \infty )</th>
<th>( N_H ) it</th>
<th>( N_H ) it</th>
<th>( \text{it} )</th>
<th>( \text{cond} )</th>
<th>( \text{it} )</th>
<th>( \text{cond} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>576</td>
<td>9024</td>
<td>528 11 1.87</td>
<td>106 20 4.71</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4608</td>
<td>69888</td>
<td>920 11 1.88</td>
<td>104 21 5.53</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9000</td>
<td>135600</td>
<td>1366 11 1.84</td>
<td>130 26 7.14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15552</td>
<td>233280</td>
<td>1544 11 1.86</td>
<td>134 26 6.93</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30375</td>
<td>453600</td>
<td>2133 11 1.96</td>
<td>144 27 7.03</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Cube mesh, \( k = 2 \), procs = 32**

<table>
<thead>
<tr>
<th>nel</th>
<th>dofs</th>
<th>( v_{t, \text{tol}} = 2 )</th>
<th>( v_{t, \text{tol}} = \infty )</th>
<th>( N_H ) it</th>
<th>( N_H ) it</th>
<th>( \text{it} )</th>
<th>( \text{cond} )</th>
<th>( \text{it} )</th>
<th>( \text{cond} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>40320</td>
<td>2148 11 1.88</td>
<td>107 115 171.63</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1728</td>
<td>133920</td>
<td>3178 12 2.00</td>
<td>107 93 97.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4096</td>
<td>314880</td>
<td>3623 12 1.99</td>
<td>106 89 93.50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8000</td>
<td>612000</td>
<td>4103 11 1.95</td>
<td>99 86 85.86</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13824</td>
<td>1054080</td>
<td>7028 12 2.01</td>
<td>146 131 648.41</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Cube mesh, \( k = 3 \), procs = 32**

<table>
<thead>
<tr>
<th>nel</th>
<th>dofs</th>
<th>( v_{t, \text{tol}} = 2 )</th>
<th>( v_{t, \text{tol}} = \infty )</th>
<th>( N_H ) it</th>
<th>( N_H ) it</th>
<th>( \text{it} )</th>
<th>( \text{cond} )</th>
<th>( \text{it} )</th>
<th>( \text{cond} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>72</td>
<td>6280</td>
<td>1309 10 1.73</td>
<td>109 228 1210.27</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>576</td>
<td>48320</td>
<td>2089 12 1.95</td>
<td>106 123 204.54</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4608</td>
<td>378880</td>
<td>3609 13 2.00</td>
<td>104 90 87.50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9000</td>
<td>737000</td>
<td>6608 12 2.19</td>
<td>130 159 552.21</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15552</td>
<td>1270080</td>
<td>7175 12 2.04</td>
<td>134 112 255.88</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>nel</th>
<th>dofs</th>
<th>( v_{t, \text{tol}} = 2 )</th>
<th>( v_{t, \text{tol}} = \infty )</th>
<th>( N_H ) it</th>
<th>( N_H ) it</th>
<th>( \text{it} )</th>
<th>( \text{cond} )</th>
<th>( \text{it} )</th>
<th>( \text{cond} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>576</td>
<td>9024</td>
<td>528 11 1.87</td>
<td>106 20 4.71</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4608</td>
<td>69888</td>
<td>920 11 1.88</td>
<td>104 21 5.53</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9000</td>
<td>135600</td>
<td>1366 11 1.84</td>
<td>130 26 7.14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15552</td>
<td>233280</td>
<td>1544 11 1.86</td>
<td>134 26 6.93</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30375</td>
<td>453600</td>
<td>2133 11 1.96</td>
<td>144 27 7.03</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4: Test 3. Optimality with respect to the mesh size, **Octa** meshes. nel:=number of polyhedra; dofs:=degrees of freedom; \( N_H \):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
<table>
<thead>
<tr>
<th>nel</th>
<th>dofs</th>
<th>$\nu_{tol} = 2$</th>
<th>$\nu_{tol} = \infty$</th>
<th>$N_{\Pi}$</th>
<th>it</th>
<th>cond</th>
<th>$N_{\Pi}$</th>
<th>it</th>
<th>cond</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>23509</td>
<td>812 12</td>
<td>2.05</td>
<td>163</td>
<td>18</td>
<td>4.18</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2000</td>
<td>47513</td>
<td>938 12</td>
<td>2.03</td>
<td>157</td>
<td>19</td>
<td>4.97</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4000</td>
<td>95785</td>
<td>1018 12</td>
<td>2.04</td>
<td>158</td>
<td>18</td>
<td>4.27</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8000</td>
<td>192860</td>
<td>1170 12</td>
<td>2.06</td>
<td>160</td>
<td>19</td>
<td>4.59</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16000</td>
<td>388042</td>
<td>1207 12</td>
<td>2.00</td>
<td>147</td>
<td>19</td>
<td>4.60</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>nel</th>
<th>dofs</th>
<th>$\nu_{tol} = 2$</th>
<th>$\nu_{tol} = \infty$</th>
<th>$N_{\Pi}$</th>
<th>it</th>
<th>cond</th>
<th>$N_{\Pi}$</th>
<th>it</th>
<th>cond</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>57018</td>
<td>3230 10</td>
<td>2.00</td>
<td>163</td>
<td>40</td>
<td>22.99</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2000</td>
<td>115026</td>
<td>3978 10</td>
<td>1.89</td>
<td>157</td>
<td>38</td>
<td>19.70</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4000</td>
<td>231570</td>
<td>4966 10</td>
<td>1.96</td>
<td>158</td>
<td>35</td>
<td>14.18</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8000</td>
<td>465720</td>
<td>5998 11</td>
<td>1.92</td>
<td>159</td>
<td>34</td>
<td>16.12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16000</td>
<td>936084</td>
<td>7100 11</td>
<td>1.89</td>
<td>147</td>
<td>32</td>
<td>12.91</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>nel</th>
<th>dofs</th>
<th>$\nu_{tol} = 2$</th>
<th>$\nu_{tol} = \infty$</th>
<th>$N_{\Pi}$</th>
<th>it</th>
<th>cond</th>
<th>$N_{\Pi}$</th>
<th>it</th>
<th>cond</th>
</tr>
</thead>
<tbody>
<tr>
<td>125</td>
<td>13275</td>
<td>2811 12</td>
<td>2.15</td>
<td>165</td>
<td>146</td>
<td>303.36</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>110030</td>
<td>6810 12</td>
<td>2.03</td>
<td>163</td>
<td>101</td>
<td>141.87</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2000</td>
<td>221710</td>
<td>8540 11</td>
<td>1.96</td>
<td>157</td>
<td>97</td>
<td>150.03</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4000</td>
<td>445950</td>
<td>10729 11</td>
<td>1.90</td>
<td>158</td>
<td>91</td>
<td>119.28</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8000</td>
<td>896200</td>
<td>13091 11</td>
<td>1.98</td>
<td>157</td>
<td>103</td>
<td>118.69</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5: Test 3. Optimality with respect to the mesh size; CVT meshes. nel:=number of polyhedra; dofs:=degrees of freedom; $N_{\Pi}$:=number of primal constraints; it:=CG iterations; cond:=experimental condition number.

<table>
<thead>
<tr>
<th>k</th>
<th>dofs</th>
<th>$\nu_{tol} = 2$</th>
<th>$\nu_{tol} = \infty$</th>
<th>$N_{\Pi}$</th>
<th>it</th>
<th>cond</th>
<th>$N_{\Pi}$</th>
<th>it</th>
<th>cond</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>55552</td>
<td>866 11</td>
<td>1.92</td>
<td>106</td>
<td>20</td>
<td>5.02</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>152064</td>
<td>2258 11</td>
<td>1.76</td>
<td>106</td>
<td>31</td>
<td>11.98</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>314880</td>
<td>3623 12</td>
<td>1.99</td>
<td>106</td>
<td>89</td>
<td>93.50</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>560384</td>
<td>7086 11</td>
<td>1.92</td>
<td>106</td>
<td>382</td>
<td>2052.33</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>904960</td>
<td>9103 12</td>
<td>2.13</td>
<td>106</td>
<td>1353</td>
<td>28663.80</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>k</th>
<th>dofs</th>
<th>$\nu_{tol} = 2$</th>
<th>$\nu_{tol} = \infty$</th>
<th>$N_{\Pi}$</th>
<th>it</th>
<th>cond</th>
<th>$N_{\Pi}$</th>
<th>it</th>
<th>cond</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>9024</td>
<td>528 11</td>
<td>1.87</td>
<td>106</td>
<td>20</td>
<td>4.71</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>23808</td>
<td>1224 11</td>
<td>1.85</td>
<td>106</td>
<td>38</td>
<td>17.31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>48320</td>
<td>2089 12</td>
<td>1.95</td>
<td>106</td>
<td>123</td>
<td>204.54</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>84864</td>
<td>3133 11</td>
<td>1.86</td>
<td>106</td>
<td>510</td>
<td>4913.25</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>135744</td>
<td>4164 11</td>
<td>1.88</td>
<td>106</td>
<td>1754</td>
<td>62878.43</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

The number of primal constraints ($N_{\Pi}$) 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.
5.5. Test 5: robustness with respect to the stabilization parameter $\tau$

In this test, we investigate how the stabilization parameter $\tau$ 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 $\tau$ 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 $\tau$, 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 $\tau$. When employing the minimal primal space solver ($\nu_{tol} = \infty$), instead, the condition number and iterations counts are significantly affected from the value of $\tau$, being the choice $\nu = 1$ 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 $\nu$ of the bilinear form $a(\cdot, \cdot)$ 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 $\nu$, since both the

Table 7: Test 5. Robustness with respect to the stabilization parameter $\tau$. $N_{\Pi}$:=number of primal constraints; it:=CG iterations; cond:=experimental condition number.

<table>
<thead>
<tr>
<th>$\tau$</th>
<th>$\nu_{tol} = 5$</th>
<th>$\nu_{tol} = \infty$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$N_{\Pi}$</td>
<td>it</td>
</tr>
<tr>
<td>0.01</td>
<td>3128 19</td>
<td>4.52</td>
</tr>
<tr>
<td>1</td>
<td>147 19</td>
<td>4.60</td>
</tr>
<tr>
<td>100</td>
<td>4732 19</td>
<td>4.39</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\tau$</th>
<th>$\nu_{tol} = 5$</th>
<th>$\nu_{tol} = \infty$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$N_{\Pi}$</td>
<td>it</td>
</tr>
<tr>
<td>0.01</td>
<td>1184 21</td>
<td>5.56</td>
</tr>
<tr>
<td>100</td>
<td>1844 22</td>
<td>5.62</td>
</tr>
</tbody>
</table>

Table 8: Test 6. Robustness with respect to jumping coefficients.
CVT mesh, nel = 16000, dofs = 388042
k = 1, procs = 32

<table>
<thead>
<tr>
<th>$\nu_{red}$</th>
<th>$\nu_{tol} = 5$</th>
<th>$\nu_{tol} = \infty$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$N_\Pi$</td>
<td>it</td>
</tr>
<tr>
<td>1e-4</td>
<td>253</td>
<td>27</td>
</tr>
<tr>
<td>1e-2</td>
<td>226</td>
<td>22</td>
</tr>
<tr>
<td>1</td>
<td>147</td>
<td>19</td>
</tr>
<tr>
<td>1e+2</td>
<td>214</td>
<td>20</td>
</tr>
<tr>
<td>1e+4</td>
<td>237</td>
<td>20</td>
</tr>
</tbody>
</table>

CVT mesh, nel = 8000, dofs = 465720
k = 2, procs = 32

<table>
<thead>
<tr>
<th>$\nu_{red}$</th>
<th>$\nu_{tol} = 5$</th>
<th>$\nu_{tol} = \infty$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$N_\Pi$</td>
<td>it</td>
</tr>
<tr>
<td>1e-4</td>
<td>1614</td>
<td>26</td>
</tr>
<tr>
<td>1e-2</td>
<td>1582</td>
<td>25</td>
</tr>
<tr>
<td>1</td>
<td>1184</td>
<td>21</td>
</tr>
<tr>
<td>1e+2</td>
<td>1579</td>
<td>22</td>
</tr>
<tr>
<td>1e+4</td>
<td>1619</td>
<td>22</td>
</tr>
</tbody>
</table>

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

The 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 $\nu_{red}$. 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 $\nu_{red}$ is far from 1.

Cube meshes, procs = 64

<table>
<thead>
<tr>
<th>nel</th>
<th>k</th>
<th>dofs</th>
<th>Pardiso $T_{sol}$</th>
<th>MUMPS $T_{sol}$</th>
<th>Block-Schur $T_{sol}$</th>
<th>BDDC $\nu_{tol} = 5$ $T_{sol}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>85184</td>
<td>1</td>
<td>1124816</td>
<td>1397</td>
<td>479</td>
<td>79</td>
<td>64</td>
</tr>
<tr>
<td>32768</td>
<td>2</td>
<td>1198080</td>
<td>1708</td>
<td>621</td>
<td>155</td>
<td>32</td>
</tr>
<tr>
<td>13824</td>
<td>3</td>
<td>1054080</td>
<td>1358</td>
<td>325</td>
<td>745</td>
<td>85</td>
</tr>
</tbody>
</table>

Octa meshes, procs = 64

<table>
<thead>
<tr>
<th>nel</th>
<th>k</th>
<th>dofs</th>
<th>Pardiso $T_{sol}$</th>
<th>MUMPS $T_{sol}$</th>
<th>Block-Schur $T_{sol}$</th>
<th>BDDC $\nu_{tol} = 5$ $T_{sol}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>30375</td>
<td>1</td>
<td>435600</td>
<td>167</td>
<td>63</td>
<td>80</td>
<td>15</td>
</tr>
<tr>
<td>30375</td>
<td>2</td>
<td>1210950</td>
<td>1166</td>
<td>892</td>
<td>577</td>
<td>96</td>
</tr>
<tr>
<td>15552</td>
<td>3</td>
<td>1270080</td>
<td>1665</td>
<td>785</td>
<td>1468</td>
<td>201</td>
</tr>
</tbody>
</table>

CVT meshes, procs = 64

<table>
<thead>
<tr>
<th>nel</th>
<th>k</th>
<th>dofs</th>
<th>Pardiso $T_{sol}$</th>
<th>MUMPS $T_{sol}$</th>
<th>Block-Schur $T_{sol}$</th>
<th>BDDC $\nu_{tol} = 5$ $T_{sol}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>16000</td>
<td>1</td>
<td>388042</td>
<td>1916</td>
<td>742</td>
<td>112</td>
<td>11</td>
</tr>
<tr>
<td>16000</td>
<td>2</td>
<td>936084</td>
<td>14992</td>
<td>5926</td>
<td>690</td>
<td>70</td>
</tr>
<tr>
<td>8000</td>
<td>3</td>
<td>896200</td>
<td>15735</td>
<td>5028</td>
<td>3510</td>
<td>318</td>
</tr>
</tbody>
</table>

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 (Block-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

$$B_D = \begin{bmatrix} B_1 & 0 \\ 0 & B_2 \end{bmatrix},$$ (19)
where
\[
\begin{align*}
B_1^{-1} &= \text{diagonal preconditioner for } A, \text{ i.e. } B_1 = \text{diag}(A) \\
B_2^{-1} &= \text{exact solution of the approximate Schur complement } S_{\text{app}}
\end{align*}
\]
with \( S_{\text{app}} = -B \text{diag}(A)^{-1}B^T \). The inversion of \( S_{\text{app}} \) 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 \( \nu_{\text{tol}} = 5 \). 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 \( \nu_{\text{tol}} \). Several parallel tests demonstrate 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


