



SCUOLA INTERNAZIONALE SUPERIORE DI STUDI AVANZATI

SISSA Digital Library

An agglomeration-based multigrid solver for the discontinuous Galerkin discretization of cardiac electrophysiology

Original

An agglomeration-based multigrid solver for the discontinuous Galerkin discretization of cardiac electrophysiology / Feder, M., Africa, P.C.. - In: COMPUTER METHODS IN APPLIED MECHANICS AND ENGINEERING. - ISSN 0045-7825. - 462:(2026). [10.1016/j.cma.2026.119298]

Availability:

This version is available at: 20.500.11767/152570 since: 2026-08-17T07:28:19Z

Publisher:

Published

DOI:10.1016/j.cma.2026.119298

Terms of use:

Testo definito dall'ateneo relativo alle clausole di concessione d'uso

Publisher copyright

note finali coverpage

(Article begins on next page)

An agglomeration-based multigrid solver for the discontinuous Galerkin discretization of cardiac electrophysiology

Marco Feder^{a,*}, Pasquale Claudio Africa^b

^a*Numerical Analysis Group, University of Pisa, Largo B. Pontecorvo, 5, Pisa, 56126, Italy*

^b*mathLab, International School for Advanced Studies (SISSA), Via Bonomea, 265, Trieste, 34136, Italy*

Abstract

This work presents a novel agglomeration-based multilevel preconditioner designed to accelerate the convergence of iterative solvers for linear systems arising from the discontinuous Galerkin discretization of the monodomain model in cardiac electrophysiology. The proposed approach exploits general polytopic grids at coarser levels, obtained through the agglomeration of elements from an initial, potentially fine, mesh. By leveraging a robust and efficient agglomeration strategy, we construct a nested hierarchy of grids suitable for multilevel solver frameworks. The effectiveness and performance of the methodology are assessed through a series of numerical experiments on two- and three-dimensional domains, involving different ionic models and realistic unstructured geometries. The results demonstrate strong solver effectiveness and favorable scalability with respect to both the polynomial degree of the discretization and the number of levels selected in the multigrid preconditioner.

Keywords: agglomeration, preconditioning, iterative solvers, multilevel methods, discontinuous Galerkin methods, monodomain model, cardiac electrophysiology

1. Introduction

Computational modeling of the heart has been actively pursued as a tool for accelerating cardiovascular research. However, the clinical applicability of such models is constrained by the high complexity and computational cost, for instance when moving towards whole heart modeling and coupling different physics and scales. Hence, the development of numerical methods able to reduce the computing time while keeping accuracy becomes essential for speeding-up fundamental research and, ultimately, for translation of modeling into clinical practice [51, 74].

In this work, we focus specifically on cardiac electrophysiology models describing the electrical activation of the myocardium. The electrical activity of the heart is usually modeled through the bidomain or monodomain equations, which describe the propagation of the transmembrane potential across the heart tissue. Both models consist of an elliptic-parabolic system of nonlinear reaction-diffusion partial differential equations (PDEs). The mathematical analysis of the system has already been addressed in [51], and a wide strand of discretization approaches have been proposed during the years, for instance in [5, 11,

*Corresponding author.

Email addresses: marco.feder@dm.unipi.it (Marco Feder), pafrica@sissa.it (Pasquale Claudio Africa)

37, 58] and references therein. In order to take into account the electrochemical reactions that occur at a cellular level, these systems are coupled, through a non-linear reaction term, to a system of Ordinary Differential Equations (ODEs), modelling the inward and outward flow of ionic currents across the cell membrane [51]. During the years, a large variety of models have been developed, ranging from reduced models with only one or few unknowns such as the Rogers-McCulloch ionic model [76], or the FitzHugh and Nagumo model [50], up to the Bueno-Orovio and ten Tusscher-Panfilov ionic models [31, 81, 82]. Due to the quick upstroke of the action potential, which is caused by voltage-dependent sodium channels, a numerically robust calculation of the propagation of the wave across the tissue is well known to be computationally challenging; indeed, the rapid increase of the transmembrane potential over a few milliseconds results in a steep wave front in space, requiring high resolution in both temporal and spatial discretizations.

Overall, the resulting large-scale linear systems are prohibitive for direct solvers. In addition, the system matrix often turns out to be very ill-conditioned, leading to a deterioration in the convergence rate of standard iterative methods. Hence, the main challenge to be addressed is the reduction of the overall time-to-solution of the iterative method through the design of effective preconditioners [26].

The development of suitable preconditioners for such models is a vast and active area of research, and many approaches have been proposed over the years. Among them, we mention domain-decomposition techniques such as Balancing Domain Decomposition by Constraints (BDDC) [44] and Finite Element Tearing and Interconnecting (FETI) methods [61]. Extensions of these methodologies, along with their application to electrophysiology problems, have been extensively studied [1, 58, 70, 84]. A numerical study of AMG preconditioning applied to the bidomain model can be found in [72].

Traditional strategies exploit linear continuous elements on very fine computational grids, in order to capture the physical features of the solution. To cope with such complexity, higher order continuous elements or discontinuous Galerkin (DG) approaches have recently received more attention [5, 29, 57], thanks to their ability to capture sharp gradients. For these reasons, in this work we will consider a DG discretization of the monodomain model. Our main focus is the development of a preconditioner for accelerating the convergence of the large-scale linear system of equations.

For linear systems stemming from the discretization of elliptic operators, multigrid methods are widely recognized as one of the most efficient approaches that scale to large parallel computers. Standard geometric multigrid techniques require the construction of a hierarchy of meshes that is most often obtained by (either uniform or adaptive) refinement of a coarse initial mesh. However, in the case of very complex geometries, such as the ones stemming from realistic models, building such hierarchies is far from trivial, requiring to fallback to non-nested methods [27, 36, 49] or algebraic multigrid (AMG) solvers [83].

The use of polytopal shapes is attractive in this respect, as coarse grids can be simply generated by agglomerating mesh elements together and various multilevel approaches have been proposed in the literature [23, 30, 35, 41, 66, 68]. However, providing automated and flexible agglomeration strategies for polyhedral elements remains a challenging task which is subject of intense research [9, 10, 47]. Motivated by the good results observed in [47], we adopt the agglomeration algorithm developed therein to produce nested

sequences of agglomerated grids by exploiting R-trees [55], a spatial indexing data structure which excels at organizing geometric data through bounding boxes, particularly in contexts where performing spatial queries for large sets of geometric objects in a fast way is required. This approach is fully automated, robust, and dimension-independent. It is designed to be independent of the specific shape of the underlying mesh elements, which can indeed be hexahedral, simplicial, polytopic, and stem from unstructured geometries. Notably, such procedure is purely geometrical in that it depends exclusively on the initial mesh.

During the last few years, several multilevel strategies for methods posed on arbitrarily shaped elements have been investigated, such as in Virtual Element methods (VEM) [7, 8, 15, 73] and Hybrid High-Order (HHO) methods [42, 43]. For DG discretizations of Poisson problems posed on polytopal grids [33], both nested [13, 14] and non-nested [17] multigrid variants have been proposed and theoretically analyzed. Polytopal DG methods have also been recently applied in the field of brain electrophysiology [64, 65], which shares the same abstract formulation of the model considered here [78], and numerical investigations of non-overlapping Schwarz preconditioners have been tested in [77].

In this work, we propose a novel agglomeration-based multilevel preconditioner for the DG discretization of the monodomain model which exploits the flexibility given by polytopic elements in creating hierarchies of coarse grids, starting from given unstructured meshes of interest. To increase the computational efficiency of the proposed preconditioner, we combine our efficient agglomeration strategy with state-of-the-art matrix-free operator evaluation kernels on the finest level of the hierarchy, thanks to the tensor-product structure of the basis functions [63]. We compare the performance of our methodology with well-established preconditioners such as AMG and Block-Jacobi from the Trilinos library [56], showing the effectiveness of our preconditioner in terms of iteration counts of the preconditioned conjugate-gradient (PCG) solver and global time-to-solution. We validate our approach through several tests in two and three dimensions, employing different ionic models, polynomial degrees, and geometries of different complexity. AMG and Block-Jacobi are well known to be effective preconditioners and are a standard choice for discretizations with linear elements [38]. Our numerical results indicate the advantage of our approach in terms of both iteration counts and time-to-solution, especially for higher polynomial degrees. In addition, we assess the scalability of our preconditioner in a parallel computing environment, demonstrating its parallel efficiency. The experiments are carried out using the C++ software project POLYDEAL [48], based on the DEAL.II finite element library [20]. It provides building blocks for polytopal discontinuous DG methods and agglomeration-based multigrid, using Message Passing Interface (MPI) for distributed-memory parallelism.

The paper is structured as follows. In Section 2 we introduce the monodomain model for cardiac electrophysiology, and address its spatial and time discretization in Section 3. In Section 4 we recall the R-tree data structure and the algorithmic realization of our approach, which is used to build hierarchies of agglomerated grids. We discuss the details and setup of our preconditioner in Section 5. Finally, we present several numerical results in Section 6, and point to future developments in Section 7.

2. The mathematical model

2.1. Mathematical modeling of cardiac electrophysiology

The heart wall is organized into three layers: the thin inner endocardium, the thin outer epicardium, and the thick muscular middle layer, the myocardium. The myocardium is mainly formed by cardiomyocytes, which are specialized, striated muscle cells that are electrically excitable and drive the heart's mechanical function. When a cardiomyocyte receives an electrical stimulus, the electrochemical balance across its membrane is altered, setting off a chain of biochemical events that change the cell's *transmembrane potential* (the voltage difference between the intracellular and extracellular environments). This produces a rapid depolarization followed by a slower repolarization. These voltage changes are governed by the opening and closing of voltage-gated ion channels, which selectively allow ions such as sodium, potassium and calcium to cross the membrane. Ionic currents change the membrane potential, while the membrane potential in turn controls those currents. Neighboring cardiomyocytes are electrically coupled by gap junctions, i.e. low-resistance intercellular channels, which permit the excitation to propagate from cell to cell throughout the myocardium.

The mathematical description of these processes is made of two building blocks: a *ionic model*, describing the chemical processes taking place at the cellular scale, and an *action potential propagation model*, representing the spatial propagation of the action potential wavefront at the tissue scale. We will introduce both components in the following subsections.

2.2. The monodomain model

We start by fixing some notation. Given an open and bounded domain D , we denote by $L^2(D)$ the space of square integrable functions on D , endowed with the norm $\|\cdot\|_{0,D}$ induced by the inner product $(\cdot, \cdot)_D$. We use the standard notation $H^s(D)$, with associated norm $\|\cdot\|_{s,D}$, to indicate Sobolev spaces $W^{s,2}(D)$. Vector-valued functions are indicated in boldface.

In what follows we briefly introduce the monodomain model for cardiac electrophysiology. For a more comprehensive introduction to the topic, we refer the interested reader to [51, 74]. Given an open, bounded domain $\Omega \subset \mathbb{R}^d$ ($d = 2, 3$) and a final time $T > 0$, we introduce the transmembrane potential $u(\mathbf{x}, t): \Omega \times [0, T] \rightarrow \mathbb{R}$, and the vector valued field $\mathbf{w}(\mathbf{x}, t): \Omega \times [0, T] \rightarrow \mathbb{R}^S$, where S is the number of ionic variables involved in the ionic model at hand. The *monodomain model* then reads [51]:

$$\begin{cases} \chi_m C_m \frac{\partial u}{\partial t}(t) - \nabla \cdot (\mathbb{D} \nabla u) + \chi_m \mathcal{I}_{\text{ion}}(u(t), \mathbf{w}(t)) = \mathcal{I}_{\text{app}}(\mathbf{x}, t), & \text{in } \Omega \times (0, T], \\ \frac{\partial \mathbf{w}(t)}{\partial t} = \mathbf{H}(u(t), \mathbf{w}(t)), & \text{in } \Omega \times (0, T], \\ \mathbb{D} \nabla u \cdot \mathbf{n} = 0, & \text{on } \partial\Omega \times (0, T], \\ u(0) = u_0, \quad \mathbf{w}(0) = \mathbf{w}_0, & \text{in } \Omega. \end{cases} \quad (1)$$

The conductivity tensor $\mathbb{D}$ is defined as:

$$\mathbb{D} := \sigma_l \mathbf{f}_0 \otimes \mathbf{f}_0 + \sigma_t \mathbf{s}_0 \otimes \mathbf{s}_0 + \sigma_n \mathbf{n}_0 \otimes \mathbf{n}_0,$$

where the vector fields $\mathbf{f}_0$, $\mathbf{s}_0$, and $\mathbf{n}_0$ express the fiber, the sheetlet, and the sheet-normal (cross-fiber) directions, respectively. The longitudinal, transversal, and normal conductivities are denoted by σ_l , σ_t , and σ_n , respectively. Notice that $\mathbf{f}_0, \mathbf{s}_0, \mathbf{n}_0 \in [L^\infty(\Omega)]^3$, and $\sigma_l, \sigma_t, \sigma_n > 0$ so that $\mathbb{D} \in [L^\infty(\Omega)]^{3 \times 3}$ is symmetric and positive definite almost anywhere in Ω . The generation of myocardial fibers has been performed using Laplace-Dirichlet Rule-Based Methods proposed and implemented in [3, 71]. The membrane cell capacitance is denoted by C_m , while χ_m is the membrane surface-to-volume ratio. The mathematical expression of the functions $\mathcal{I}_{\text{ion}}(u, \mathbf{w})$ and $\mathbf{H}(u, \mathbf{w})$ strictly depend on the choice of the ionic model, and will be specified later. The action potential is triggered by an external applied current $\mathcal{I}_{\text{app}}(\mathbf{x}, t)$, which mimics the presence of a (natural or artificial) pacemaker, while the $\mathcal{I}_{\text{ion}}(u, \mathbf{w})$ term is responsible for describing the electric current generated by the flux of ionic species across the cell membrane.

We close the system with suitable initial conditions u_0 , $\mathbf{w}_0$, and homogeneous Neumann boundary conditions on $\partial\Omega$ to model an electrically isolated domain. We denote by $\mathbf{n}$ the outward normal unit vector to the boundary.

2.3. Ionic models

Over the years, several ionic models have been proposed in the literature. Most of them are written as systems of ordinary differential equations (ODEs):

$$\begin{cases} \frac{\partial \mathbf{w}(t)}{\partial t} = \mathbf{H}(u(t), \mathbf{w}(t)), & t \in (0, T], \\ u(0) = u_0, \quad \mathbf{w}(0) = \mathbf{w}_0, \end{cases} \quad (2)$$

where the unknowns are the transmembrane potential $u = u(t)$ and the vector $\mathbf{w} = (w_1, \dots, w_S)$ collecting S ionic variables. The dynamic of the ionic variables is governed by the functions $\mathbf{H}(u, \mathbf{w})$ and $\mathcal{I}_{\text{ion}}(u, \mathbf{w})$ which couple the gating variables with the evolution of the action potential. In this work, we will investigate two phenomenological ionic models: the FitzHugh-Nagumo [50] and Bueno-Orovio [31] models, which we briefly describe hereafter.

2.3.1. FitzHugh-Nagumo

The FitzHugh-Nagumo model is described by the following scalar ODE:

$$\begin{cases} \frac{\partial w}{\partial t} = \varepsilon(u - \Gamma w) & t \in (0, T], \\ u(0) = u_0, w(0) = w_0. \end{cases} \quad (3)$$

The $\mathcal{I}_{\text{ion}}(u, w)$ term associated with this model has the expression:

$$\mathcal{I}_{\text{ion}}(u(t), w(t)) = ku(u - a)(u - 1) + w,$$

where $(k, a, \varepsilon, \Gamma)$ are known parameters to tune the ionic model. Given its simplicity, it will be used in the two-dimensional numerical experiments as a way to validate the proposed preconditioner.

2.3.2. Bueno-Orovio

The Bueno-Orovio model introduced in [31] has $S=3$ gating variables $\mathbf{w} = (w_0, w_1, w_2)$ and is described by the following system of ODEs:

$$\begin{cases} \frac{\partial w_0(t)}{\partial t} = [(b_0(u) - a_0(u))w_0 + a_0(u)w_0^\infty(u)] & t \in (0, T], \\ \frac{\partial w_1(t)}{\partial t} = [(b_1(u) - a_1(u))w_1 + a_1(u)w_1^\infty(u)] & t \in (0, T], \\ \frac{\partial w_2(t)}{\partial t} = [(b_2(u) - a_2(u))w_2 + a_2(u)w_2^\infty(u)] & t \in (0, T], \\ u(0) = u_0, \mathbf{w}(0) = \mathbf{w}_0, \end{cases} \quad (4)$$

where

$$\begin{aligned} a_0(u) &= \frac{1 - H_{V_1}(u)}{H_{V_1}^-(u)(\tau_1'' - \tau_1') + \tau_1'}, & a_1(u) &= \frac{1 - H_{V_2}(u)}{H_{V_2}^-(u)(\tau_2'' - \tau_2') + \tau_2'}, \\ a_2(u) &= \frac{1}{H_{V_2}(u)(\tau_3'' - \tau_3') + \tau_3'}, & b_0(u) &= -\frac{H_{V_1}(u)}{\tau_1^+}, \\ b_1(u) &= -\frac{H_{V_2}(u)}{\tau_2^+}, & b_2(u) &= 0, \\ w_0^\infty(u) &= 1 - H_{V_1}^-(u), & w_1^\infty(u) &= H_{V_o}(u)\left(w_\infty^* - 1 + \frac{u}{\tau_2^\infty}\right) + 1 - \frac{u}{\tau_2^\infty}, \\ w_2^\infty(u) &= H_{V_3}^{K_3}(u). \end{aligned}$$

The function $H_z^\varepsilon(z) := \frac{1 + \tanh(\varepsilon(z - z_0))}{2}$ describes a smooth approximation¹ of the Heaviside function depending on the constant parameter $\varepsilon \in \mathbb{R}^+$. With this model, the ionic term is given by:

$$\mathcal{I}_{\text{ion}}(u(t), \mathbf{w}(t)) = \sum_{q \in \{fi, so, si\}} I^q(u(t), \mathbf{w}(t)), \quad (5)$$

where

$$\begin{aligned} I^{fi} &= -\frac{H_{V_1}(u)(u - V_1)(\hat{V} - u)}{\tau_{fi}} w_0, & I^{si} &= -\frac{H_{V_2}(u)}{\tau_{si}} w_1 w_2, \\ I^{so} &= \frac{(1 - H_{V_2}(u))(u - V_o)}{H_{V_o}(u)(\tau_o'' - \tau_o') + \tau_o'} + \frac{H_{V_2}(u)}{H_{V_{so}}(u)(\tau_{so}'' - \tau_{so}') + \tau_{so}'} \end{aligned}$$

Albeit relatively simple, the Bueno-Orovio model has the characteristic of capturing the main features of the electrophysiology in healthy myocardial tissues [29, 53]. The parameter values used in the numerical experiments are listed in [Appendix A](#).

3. Space and time discretization

We can now introduce the weak formulation of Problem (1). To this aim, we consider the Sobolev space $V := H^1(\Omega)$, endowed with the classical H^1 -norm $\|\cdot\|_{1,\Omega}$. We assume

¹When ε is omitted, it corresponds to the classical Heaviside function.

that forcing terms, physical parameters, and initial conditions are sufficiently regular. The weak formulation then reads: find $u(t) \in V$, $\mathbf{w}(t) \in [V]^S$ such that for every $t \in (0, T]$:

$$\begin{cases} \chi_m C_m \left(\frac{\partial u(t)}{\partial t}, v \right)_\Omega + a(u(t), v) + \chi_m (\mathcal{I}_{\text{ion}}(u(t), \mathbf{w}(t)), v)_\Omega = (\mathcal{I}_{\text{app}}(\mathbf{x}, t), v)_\Omega & \forall v \in V, \\ \left(\frac{\partial \mathbf{w}(t)}{\partial t}, \boldsymbol{\psi} \right)_\Omega = (\mathbf{H}(u(t), \mathbf{w}(t)), \boldsymbol{\psi})_\Omega & \forall \boldsymbol{\psi} \in [V]^S, \\ u(0) = u_0, \quad \mathbf{w}(0) = \mathbf{w}_0 & \text{in } \Omega, \end{cases}$$

where the bilinear form $a(\cdot, \cdot)$ is defined as

$$a(u, v) = \int_\Omega \mathbb{D} \nabla u \cdot \nabla v \, d\mathbf{x}. \quad (6)$$

Notice that the right-hand side of the equation involves only the external current term since homogeneous Neumann boundary conditions are imposed on $\partial\Omega$.

The spatial discretization of Problem (1) is performed with a standard discontinuous Galerkin formulation [75]. Let $\mathcal{T}_h$ be a shape-regular triangulation of Ω consisting of disjoint elements K . We will assume that $\mathcal{T}_h$ is either a quadrilateral or hexahedral mesh. Interfaces are defined as the intersection of the $(d-1)$ -dimensional facets of neighboring elements. Further, we denote by $\mathcal{F}_h^I$ the union of all interior faces contained within Ω and by $\mathcal{F}_h^B$ those lying on the boundary $\partial\Omega$. We define $\mathcal{Q}_p(K)$ as the space of tensor-product polynomials over the element K of degree at most $p \geq 1$ in each coordinate direction, and the discontinuous finite element space over the mesh $\mathcal{T}_h$ as:

$$V_h^p = \{v_h \in L^2(\Omega) : v_h|_K \in \mathcal{Q}_p(K), K \in \mathcal{T}_h\}.$$

Let $F \in \mathcal{F}_h^I$ be the common face between two neighboring elements K^+ and K^- , and let $\mathbf{n}^+$ and $\mathbf{n}^-$ be the outward unit normal vectors, respectively. For regular enough scalar-valued function v and a vector-valued function $\mathbf{q}$, we define the average and jump operators as follows [21]:

$$\begin{aligned} \llbracket v \rrbracket &= \frac{1}{2}(v^+ + v^-), & \llbracket v \rrbracket &= v^+ \mathbf{n}^+ + v^- \mathbf{n}^-, & \text{on } F \in \mathcal{F}_h^I, \\ \llbracket \mathbf{q} \rrbracket &= \frac{1}{2}(\mathbf{q}^+ + \mathbf{q}^-), & \llbracket \mathbf{q} \rrbracket &= \mathbf{q}^+ \cdot \mathbf{n}^+ + \mathbf{q}^- \cdot \mathbf{n}^-, & \text{on } F \in \mathcal{F}_h^I. \end{aligned}$$

The superscripts $\pm$ indicate the traces taken from the interior of elements $K^\pm$. This setting enables us to introduce the following discrete bilinear forms $a_h(\cdot, \cdot) : V_h^p \times V_h^p \rightarrow \mathbb{R}$ and $m_h(\cdot, \cdot) : V_h^p \times V_h^p \rightarrow \mathbb{R}$:

$$\begin{aligned} a_h(u_h, v_h) &= \int_\Omega \mathbb{D} \nabla_h u_h \cdot \nabla_h v_h \, d\mathbf{x} - \sum_{F \in \mathcal{F}_h^I} \int_F \left(\llbracket \mathbb{D} \nabla_h u_h \rrbracket \cdot \llbracket v_h \rrbracket + \llbracket \mathbb{D} \nabla_h v_h \rrbracket \cdot \llbracket u_h \rrbracket + \sigma \llbracket u_h \rrbracket \cdot \llbracket v_h \rrbracket \right) ds, \\ m_h(u_h, v_h) &= \int_\Omega u_h v_h \, d\mathbf{x}, \end{aligned}$$

where ∇_h denotes the broken gradient operator defined element-wise. The penalty parameter is defined edge-wise as $\sigma(\mathbf{x}) = \alpha \mathbf{n}^T \mathbb{D} \mathbf{n}(\mathbf{x}) > 0$, with $\alpha = \frac{p(p+1)|F|_{d-1}}{|K|_d}$, being $|\cdot|_d$ the d -dimensional Hausdorff measure of the corresponding geometric entity.

3.1. Semidiscrete and fully-discrete formulation

The semi-discrete formulation of Problem (1) then reads: for any $t \in (0, T]$, find $u_h(t) \in V_h^p$ and $\mathbf{w}_h(t) \in [V_h^p]^S$ such that

$$\begin{cases} \chi_m C_m \left(\frac{\partial u_h(t)}{\partial t}, v_h \right)_\Omega + a_h(u_h(t), v_h) + \chi_m \left(\mathcal{I}_{\text{ion}}(u_h(t), \mathbf{w}_h(t)), v_h \right)_\Omega = \left(\mathcal{I}_{\text{app}}(\mathbf{x}, t), v_h \right)_\Omega & \forall v_h \in V_h^p, \\ \frac{\partial \mathbf{w}_h(t)}{\partial t} = \mathbf{H}(u_h(t), \mathbf{w}_h(t)), \\ u_h(0) = u_{h,0}, \quad \mathbf{w}_h(0) = \mathbf{w}_{h,0} \end{cases} \quad \text{in } \Omega,$$

where $u_{h,0}$ and $\mathbf{w}_{h,0}$ are given initial conditions. Let N be the dimension of the discrete space V_h^p , and let $\{\phi_i\}_{i=1}^N$ be the basis functions of V_h^p based on tensor products of Lagrangian polynomials. Then, for all $l = 1, \dots, S$, we have the expansions $u_h(t) = \sum_{i=1}^N U_i(t) \phi_i$ and $w_h^l(t) = \sum_{i=1}^N W_{h,i}^l(t) \phi_i$. Here $\mathbf{U}(t) \in \mathbb{R}^N$ and $\mathbf{W}_{h,l}(t) \in \mathbb{R}^N$ are vectors holding the expansion coefficients of $u_h(t)$ and $w_h^l(t)$ with respect to the basis $\{\phi_i\}_{i=1}^N$ for each time instant $t \in (0, T]$. We can hence define the relevant matrices, forcing terms and vectors as follows:

$$\begin{aligned} (A)_{i,j} &= a_h(\phi_j, \phi_i), & (M)_{i,j} &= m_h(\phi_j, \phi_i), \\ (I_{\text{ion}}^h(u, \mathbf{w}))_i &= (\mathcal{I}_{\text{ion}}(u, \mathbf{w}), \phi_i)_\Omega, & (I_{\text{app}}^h)_i &= (\mathcal{I}_{\text{app}}, \phi_i)_\Omega. \end{aligned} \quad (7)$$

With this notation, the algebraic formulation of the semi-discrete problem reads:

$$\begin{cases} \chi_m C_m M \dot{\mathbf{U}}_h + A \mathbf{U}_h + \chi_m I_{\text{ion}}^h(\mathbf{U}_h, \mathbf{W}_h) = I_{\text{app}}^h(t), \\ \dot{\mathbf{W}}_h = \mathbf{H}(\mathbf{U}_h, \mathbf{W}_h), \\ \mathbf{U}_h(0) = \mathbf{U}_0, \\ \mathbf{W}_h(0) = \mathbf{W}_0. \end{cases} \quad (8)$$

To obtain the fully-discrete system, we partition the time interval $[0, T]$ into N_T uniform subintervals, each of length $\Delta t = t_{n+1} - t_n$, with $t_n = n\Delta t$ for $n = 0, \dots, N_T - 1$. We use the subscript n to indicate the approximation of $\mathbf{U}_h(t)$ and $\mathbf{W}_h(t)$ at time t_n . For the temporal discretization, we adopt a second-order Backward Differentiation Formula (BDF2). Given initial conditions $\mathbf{U}_0$ and $\mathbf{W}_0$, together with suitable initializations of $\mathbf{U}_1$ and $\mathbf{W}_1$, the discrete scheme is: for $n = 1, \dots, N_T - 1$, find $\mathbf{U}_{n+1}$ and $\mathbf{W}_{n+1}$ such that

$$\chi_m C_m M \frac{3\mathbf{U}_{n+1} - 4\mathbf{U}_n + \mathbf{U}_{n-1}}{2\Delta t} + A \mathbf{U}_{n+1} + \chi_m I_{\text{ion}}^h(\mathbf{U}^*, \mathbf{W}_{n+1}) = I_{\text{app},n+1}^h, \quad (9)$$

which yields an explicit formula for $\mathbf{U}_{n+1}$

$$\left(3\chi_m C_m \frac{M}{2\Delta t} + A \right) \mathbf{U}_{n+1} = I_{\text{app},n+1}^h - \chi_m I_{\text{ion}}^h(\mathbf{U}^*, \mathbf{W}_{n+1}) + \chi_m C_m \frac{M}{2\Delta t} (4\mathbf{U}_n - \mathbf{U}_{n-1}). \quad (10)$$

Setting $\mathbf{U}^* = \mathbf{U}_{n+1}$ yields a fully implicit BDF2 scheme. To overcome this issue, we use the extrapolation formula $\mathbf{U}^* = 2\mathbf{U}_n - \mathbf{U}_{n-1}$, which is second-order accurate with respect to Δt [5, 54]. Therefore, given the solution $(\mathbf{U}_n, \mathbf{W}_n)$ at time t_n , the solution at time t_{n+1} is computed as follows:

- Solve the ionic model for $\mathbf{W}_{n+1}$ at each degree of freedom,
- Plug the computed $\mathbf{W}_{n+1}$ in $I_{\text{ion}}(\mathbf{U}^*, \mathbf{W}_{n+1})$ into Equation (10),
- Solve the resulting linear system for $\mathbf{U}_{n+1}$.

Note that this scheme allows crucial gains in computational efficiency: the problem for $\mathbf{U}_{n+1}$ is linear and hence the system matrix $3\chi_m C_m \frac{M}{2\Delta t} + A$ and the relative preconditioner are assembled once and for all at the beginning of the computation.

As for the evaluation of the ionic term $I_{\text{ion}}^h(\mathbf{U}^*, \mathbf{W}_{n+1})$ in (10), we opt for the so-called Ionic Current Interpolation (ICI) method [69]: we evaluate the ionic term at the degrees of freedom, and then interpolate onto quadrature nodes on mesh elements while assembling local contributions for the right-hand side vector. This procedure is cheaper and requires less memory than solving the system of ODEs at each quadrature node to compute I_{ion}^h (a procedure also known as State Variable Interpolation). The system of ODEs for $\mathbf{W}_{n+1}$ is solved using a BDF2 scheme.

Since the system matrix is symmetric and positive definite, the resulting linear system is solved with the preconditioned conjugate gradient (PCG) method. The focus of the remainder of this work will be the development of a multilevel preconditioner which exploits an efficient coarsening procedure of the given mesh to accelerate the convergence of the PCG solver.

4. R-tree based agglomeration

Before introducing our multigrid preconditioner, we briefly recall the basic properties of the R-tree data structure originally proposed in [55] and discuss its application to construct agglomerated grids following our previous work [47].

R-trees are hierarchical data structures used for the dynamic² organization of collections of d -dimensional geometric objects, representing them by their *Minimum Bounding – axis aligned – d -dimensional Rectangle*, also denoted by MBR. The minimum bounding rectangle is often referred to as *bounding box*. An internal node of an R-tree consists of the MBR that bounds all its children. In particular, each internal node stores two data: a way of identifying a child node and the MBR of all entries within this child node. The actual geometric data are stored in the *leaves* of the tree, that is, in its terminal nodes.

The original R-tree is based solely on the minimization of the measure of each MBR. Several variants have been proposed, aimed at either improving performance or flexibility, generally depending on the domain of application. It is often desirable to minimize the *overlap* between MBRs. Indeed, the larger the overlap, the larger the number of paths to be processed during queries. Moreover, the smaller the overlap the closer the agglomerated element will conform to the corresponding bounding boxes, thus making the resulting agglomerated grid qualitatively close to rectangular. Among the several available variants, we adopt the R*-tree data structure designed in [25]. In the remainder of the paper, we

²Dynamic here means that no global reorganization is required upon insertion or deletion of new elements of the tree.

will not make any distinction between R-trees and R*-trees, as we always employ the latter.

In our implementation, we rely on the BOOST.GEOMETRY module supplied by the BOOST C++ libraries [28] for the construction and manipulation of R-trees. Its kernels are designed to be agnostic with respect to the number of space dimensions, coordinate systems, and types.

4.1. Agglomeration algorithm

Here we describe the construction of an agglomerated mesh starting from a given finite element mesh, together with the associated R-tree data structure. Let $\mathcal{T}_h$ be a given mesh and assume the associated R-tree of order (m, M) has been constructed. We denote by L the total number of levels of the R-tree and with N_l the set of nodes in level l , for $l \in \{0, \dots, L - 1\}$. Our agglomeration strategy depends on an input parameter $l \in \{0, \dots, L - 1\}$ which is the index of the level to be used to generate the final agglomerates. The basic idea consists of looping through the nodes N_l and, for each node $n \in N_l$, descending recursively its children until leaf nodes are reached³. These nodes share the same ancestor node n (on level l) and thus are *agglomerated*. We store such elements in a vector $v[n]$, indexed by the node n . This procedure is outlined in Algorithm 1 and Algorithm 2.

Algorithm 1 Creation of agglomerates.

Input: R-tree R of order (m, M)

Input: $l \in \{0, \dots, L - 1\}$ target level.

Output: v vector s.t. $v[n]$ stores leafs associated to node n

```

1: function COMPUTEAGGLOMERATES( $R, l$ )
2:   for node  $n$  in  $N_l$  do
3:      $v[n] \leftarrow \text{EXTRACTLEAVES}(l, n)$  ▷ Defined in Algorithm 2
4:   end for
5:   return  $v$ 
6: end function

```

³Such technique of traversing a tree data structure is also known as depth-first search (DFS).

Algorithm 2 Recursive extraction of leafs from a node n on level l .

Input: $l \in \{0, \dots, L - 1\}$ target level

Input: $n \in N_l$

Output: vector v containing leafs which share the ancestor node n .

```

1: function EXTRACTLEAFS( $l, n$ )
2:   if  $l = 0$  then
3:      $v$ .PUSH_BACK(GET_LEAFS( $n$ ))
4:   return  $v$ 
5:   end if
6:   for node  $ch_n$  in CHILDREN( $n$ ) do
7:      $v$ .PUSH_BACK(EXTRACTLEAFS( $l - 1, ch_n$ ))
8:   end for
9:   return  $v$ 
10: end function

```

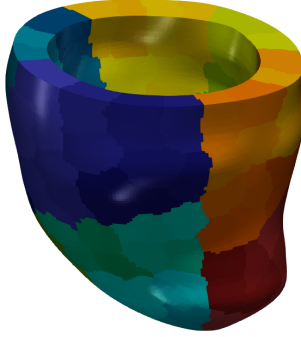
After the recursive visit of the children of a node n , a list of original mesh elements can be stored in the vector $v[n]$ and flagged appropriately for agglomeration. Since the R-tree data structure provides a spatial partition of mesh elements, each of which is uniquely associated with a node, the traversal of all nodes on a level l provides a partition of mesh elements into agglomerates. The overall agglomeration procedure can be summarized as follows:

- Compute $\{\text{MBR}(T_i)\}_i$ for $i = 1, \dots, \text{Card}(\mathcal{T}_h)$,
- Build the R-tree data structure using $\{\text{MBR}(T_i)\}_i$,
- Choose one level $l \in \{0, \dots, L - 1\}$ and apply Algorithm 1,
- For each node n , flag together mesh elements stored in $v[n]$.

In practice, elements in $v[n]$ are mesh iterators, i.e. lightweight objects such as pointers that uniquely identify elements of $\mathcal{T}_h$. The construction of the R-tree and the actual identification of agglomerates turns out to be a highly efficient operation also for large three-dimensional geometries. The interested reader is referred to [47] for a detailed analysis of the computational cost of the proposed agglomeration strategy and its application to several three-dimensional geometries. Notably, Algorithm 1 can be employed to generate sequences of *nested* agglomerated meshes for multilevel methods. As a consequence, such meshes can be used as levels of a hierarchy in a multigrid algorithm, allowing the usage of simpler and much cheaper intergrid transfer operators, when compared to non-nested variants; as a matter of fact, efficient intergrid transfer operators for non-matching meshes are far from trivial even for simple geometries and their construction constitutes a severe bottleneck in terms of computational efforts, becoming critical in 3D [17, 42]. In the context of Lagrangian finite elements on standard hexahedral or quadrilateral grids, a

completely parallel and matrix-free implementation of the non-nested geometric multigrid method has been recently addressed in [49].

We end this section by showing an example of agglomeration based on R-trees for a realistic 3D mesh of a left ventricle, made of 374,022 hexahedral elements, obtained from real personalized geometries [79]. In practical simulations, the computational workload is split among processes through a mesh partitioner such as PARMETIS [59]. Then, each MPI process owns a subdomain over which it performs local computations. A visualization of the MPI partitioning of the ventricle mesh into 128 subdomains is shown in Fig. 1a. Next, *within* each locally owned partition, each process generates agglomerates using the tree data structure outlined above. An example agglomerate element (on the boundary of the ventricle mesh) is shown in Fig. 1b, in grey. Its 8 sub-agglomerates, belonging to the next finer level of the hierarchy, are depicted in Figs. 1c and 1d. For better visualization, the sub-agglomerates, identified by different colors, are displayed in either of the two plots. We do not show results with agglomeration based on graph partitioning, but we refer the reader to [47] for several comparisons between the two approaches. A detailed analysis of different agglomeration strategies in the context of multilevel solvers can be found in [41].



(a) Hexahedral ventricle mesh partitioned among 128 MPI processes.

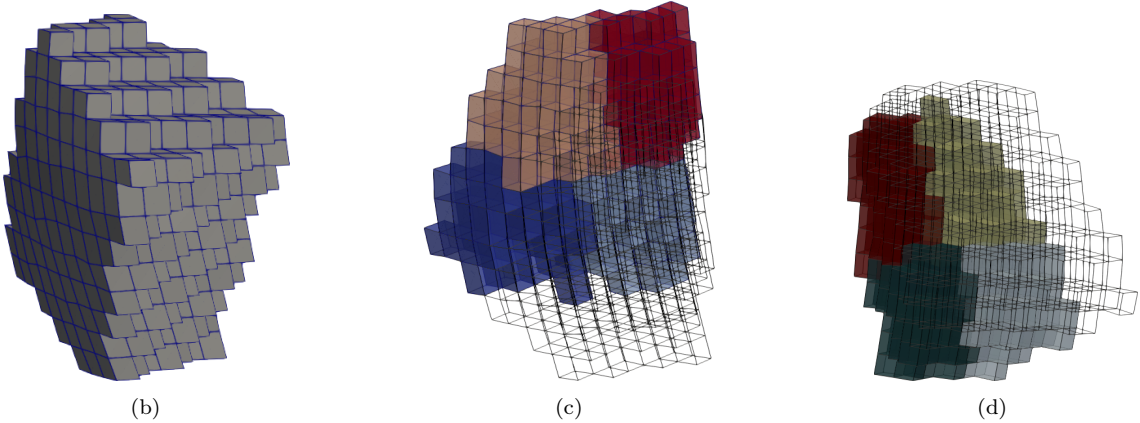


Figure 1: (a) MPI partitioning of the realistic hexahedral ventricle mesh into 128 subdomains. Each color corresponds to a different MPI process. Bottom row: detailed views of an agglomerate on the boundary of the domain (b) and its 8 sub-agglomerates (c) and (d), each displayed in a different color.

5. Agglomeration-based multilevel preconditioning

In this section, we propose an agglomeration-based multilevel preconditioner to accelerate the convergence of the PCG solver. Geometric multigrid methods typically start from a coarse mesh which is then repeatedly subdivided in finer meshes to generate a hierarchy of levels for which intergrid transfers are simple and fast. Hence, they hinder starting from a *predetermined* fine grid.

A popular and effective way to obviate this issue is to use AMG methods, which require only explicit knowledge of the system matrix, without relying on any geometrical information. Algebraic multigrid methodologies are widely employed for linear systems arising from scalar elliptic PDEs, and are renowned for their scalable performance [39, 46].

When applied to finite element discretizations, AMG preconditioners typically target matrices arising from linear continuous elements. In the discontinuous Galerkin setting, however, the redundancy of degrees of freedom associated with the same grid point is known to hinder the creation of aggregates. To address this issue, AMG solvers tailored to DG schemes were developed in [16, 24].

Here, instead, we assume a target fine mesh $\mathcal{T}_h$ over which we need to solve the problem to be given. Rather than uniformly refining $\mathcal{T}_h$, or feeding the system matrix to an AMG solver, we take a hybrid route and use the built-in flexibility of DG methods, in conjunction with the coarsening strategy developed in [47], to build a sequence of *nested* meshes and *coarser* operators that can be employed as level matrices in a multilevel method. The nestedness of the sublevels implies that transfer operators are much cheaper to compute compared to non-nested variants. All in all, the resulting multilevel preconditioner will be built in an algebraic manner, injecting geometric information delivered by the agglomeration routine. Therefore, we propose a method that is somewhat dual in that it can be interpreted both as algebraic and geometric.

The convergence properties of our approach rely on the theoretical framework developed in [18] for multigrid methods applied to Interior Penalty DG discretizations on meshes of agglomerated elements.

5.1. Inherited bilinear forms

Using the agglomeration procedure described in Section 4, we build a sequence of L nested meshes $\{\mathcal{T}_l\}_{l=0}^{L-1}$ with associated finite element spaces $\{V_l\}_{l=0}^{L-1}$. By construction, the original mesh and finite element space correspond to $\mathcal{T}_0 \equiv \mathcal{T}_h$ and $V_0 \equiv V_h^p$, respectively. We index the sequence of meshes and spaces so that to the lowest index corresponds the finest level. Since grids are nested, also the corresponding finite dimensional subspaces are nested:

$$V_{L-1} \subset V_{L-2} \subset \cdots \subset V_0.$$

On the finest grid $\mathcal{T}_0$, we consider the original bilinear form $\mathcal{A}_0 : V_0 \times V_0 \rightarrow \mathbb{R}$ defined by:

$$\mathcal{A}_0(u, v) := 3\chi_m C_m \frac{m_h(u, v)}{2\Delta t} + a_h(u, v) \quad \forall u, v \in V_0, \quad (11)$$

with matrix representation $A_0 := 3\chi_m C_m \frac{M}{2\Delta t} + A$, as introduced in Section 3.

On each bounding box B_K of an agglomerated polytopic element $K \in \mathcal{T}_l, l \in \{1, \dots, L-1\}$, the basis functions are constructed by defining on B_K the standard polynomial space $\mathcal{Q}^p(B_K)$. As $\overline{K} \subset \overline{B_K}$, the basis on K is defined by restricting each basis function to K . Other choices are possible, and we refer to the textbook [34] for an extensive discussion and relevant implementation details.

There are essentially two ways to build a sequence of coarser operators: the *inherited* approach [18], where the discrete operators are recursively built by restricting the fine grid operators, and the *non-inherited* one, where bilinear forms are explicitly assembled on each agglomerated level $\{\mathcal{T}_l\}_{l=1}^{L-1}$ of the hierarchy. Assembling bilinear forms on the coarser levels involves the computational burden of numerically integrating over agglomerated elements. To avoid such cost, we consider the inherited approach, where operators are defined iteratively from the restriction of the original bilinear form $\mathcal{A}_0$ as follows:

$$\mathcal{A}_{l+1}(u, v) := \mathcal{A}_l(\mathcal{P}_{l+1}^l u, \mathcal{P}_{l+1}^l v) \quad \forall u, v \in V_{l+1}, \quad (12)$$

with associated matrices:

$$A_{l+1} := (\mathcal{P}_{l+1}^l)^T A_l \mathcal{P}_{l+1}^l, \quad (13)$$

where $\mathcal{P}_{l+1}^l$ denotes the prolongation from the coarser level $l+1$ to the finer level l . As usual, the restriction operator $\mathcal{R}_l^{l+1}$ is defined as the adjoint of the prolongation. This technique for constructing coarser operators is one of the fundamental components of AMG methods and is commonly referred to as the *Galerkin projection* or *Galerkin triple product*. The inherited procedure is summarized in Algorithm 3.

From the implementation standpoint, we observe that each prolongation matrix $\mathcal{P}_{l+1}^l \in \mathbb{R}^{n_l \times n_{l+1}}$, where $n_l = \dim V_l$ and $n_{l+1} = \dim V_{l+1}$, is a distributed matrix whose parallel layout is easily defined thanks to nestedness of the partitions and the fact that in a DG discretization degrees of freedom are non-overlapping. Moreover, the simple block structure of the prolongation matrices allows to perform restriction and prolongation of vectors in a matrix-free fashion by using the local action of each block element-wise.

Avoiding assembling the coarser operators $\{A_l\}_{l=1}^{L-1}$ explicitly is quite attractive, as looping over all the polytopes on every level becomes increasingly expensive when agglomerated elements are composed of many sub-elements, particularly for high polynomial degrees, due to the large number of quadrature points. In principle, if homogeneous coefficients are considered, such cost could be dramatically reduced by using quadrature-free approaches for polytopic elements that reduce the integration over a polytope to only boundary evaluations [12]. Regardless of the quadrature approach, the original mesh $\mathcal{T}_h$ needs to be exploited to assemble all operators, with a huge gain in favor of the quadrature-free approach if it can be employed.

Another appealing feature of the inherited technique is its easiness to be added in existing finite element frameworks. In particular, its parallel implementation is simpler compared to the non-inherited approach. Indeed, in the non-inherited approach, flux computations on processor partition boundaries require accessing data from ghost agglomerated elements, which are composed of many layers of fine ghost cells that are not locally owned. With the inherited approach, instead, the polytopic shapes obtained by

agglomeration serve as a tool to build the sequence of prolongation operators $\{\mathcal{P}_{l+1}^l\}_{l \geq 0}$, which are then used to construct coarse-level matrices.

On the other hand, for multigrid schemes applied to DG discretizations of the Poisson problem on standard grids, it has been proven that only the non-inherited multigrid provides uniform convergence with respect to the number of levels [18]. Such behavior was verified also in [30] when using one V-cycle of agglomerated multigrid as a preconditioner, and subsequently fixed by adjusting the amount of stabilization in each level.

Remaining in the realm of agglomerated multigrid, we mention that in [47] the non-inherited variant applied to hierarchies generated by the agglomeration algorithm in Section 4 has been shown to work effectively, yielding uniform convergence with respect to the number of levels for several test cases. However, for the model considered in this work, numerical evidence shows that the number of iterations is not affected by the number of levels also when the inherited approach is used, cf. Section 6 for details.

We point out that, despite this work focuses on the monodomain model, we foresee the application of the proposed preconditioner to a variety of other symmetric and positive definite problems.

Algorithm 3 Construction of level operators $\{A_l\}_l$.

Input: Mesh $\mathcal{T}_h$.

Output: Sequence of level operators $\{A_l\}_l$.

```

1: function GENERATECOARSEROPERATORS
2:   READ_MESH( $\mathcal{T}_h$ )
3:    $A_0 \leftarrow$  ASSEMBLE_SYSTEM( $\mathcal{T}_h$ )
4:   BUILD_AGGLOMERATED_HIERARCHY( $\mathcal{T}_h$ )  $\triangleright$  Coarser levels  $\{\mathcal{T}_l\}_{l=1}^{L-1}$  (Section 4)
5:   for  $l \leftarrow 0$  to  $L - 1$  do
6:      $\mathcal{P}_{l+1}^l \leftarrow$  COMPUTE_PROLONGATION( $l, l + 1$ )
7:      $A_{l+1} \leftarrow (\mathcal{P}_{l+1}^l)^T A_l \mathcal{P}_{l+1}^l$   $\triangleright$  Galerkin product
8:   end for
9:   return  $\{A_l\}_l$ 
10: end function
```

5.2. Matrix-free operator evaluation

Within classical Krylov methods, the main computational kernel is the evaluation of the action of the operator on a vector. On current hardware architectures, this is rooted in high memory traffic from loading matrix entries into compute units performing arithmetic operations. For polynomial degrees $p \geq 2$, the increased bandwidth of the system matrix leads to a significant reduction in the throughput per degree of freedom. Matrix-free algorithms, conversely, avoid storing the matrix representation of the operator in order to reduce memory traffic, typically yielding improved performance for $p \geq 2$. Hence, whenever the iterative solver needs to evaluate the action of the operator, the matrix-free approach evaluates the spatial integrals of the finite element discretization by numerical quadrature rules. State-of-the-art matrix-free implementations exploit

sum-factorization techniques for tensor-product basis functions [67] and parallelization via mesh partitioning [32]. When tensor-product shape functions are combined with tensor-product quadrature rules, this computation becomes highly efficient through sum-factorization, which transforms the standard matrix-based operator evaluation involving $(p+1)^{2d}$ operations for $(p+1)^d$ degrees of freedom with polynomial degree p in d spatial dimensions at $(p+1)^d$ quadrature points into a sequence of d one-dimensional operations, each requiring $(p+1)^{d+1}$ operations. For three-dimensional problems, this approach reduces the algorithmic complexity by two orders in p . Additionally, SIMD vectorization (Single Instruction Multiple Data) over batches of elements is employed to further enhance performance [62, 63].

Irrespective of the chosen preconditioner, the action of the finest operator $3\chi_m C_m \frac{M}{2\Delta t} + A$ within the PCG solver is computed using the matrix-free kernels available, while coarse-level operators $\{A_l\}_{l=1}^{L-1}$ are applied in a matrix-based way due to the loss of the tensor product structure of the basis functions on agglomerated elements. With reference to our agglomeration-based preconditioner, also the Chebyshev smoother exploits the efficient operator evaluation on the finest level. This hybrid approach exploits the reduced memory footprint of the matrix-free evaluation on the fine level, where the number of degrees of freedom is the largest, while keeping the computational cost of coarse-level operator applications low, as the size of the coarse problems is increasingly smaller. In summary, within the PCG solver, the action of the finest-level matrix $3\chi_m C_m \frac{M}{2\Delta t} + A$ on a vector is computed in a matrix-free fashion, while the application of the preconditioners relies on assembled matrices on coarser levels. The evaluation of the right-hand side in Equation (10) is performed in a matrix-free fashion in all numerical examples.

Notably, for the inherited method of Algorithm 3, the matrix $3\chi_m C_m \frac{M}{2\Delta t} + A$ is assembled only once to generate the coarse-level operators and is then discarded before the time stepping procedure begins. When AMG or Block-Jacobi are employed as preconditioners, the system matrix must be assembled and handed to the setup phase to construct either the multilevel hierarchy (AMG) or to build a local solver for each block (Block-Jacobi). All the numerical experiments presented in this work directly leverage the matrix-free infrastructure of the DEAL.II library for the efficient discontinuous Galerkin operator evaluation [63].

6. Numerical results

In this section, we investigate the capabilities of the agglomeration-based preconditioner introduced in Section 5 for different scenarios by varying dimensions, geometry, ionic models, and polynomial degrees. As with any multilevel method, our multigrid preconditioner is built upon a few key components: a hierarchy of meshes, intergrid transfer operators between levels, a smoother, a sequence of operators $\{A_l\}_{l=0}^{L-1}$, and a coarse-grid solver. The nestedness of the partitions imply that transfer operators are the natural embedding for prolongation and its adjoint for restriction, while the operators on each level are built through the inherited approach described in Section 5. The smoother chosen is a Chebyshev-accelerated Jacobi scheme [2] of degree 3, which uses the inverse of the matrix diagonal on each level (precomputed before solving) and the level operator

application for computing residuals in the Jacobi-type iteration. Optimal parameters are determined by an eigenvalue estimation based on Lanczos iterations. In our simulations, we use 3 pre- and post-smoothing steps. We compare our multilevel approach against the TRILINOSML implementation of AMG [52] and, in order to better position the proposed multigrid method, we also include in our comparison a Block-Jacobi preconditioner with ILU(0) solver on each block from TRILINOS. In this work we use multilevel methods as preconditioners, since it is well known that multigrid methods are more robust when used as preconditioners rather than solvers [80]. The preconditioned conjugate-gradient solver applies a single V-cycle (see e.g. [83]) of either the agglomerated multigrid or the algebraic multigrid preconditioner. The AMG preconditioner parameters, reported in Table A.6, are set according to best practices. All preconditioners are initialized, once and for all, before the time stepping begins.

For all tests, we terminate the PCG iterations when the absolute residual norm falls below 10^{-14} . At time step t^{n+1} , we use the solution from the previous time step t^n as initial guess for the PCG solver. Given the small time step size Δt used in our simulations, this initial guess already closely approximates the solution, requiring only few iterations to meet the convergence criterion. In the left ventricle simulations, we have verified that the norm of the starting residual of the linear system is approximately 10^{-6} throughout the entire time stepping process. Consequently, a stopping tolerance of 10^{-14} corresponds to a eight-order-of-magnitude reduction in the residual norm.

All the numerical experiments are performed using the C++ software project POLYDEAL [47, 48], which is built on top of the finite element library DEAL.II [19, 20]. It provides building blocks for solving PDEs using high-order discontinuous Galerkin methods on polytopic meshes. In particular, it supports the R-tree based agglomeration strategy and the agglomerated multigrid framework described in the previous section. It is memory distributed using the Message Passing Interface (MPI) standard for parallel computing, and exploits P4EST [32] and PARMETIS [60] for distributed mesh refinement and partitioning, while relying on TRILINOS [56] as main linear algebra backend. All our experiments are publicly available at the GitHub page of POLYDEAL [48], and have been run in parallel using different numbers of MPI processes. Detailed instructions to run example programs are provided in the repository.

6.1. Two-dimensional test case

We start with the following two-dimensional test, taken from [29], aiming at assessing both the validity of our implementation with a test case which exhibits both the depolarization and repolarization phases, as well as to test the proposed preconditioning strategy. We consider Problem (1), over $\Omega = (0, 1)^2$, coupled with the FitzHugh-Nagumo ionic model described by the ODE (3). The parameters of the ionic model are identical to the ones used in [29], which we report in Table A.9 for the sake of completeness. The unit square is refined 7 times, resulting in a fine mesh $\mathcal{T}_h$ consisting of 16,384 quadrilateral elements. The number of degrees of freedom for different polynomial degrees is reported in Table 1. The mesh $\mathcal{T}_h$ is then partitioned through the mesh partitioner PARMETIS [60], where each MPI process owns a subdomain made of contiguous elements. This implies that, in general, the resulting MPI partitions are not describing structured grids anymore,

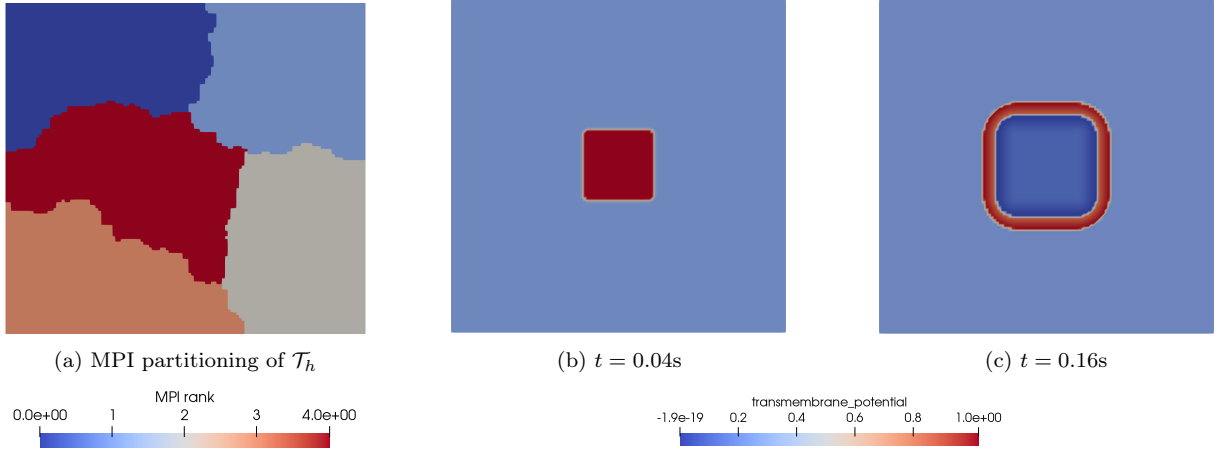


Figure 2: (a) The computational mesh partitioned using PARMETIS among five MPI processes. To build the multigrid hierarchy, aggregates are generated using the locally owned cells within each MPI rank. Snapshots of the transmembrane potential $\mathbf{U}$ at (b) $t = 0.04\text{s}$ and (c) $t = 0.16\text{s}$ after the external application of the current $\mathcal{I}_{\text{app}}(\mathbf{x}, t)$ in Equation (14).

as shown in Fig. 2a. As previously mentioned in Section 4.1, we generate agglomerates *within* each locally owned partition. As such, this test is representative of more general configurations in which unstructured geometries are considered. Indeed, since the original mesh is structured, the grids generated by the R-tree algorithm without partitioning the mesh among processes⁴ would be a structured hierarchy of coarser meshes.

The applied current is set to be:

$$\mathcal{I}_{\text{app}}(\mathbf{x}, t) = 2 \cdot 10^6 \mathbb{1}_{[0.4, 0.6]}(x) \mathbb{1}_{[0.4, 0.6]}(y) \mathbb{1}_{[0, 10^{-3}]}(t), \quad (14)$$

where $\mathbb{1}_{[a, b]}(\cdot)$ is the indicator function over the interval $[a, b]$. Such definition corresponds to a temporary electric shock localized in the square $[0.4, 0.6]^2$. We solve the problem until time $T = 0.4\text{s}$, with step size $\Delta t = 10^{-4}\text{s}$. The potential u and the gating variable w are set at rest.⁵ We show the activation and the subsequent propagation of the transmembrane potential u in Figs. 2b and 2c, obtained with polynomial degree $p = 2$. As shown, the wavefront is accurately captured by the DG discretization.

As coarse grid solver, we use the parallel sparse direct solver MUMPS [6]. The settings for the agglomerated multigrid preconditioner are shown in Table A.7. We report in Fig. 3 the number of outer iterations required by conjugate gradient for each time step, varying the polynomial degree p from 1 to 4. For all polynomial degrees, the agglomeration-based multilevel strategy achieves lower iterations counts compared to the AMG preconditioner, even for the lowest-order case $p = 1$, which is the most favorable for AMG solvers. For $p = 4$, we set the AMG aggregation threshold parameter to 0.1, as the default value of 0.2 failed to build a multigrid hierarchy. The Block-Jacobi preconditioner shows iteration counts similar to AMG, with an increase with respect to the polynomial degree p .

⁴Which means a serial run with 1 MPI process.

⁵Using the FitzHug-Nagumo model, $\mathbf{w}$ is a scalar unknown.

Polynomial degree p	DoFs
1	65,536
2	147,456
3	262,144
4	409,600

Table 1: Number of degrees of freedom for the two-dimensional test with polynomial degrees $p=1, \dots, 4$.

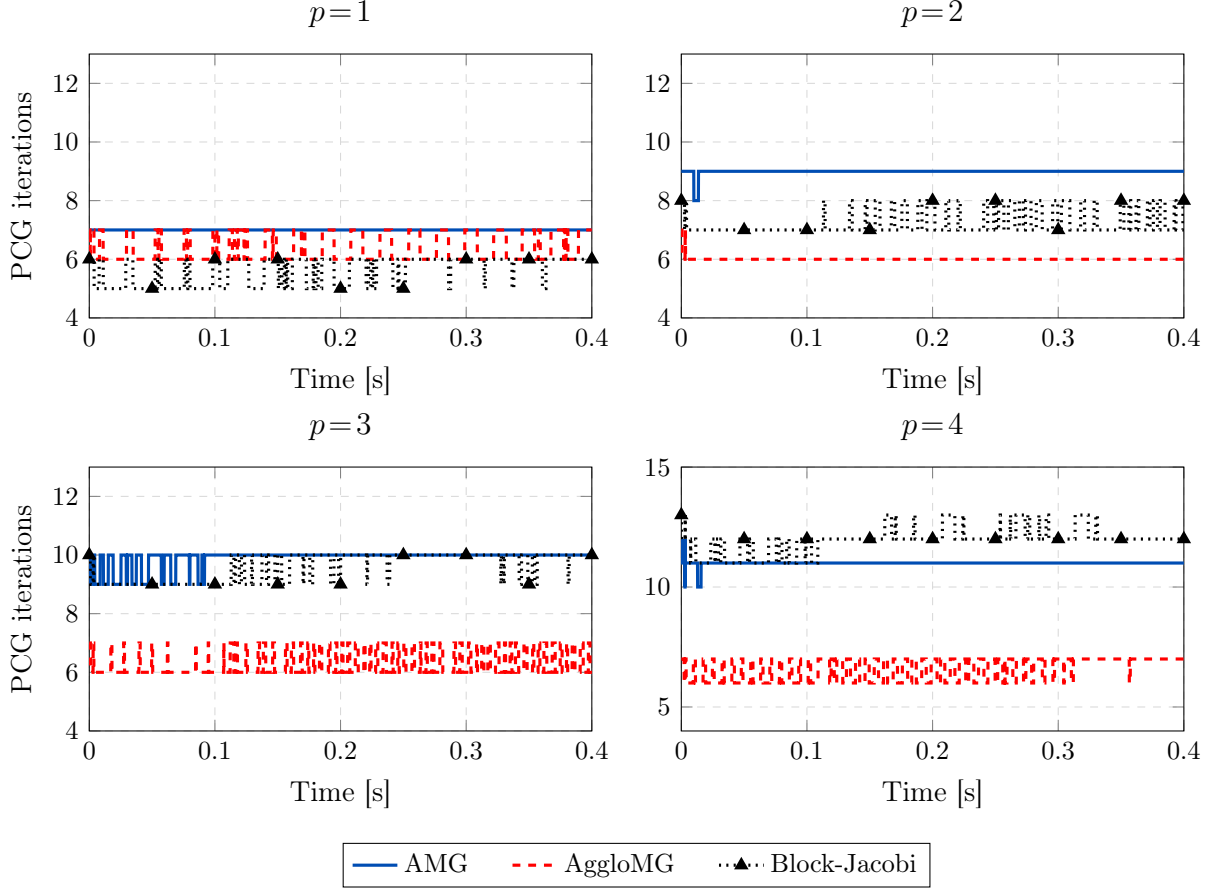


Figure 3: Number of PCG iterations per time step for Problem (1), comparing AMG, agglomerated multigrid (AggloMG), and Block-Jacobi for polynomial degrees $p = 1, 2, 3, 4$.

Overall, the agglomeration-based preconditioner displays a clear robustness with respect to the polynomial degree p , while the AMG and Block-Jacobi iterations are increasingly higher as p grows, doubling the number of iterations of the agglomeration-based preconditioner when $p = 4$. Conversely, the agglomeration-based preconditioner yields for every p a number of iterations that is essentially constant and ranging from 6 to 7, confirming the effectiveness of the proposed strategy.

6.2. Idealized three-dimensional test case

As outlined in Section 5, the main drawback of building the coarse operators through an inherited approach is that the number of iterations depends on the number of levels in the multigrid hierarchy [18]. For this reason, we investigate this dependence for the linear system of equations arising from Equation (10). To provide a meaningful test, we consider the idealized left ventricle shown in Fig. 4a, consisting of 407,904 hexahedral elements. The fiber field computed on this geometry is shown in Fig. 5a. Partitioning the mesh $\mathcal{T}_h$ among 5 processes, we obtain the levels with the cardinalities reported in Table 2. We solve the monodomain problem with different number of levels and polynomial degrees p from 1 to 3 in order to verify whether the number of PCG iterations is affected by the number of levels in the hierarchy. We use a time step $\Delta t = 10^{-4}$ s, solving Problem (1) until the final time $T = 0.4$ s. The settings for the agglomerated multigrid preconditioner are shown in Table A.8, while we use the Bueno-Orovio ionic model defined by the system of ODEs in (4), with parameters reported in Table A.10. The external stimulus $\mathcal{I}_{\text{app}}(\mathbf{x}, t)$ is localized at three points of the ventricle for 3ms with a magnitude of 300s^{-1} . As initial data we take $u = 0$ for the transmembrane potential, while the initial value for the gating variables are $w_0 = w_1 = 1$, and $w_2 = 0$. The transmembrane potential in millivolts is obtained by post-processing the numerical solution using the formula $u_{\text{mV}} = (85.7u - 84)\text{[mV]}$ [31]. We show in Fig. 4b the time evolution of the transmembrane potential at a selected point of the ventricle, where the typical depolarization and repolarization phases are clearly visible.

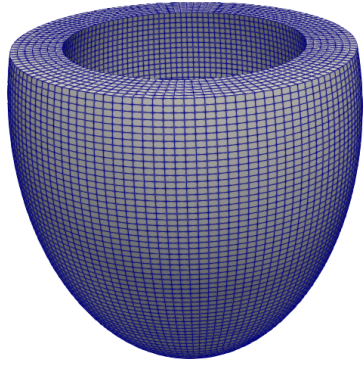
We report in Table 3 the statistics of the number of PCG iterations per time step for all the considered configurations in terms of polynomial degrees and number of levels. The mean values and the standard deviations confirm that for this problem the number of iterations is essentially independent of the number of levels employed. Notably, iteration counts are low and robust for all polynomial degrees.

This experimental observations are not in contradiction with the theoretical results in [18, 30], as here we are not solving a classical Poisson system, but rather the monodomain problem, where the presence of the dominant term $\frac{M}{\Delta t}$ shifts the spectrum of the system away from zero. We remark that the choice of Δt is dictated by the need to accurately capture the upstroke across the depolarization phase of the action potential, which typically requires small time steps [5, 38]. In particular, Δt is bounded from above by the stimulus duration (3ms in our case). We have further verified (Table A.11) that the number of PCG iterations increases with Δt for all the solvers considered (consistent with the improvement in conditioning as Δt decreases), the increase being most pronounced for Block-Jacobi. Across the admissible range of time steps and for all polynomial degrees, the agglomeration-based multigrid preconditioner remains robust and consistently yields the lowest iteration counts.

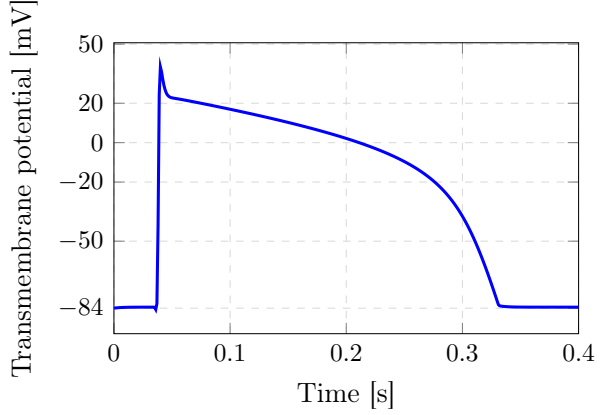
In the forthcoming section, we provide a comparison with AMG and Block-Jacobi on a three-dimensional test case with a realistic geometry.

6.3. Realistic three-dimensional test case

In the following experiment, we use the realistic CAD-modeled left-ventricle mesh $\mathcal{T}_h$ introduced in Section 4.1, together with the same model and solver parameters as in the



(a) Idealized mesh of the left ventricle.



(b) Evolution of the transmembrane potential at a point.

Figure 4: Idealized left ventricle test case: (a) hexahedral mesh of the ellipsoid and (b) time evolution of the transmembrane potential at a selected point.

Idealized ventricle (5 processes)	
Level index l	Card($\mathcal{T}_l$)
$l = 0$	407,904
$l = 1$	50,990
$l = 2$	6,375
$l = 3$	800
$l = 4$	100

Table 2: Coarsened hierarchy $\{\mathcal{T}_l\}_{l=0}^4$ for the mesh representing the left ventricle, partitioning the original mesh across 5 processes.

p	DoFs	No. of Levels (L)	PCG Iterations			
			Mean	Std Dev	Min	Max
1	3,263,232	2	7.6	0.6	6	9
		3	7.5	0.6	6	9
		4	7.3	0.9	5	9
2	11,013,408	2	8.1	0.7	7	10
		3	8.1	0.7	7	10
		4	8.1	0.8	7	10
3	26,105,856	2	8.7	0.9	7	11
		3	8.5	1.0	7	11
		4	8.5	1.0	7	11

Table 3: Statistics (mean value, standard deviation, and min-max) of PCG iterations per time step for the idealized ventricle test case across different numbers of multigrid levels L and polynomial degrees p .

previous idealized test case, except for the conductivities which are set to $\sigma_l = 10 \times 10^{-5} [\text{m}^2 \text{s}^{-2}]$, $\sigma_t = 2 \times 10^{-5} [\text{m}^2 \text{s}^{-2}]$, and $\sigma_n = 2 \times 10^{-5} [\text{m}^2 \text{s}^{-2}]$. The fiber field computed on this mesh is shown in Fig. 5b.

A visualization of the mesh and of the propagation of the transmembrane potential $\mathbf{U}$ is shown in Figure 6. In our investigation, we consider polynomial degrees from $p=1$ to $p=3$.

The multigrid hierarchy has been generated in parallel using the agglomeration procedure outlined in Algorithm 1. The number of mesh elements and degrees of freedom (with $\mathcal{Q}^1$ elements) at each level $l \in \{0, \dots, L-1\}$ is shown in Table 5 when the full ventricle geometry is distributed among an increasing number of MPI processes. Notably,

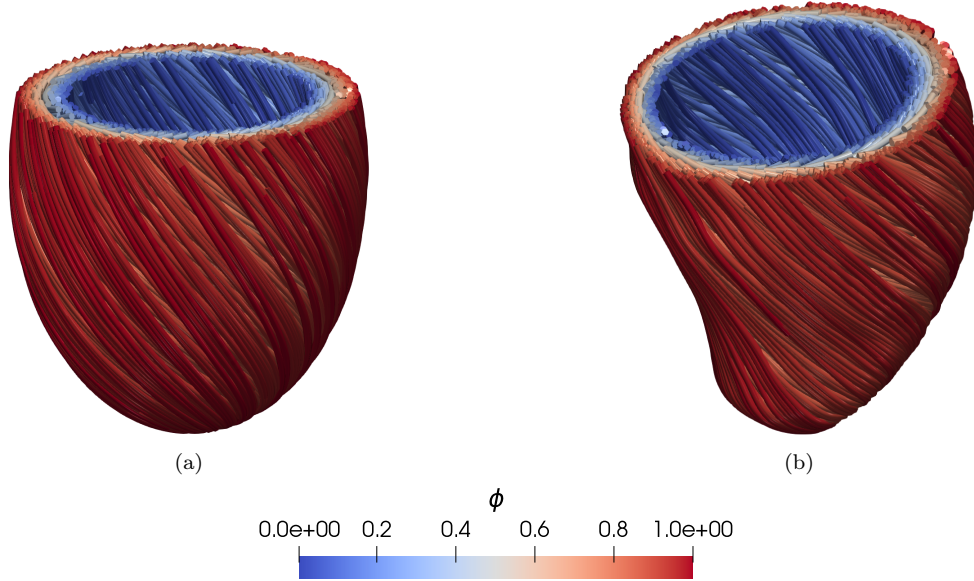


Figure 5: Fiber field computed using the Laplace-Dirichlet Rule-Based Methods [3, 71] visualized as streamlines: (a) Idealized left ventricle; (b) Realistic left ventricle. The solution ϕ is the transmural function, where $\phi=0$ at the endocardium and $\phi=1$ at the epicardium.

the decrease of the global number of elements between consecutive levels has a ratio very close to 8, which is the value one would obtain by halving the mesh step size of a uniform hexahedral grid. We observe again good balancing in terms of workload per process and ratios between the cardinality of consecutive grids. This is possible thanks to the combination of the graph partitioner PARMETIS, applied to the initial grid $\mathcal{T}_h$, and the balanced number of elements achieved by the agglomeration routine.

To ensure efficient multigrid convergence, the hierarchy must contain enough levels L such that the coarsest grid has an adequate size, preventing the coarse-level solve from becoming a bottleneck in the V-cycle. Direct solvers on the coarsest level pose scalability challenges in distributed-memory settings, as the resulting small system size leads to communication overhead that far exceeds computational work. One approach, implemented in HYPRE [46], restricts the coarse solve to a subset of MPI ranks while idling the remaining processes, followed by redistribution of the solution. For our three-dimensional experiments, we instead apply as coarse solver the PCG method with an algebraic multigrid V-cycle preconditioner, limiting the maximum size of the coarsest-level matrix to a sufficiently large size so that all processes are involved in the computation. Since PCG is employed as the coarse solver, our preconditioner no longer represents a fixed linear operator, which in principle calls for the *flexible* variant of the conjugate gradient method⁶.

We show some snapshots of the potential at selected time steps in Fig. 7, where the

⁶In practice, the classical conjugate gradient method required the same number of iterations, likely due to the rapid and accurate convergence of the coarse-level PCG iterations.

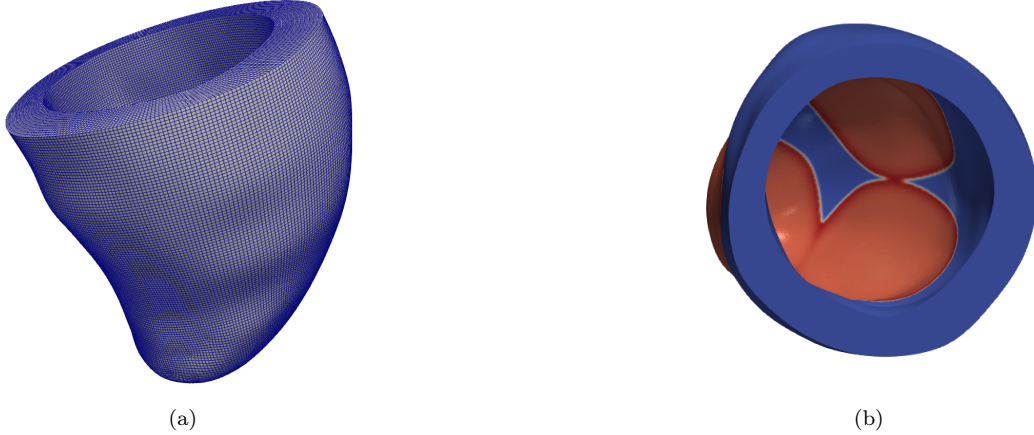


Figure 6: (a) Hexahedral mesh $\mathcal{T}_h$ representing a realistic left ventricle. (b) Propagation of the transmembrane potential U .

propagation of the steep wavefront is visible. We report the iterations' path per time step with linear, quadratic, and cubic polynomial degrees in Fig. 8. For all polynomial degrees and preconditioners, the number of PCG iterations have a maximum during the initial propagation of the potential, and a successive drop around $t = 0.12$ s, when the solution does not exhibit abrupt variations. The observed patterns follow what was observed in the two-dimensional case, where the polytopic approach achieves lower iterations counts than AMG and Block-Jacobi for all the polynomial degrees. In particular, for $p = 2$, the agglomerated multigrid preconditioner again exhibits robust behavior, with iteration counts oscillating between 6 and 11 for most of the simulation. When $p = 3$, iterations range from 8 to 13, far below the counts of AMG, which reach up to 21 iterations, and of Block-Jacobi, which reach up to 68 iterations.

6.4. Performance results on realistic three-dimensional test

We now examine the performance of all the considered preconditioners for the realistic three-dimensional ventricle test case. To this end, we do not only consider iteration counts as a metric, but also the wall-clock time spent in the PCG iterations, as well as the total time-to-solution of the whole simulation. The simulations have been run on the MareNostrum 5 cluster at Barcelona Supercomputing Center, where each compute node features two Intel Sapphire Rapids 8460Y+ processors (80 cores per node at 2.3GHz) and 512GB of main memory. All nodes have AVX-512 ISA extension, which allows 8 double-precision or 16 single-precision numbers to be processed per instruction when employing the matrix-free kernels detailed in Section 5.2. The timings reported in the forthcoming figures and tables are in seconds.

Since our approach builds the coarse operators through the inherited strategy, we quantify the memory footprint of the proposed multigrid hierarchies and the cost of V-cycle application, using as metrics the operator complexity, defined as:

$$C_{\text{op}} := \frac{\sum_{l=0}^{L-1} \text{nnz}(A_l)}{\text{nnz}(A_0)} > 1, \quad (15)$$

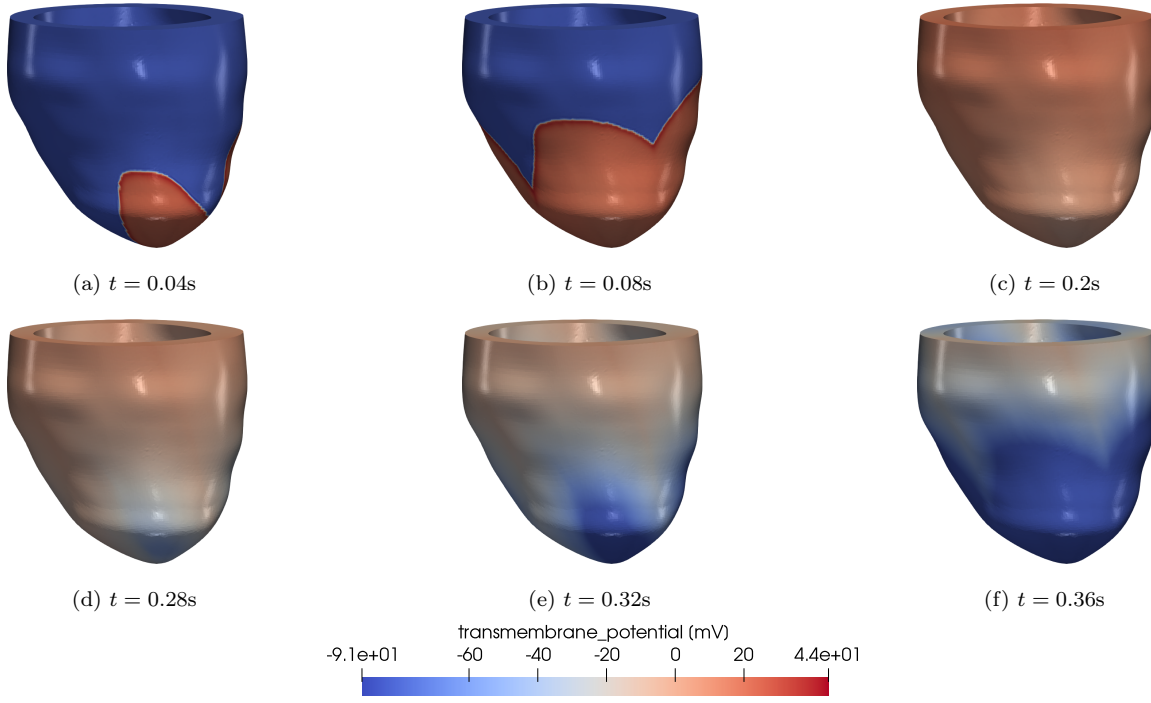


Figure 7: Snapshots of the transmembrane potential U at selected time steps after the external application of the current $\mathcal{I}_{\text{app}}(\mathbf{x}, t)$ at three locations inside the ventricle.

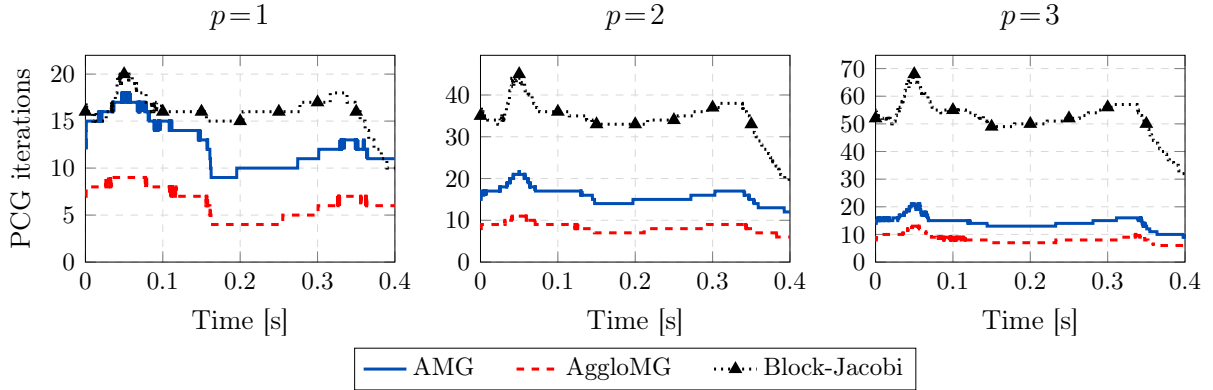


Figure 8: Number of PCG iterations per time step for Problem (1), comparing AMG, agglomerated multigrid (AggloMG), and Block-Jacobi for polynomial degrees $p=1, 2, 3$ for the three-dimensional ventricle test.

which is a customary measure considered in AMG methodologies [45]. In the definition above, $\text{nnz}(A_l)$ denotes the number of non-zero entries of the operator A_l at level l .

Fig. 9 displays the wall-clock time of each PCG iteration for all preconditioners when considering the whole simulation up to $T=0.4\text{s}$. The plots highlight the higher efficiency of the agglomeration-based multigrid preconditioner against the chosen AMG implementation. The agglomeration-based approach achieves consistently lower times for all poly-

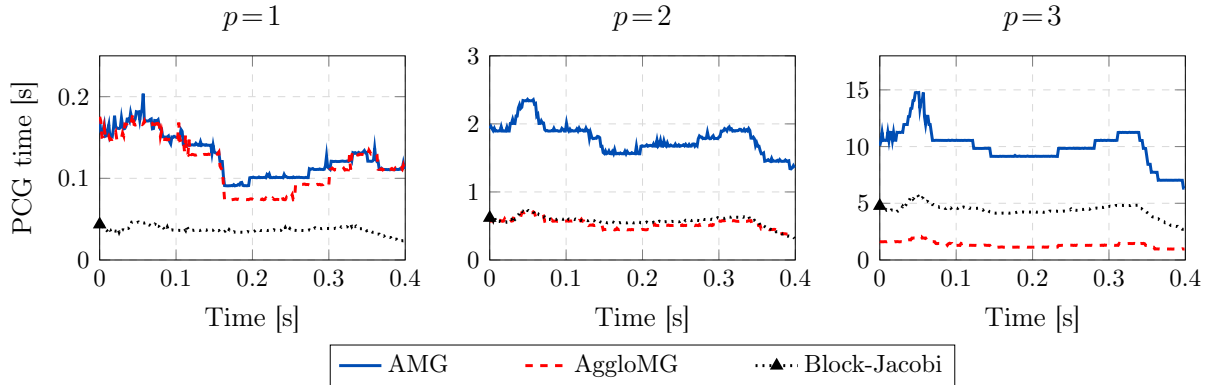


Figure 9: PCG iteration time per time step for Problem (1), comparing AMG and agglomerated multigrid (AggloMG) for polynomial degrees $p=1, 2, 3$ for the three-dimensional ventricle test.

nomial degrees, yielding compatible performance also with linear elements, where AMG is known to be very competitive. Despite the higher number of iterations, Block-Jacobi delivers the lowest total time-to-solution for linear elements, owing to its negligible per-iteration cost. With quadratic elements, the gap between agglomerated multigrid and AMG is even more significant. Block-Jacobi yields times which are slightly higher than agglomeration-based multigrid. When cubic elements are considered, the performance gap between agglomerated multigrid and Block-Jacobi is evident, with the latter showing significantly higher times, but better performance than AMG.

In Table 4, we report some statistics about the PCG iterations for all preconditioners and polynomial degrees. From there, it can be appreciated that with linear elements the time per PCG iteration is lower for the Block-Jacobi preconditioner, whereas AMG and agglomerated multigrid show higher times. However, the agglomeration-based multigrid preconditioner shows a speedup with respect to AMG and Block-Jacobi that increases with the polynomial degree p : when $p=2$ it reaches a speedup factor greater than 3 against AMG. When $p=3$, it achieves a factor of 3 against Block-Jacobi and a factor of roughly 7.5 against AMG, yielding a significant improvement in the overall time-to-solution of the whole simulation.

The performance of the preconditioners when employing either matrix-based or matrix-free operator evaluations within the Krylov solver is compared in Fig. 10. Specifically, we show the time to perform a single PCG iteration for polynomial degrees $p=1$, $p=2$, and $p=3$. In almost all instances, matrix-free evaluations (orange bars) improve the performance of the PCG iteration, compared to their matrix-based counterparts (green bars). When $p=1$, the speedup for agglomerated multigrid with matrix-free actions (instead of matrix-based ones) is moderate, while essentially no gain is observed for AMG and Block-Jacobi with matrix-free operator evaluations. Matrix-based variants show higher times, which are instead in favor of Block-Jacobi.

The gain is substantially higher when $p=2$, with the agglomeration-based multigrid reaching a speedup factor greater than 4 with respect to its matrix-based variant. The AMG and Block-Jacobi preconditioners with matrix-free operator evaluations show a

more pronounced speedup with respect to the matrix-based version. Such factors are, however, much smaller than the speedup obtained for the agglomeration-based multigrid. When $p=3$, the agglomeration-based multigrid with matrix-free actions outperforms its matrix-based variant by a factor of 9. It also shows a speedup greater than 3 compared to Block-Jacobi. Another observation that can be drawn from Fig. 10 is that the performance gap between the agglomeration-based multigrid (with matrix-free evaluations) and AMG (with matrix-based evaluations, used in a classical black-box fashion) is quite significant, with a speedup factor of 3.6 for $p=2$ and 7.8 for $p=3$. Comparing in the same way to simple Block-Jacobi with matrix-based operator evaluations, the speedup is 1.7 for $p=2$ and 5.8 for $p=3$.

Regarding the setup phase (column t_{setup} in Table 4), the chosen AMG and Block-Jacobi implementations show a globally lower cost than the agglomeration-based one. Besides, this cost is amortized during the time stepping procedure, where the preconditioner is recycled at each time step.

To better understand the composition of the setup phase for the agglomeration-based preconditioner, we report in Fig. 11 the breakdown of the setup cost into its main components: building the R-tree data structure from the fine mesh $\mathcal{T}_h$, traversing the tree hierarchy to create agglomerates, storing transfer matrices $\{\mathcal{P}_{l+1}^l\}_{l \geq 0}$, and computing the coarser operators $\{A_l\}_{l=1}^{L-1}$ through the Galerkin product in Equation (13). It is evident from the breakdown that the most expensive operation is the computation of the Galerkin triple product, which involves sparse matrix-matrix multiplications. This is particularly evident when increasing the polynomial degree to $p=2$ and $p=3$, with 80% or more of the total setup time being spent in this phase. The second most expensive operation is the computation and storage of the intergrid transfer operators, reaching almost 20% of the total setup time for $p=2$ and 10% for $p=3$. On the other hand, building the R-tree and creating agglomerates are totally inexpensive operations, taking together less than 1% of the total setup time. Notably, this computational kernel depends *only* on the geometry, i.e. the initial mesh $\mathcal{T}_h$, and not on the polynomial degree p .

We detail in Fig. 12 the dependence of all preconditioners on the number of MPI processes employed. We report, for increasing numbers of processes, the average number of PCG iterations⁷ per time step (left) and the operator complexity⁸ (right). The first row shows the results for $p=1$, while the second row for $p=2$. The number of PCG iterations for AMG and agglomeration-based multigrid is clearly stable with respect to the number of processes involved, with the agglomeration-based multigrid showing lower iteration counts. Iteration counts for Block-Jacobi show an expected increase with the number of processes due to its limited algorithmic scalability. The computed complexities for $p=1$ are essentially independent of the number of processes for both multilevel approaches, with AMG showing values very close to one, while the agglomeration-based multigrid shows a constant value of 1.2 and remains nicely bounded. Analogous behavior is observed for $p=2$: although iteration counts are higher than for $p=1$, both AMG and agglomeration-

⁷Averaged over the first twenty time steps.

⁸The operator complexity for AMG has been computed using the interface exposed by TRILINOSML.

based multigrid remain stable with respect to the number of processes, while Block-Jacobi shows again a slight increase. Operator complexities remain stable, though with a smaller gap; AMG shows slightly higher values for the operator complexity, around 1.2. Notably, the agglomeration-based multigrid is not affected by the variation in polynomial degree, showing again an essentially constant operator complexity of 1.2.

Finally, we perform a strong scaling test for the agglomeration-based multigrid preconditioner. We consider the realistic left ventricle test case with polynomial degree $p=1$, using a mesh $\mathcal{T}_h$ comprising 2,992,176 elements, which corresponds to 23,937,408 degrees of freedom. We run the simulation for twenty time steps with $\Delta t = 10^{-4}$ s, varying the number of MPI processes from 80 to 5,120 (i.e., from 1 to 64 compute nodes). Fig. 13 (top row) shows the strong scaling behavior of several components of the simulations. In particular, the linear solver exhibits ideal scaling up to 32 nodes, with efficiency remaining above 90% up to 32 nodes (2,560 processes), demonstrating the good parallel scalability of the proposed approach. Notice that at the highest number of processes (64 nodes, 5,120 processes), less than 10,000 degrees of freedom are assigned to each process, hence justifying the mild degradation due to communication overhead, which matches previous observations in the literature [22]. The other components of the simulation, such as assembly of the system matrix (performed only once), and right-hand side assembly (performed at every time-step), also show ideal scaling behavior. In the bottom row of Fig. 13, we repeat the experiment with $p=2$, corresponding to 80,788,752 degrees of freedom, from 4 up to 64 compute nodes. The linear solver shows almost ideal scaling up to 64 nodes, confirming the good scalability properties of the agglomeration-based multigrid preconditioner also for higher polynomial degrees.

All in all, we believe the series of experiments above shows the potential of the agglomeration-based preconditioner, particularly when comparing against highly optimized and widely used solvers. The high efficiency of our aggregation strategy suggests that it could be readily integrated into existing high-performance AMG frameworks (such as PSCTOOLKIT [40]), in order to provide *geometrically informed* AMG hierarchies. Nevertheless, our implementation can be optimized further: within the Galerkin triple product computation, and during the evaluation of coarser operators. We leave these improvements to future work. Despite these possible enhancements, we conclude this discussion by observing that the agglomeration-based preconditioner is an efficient and competitive solver both in terms of iteration counts and time-to-solution for the present problem, with its advantage over AMG and Block-Jacobi growing with the polynomial degree.

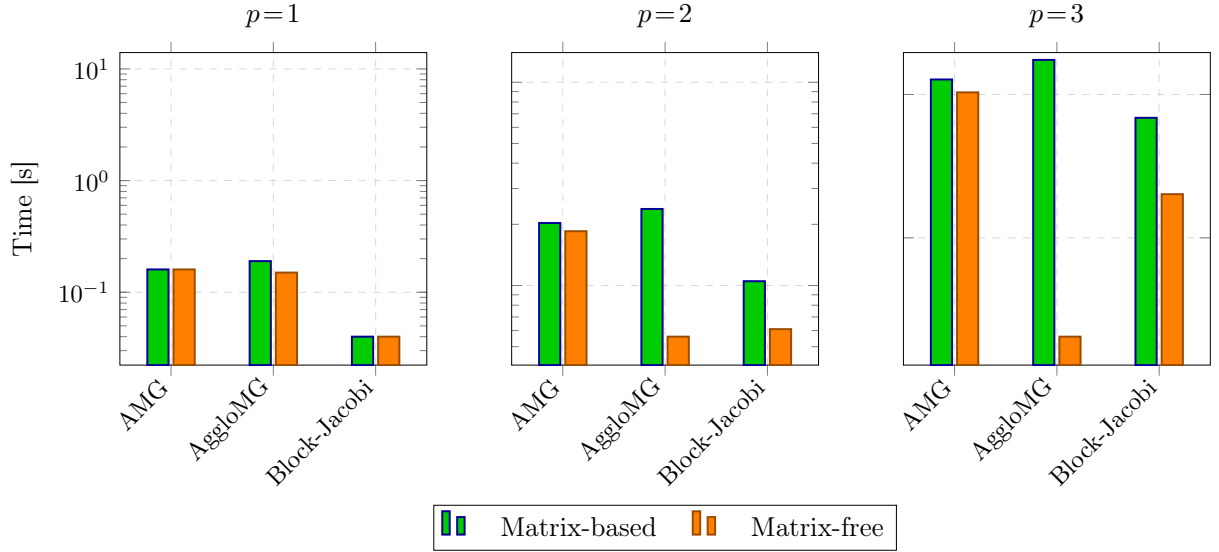


Figure 10: Comparison of average wall-clock time (measured in seconds) to perform one time step for polynomial degrees $p=1$, $p=2$, and $p=3$ on the realistic left ventricle mesh with 256 MPI processes. Blue bars represent matrix-based variants, orange bars represent results obtained with matrix-free evaluations of the finest operator.

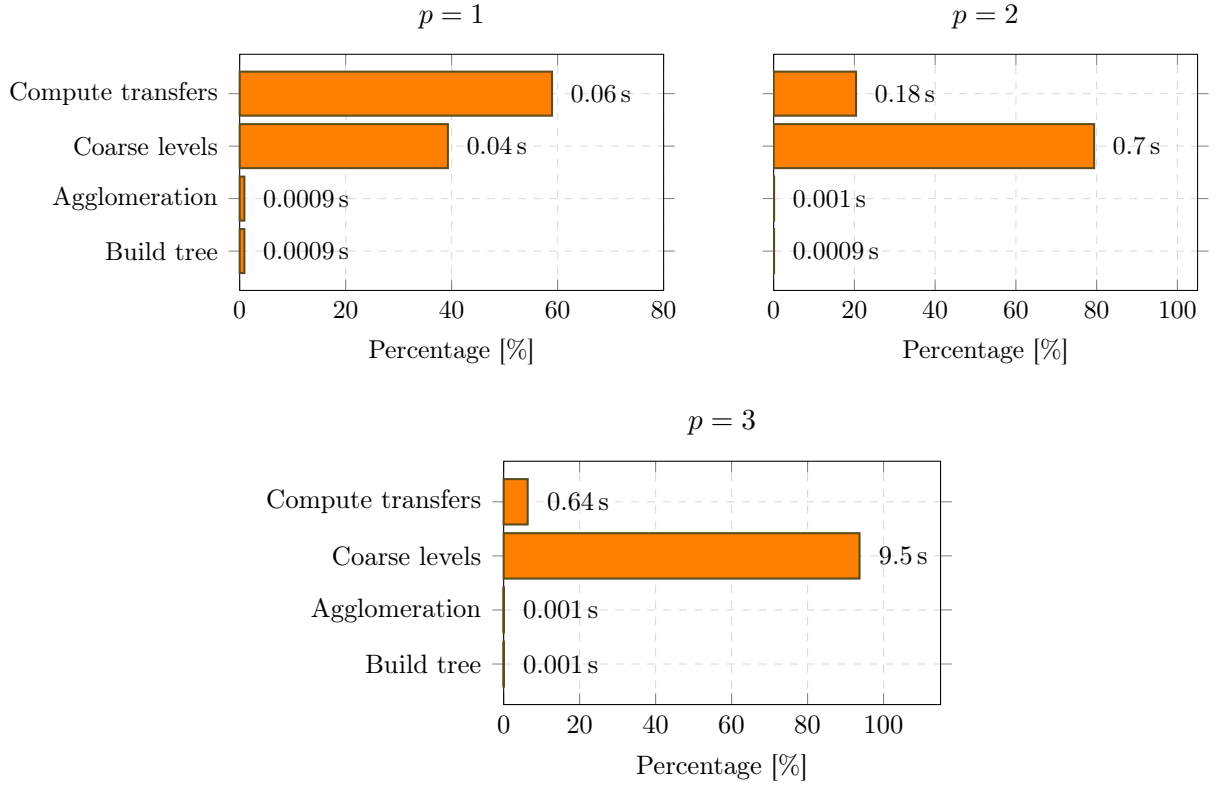


Figure 11: Breakdown of the setup phase cost for the agglomeration-based multigrid preconditioner on the three-dimensional ventricle test case with $p=1$ (top left), $p=2$ (top right), and $p=3$ (bottom) with 256 MPI processes. The percentages represent the relative time spent in each component.

p	Preconditioner	t_{setup} [s]	It.	t_{CG} [s]	t_{TOT} [s]	DoFs
1	AMG	0.11	13	0.12	5.68×10^2	2,992,176
	AggloMG	0.27	6	0.11	4.83×10^2	
	Block-Jacobi	0.05	16	0.03	1.60×10^2	
2	AMG	0.68	15	1.78	7.19×10^3	10,098,594
	AggloMG	1.07	8	0.52	2.14×10^3	
	Block-Jacobi	0.70	34	0.57	2.33×10^3	
3	AMG	4.36	14	10.0	4.01×10^4	23,937,408
	AggloMG	10.8	8	1.32	5.42×10^3	
	Block-Jacobi	5.9	52	4.39	1.76×10^4	

Table 4: Timing comparison between AMG, agglomerated multigrid (AggloMG), and Block-Jacobi preconditioners for polynomial degrees $p = 1, 2, 3$ on the three-dimensional ventricle test with 256 MPI processes. Wall-clock times for the setup phase (t_{setup}), the average wall-clock time per time step (t_{CG}), and the overall simulation time (t_{TOT}) are reported, along with the average number of PCG iterations per time step (It.), rounded to the nearest integer. The total number of time steps is 4000.

Hierarchy for ventricle mesh (128 processes)				Hierarchy for ventricle mesh (256 processes)		
Level l	Card($\mathcal{T}_l$)	DoFs	Card($\mathcal{T}_{l-1}$)/Card($\mathcal{T}_l$)	Card($\mathcal{T}_l$)	DoFs	Card($\mathcal{T}_{l-1}$)/Card($\mathcal{T}_l$)
$l = 0$	374,022	2,992,176		374,022	2,992,176	
$l = 1$	46,809	374,472	~ 7.9	46,871	374,968	~ 7.9
$l = 2$	5,908	47,264	~ 7.9	5,987	47,896	~ 7.8
$l = 3$	768	6,144	~ 7.7	768	6,144	~ 7.8

Table 5: Coarsened hierarchies $\{\mathcal{T}_l\}_{l=0}^3$ on top of the original mesh $\mathcal{T}_0 \equiv \mathcal{T}_h$, partitioning the realistic ventricle mesh across different numbers of processes. The number of agglomerates and degrees of freedom (with $p = 1$) per level are shown in the second and third columns, while the ratio between the cardinality of consecutive grids is reported in the last column.

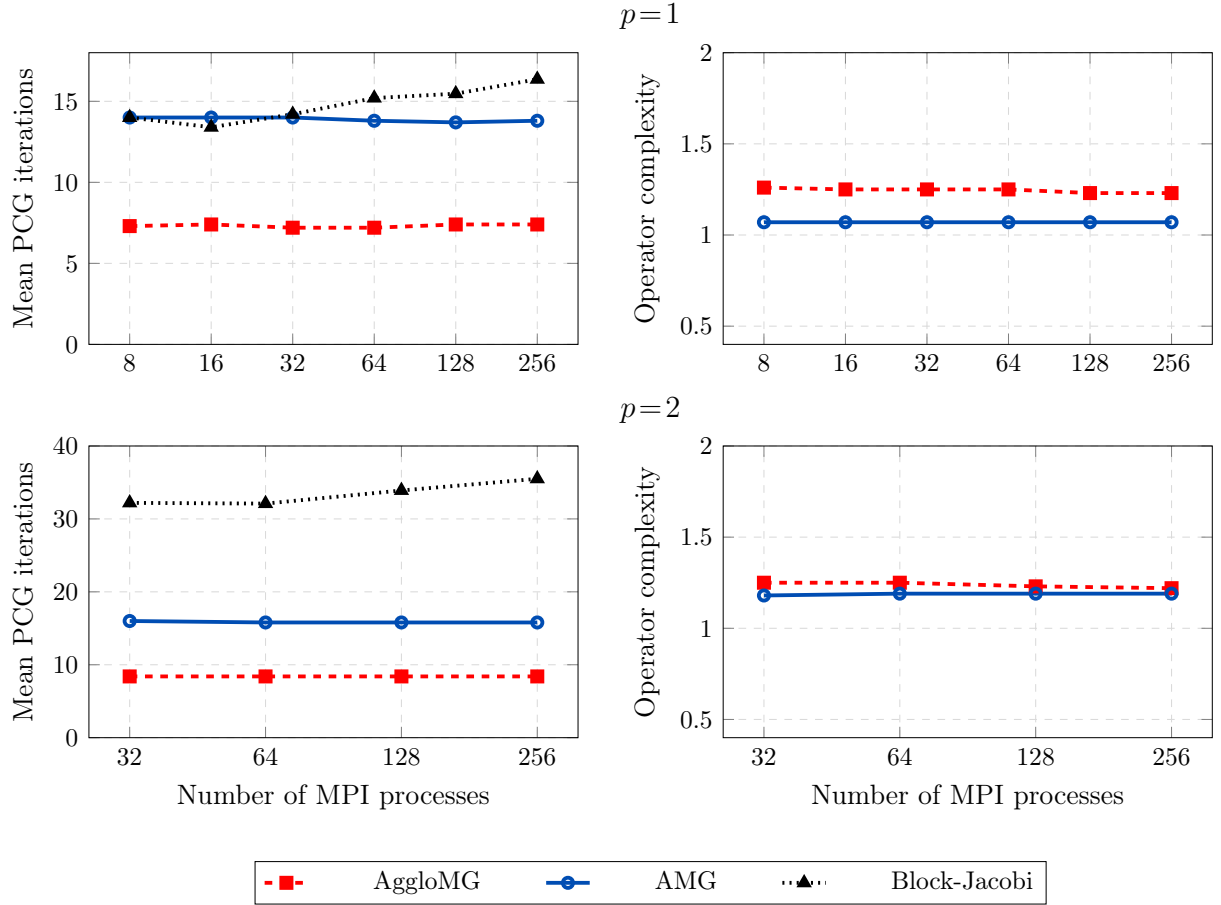


Figure 12: Scalability analysis with respect to the number of MPI processes for the three-dimensional ventricle test case with polynomial degrees $p=1$ (top row) and $p=2$ (bottom row). Left column: average number of PCG iterations per time step. The number of iterations is averaged over the first 20 time steps. Right column: operator complexity defined as the ratio of total non-zeros in the multigrid hierarchy to the non-zeros in the fine-grid operator. Notice that the operator complexity is a metric defined only for multigrid approaches, and hence it is not reported for Block-Jacobi.

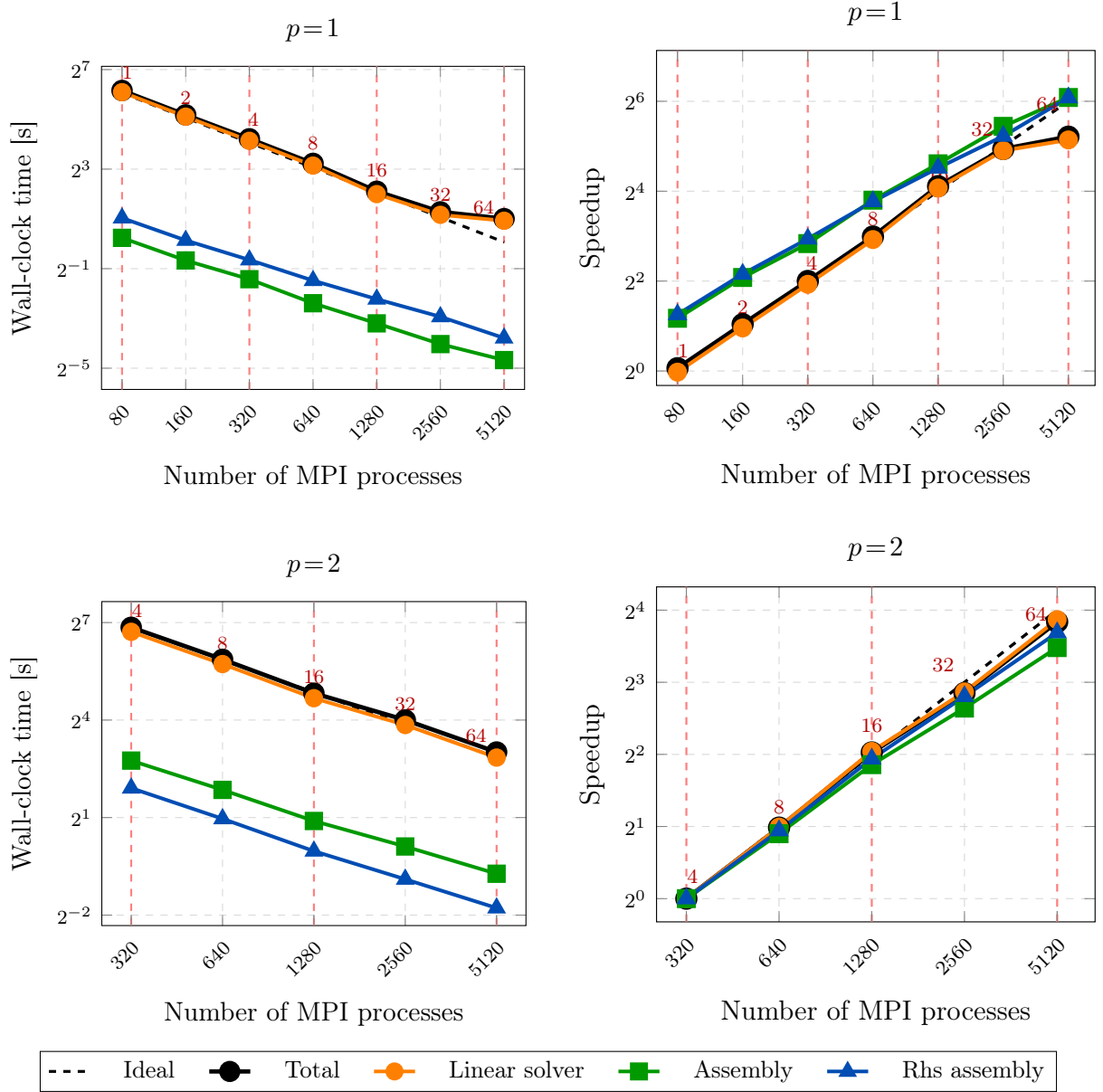


Figure 13: Strong scaling results for the realistic ventricle test case with $p=1$ (top) and $p=2$ (bottom), using 23,937,408 and 80,788,752 degrees of freedom, respectively. Left: wall-clock time for different simulation components. Right: speedup computed relative to the baseline configuration with 80 processes for $p=1$ and 320 processes for $p=2$. Node counts are annotated in red.

7. Conclusions and outlook

This work addressed the development of a novel multilevel preconditioner for the discontinuous Galerkin discretization of the monodomain problem in cardiac electrophysiology. The preconditioner exploits the flexibility of discontinuous Galerkin methods in terms of their ability to handle general polytopic shapes, which can be generated by agglomeration of fine mesh elements. The R-tree-based agglomeration algorithm from [47] enables the construction of nested hierarchies of coarser polytopic grids and their associated operators, yielding a robust and computationally efficient multigrid preconditioner. Our preconditioner exhibits good scalability with respect to both the polynomial degree and the number of levels, maintaining low iteration counts for a cohort of simulations. We employ state-of-the-art matrix-free operator evaluation to further improve computational efficiency. The effectiveness of the proposed methodology and its performance are validated through a sequence of tests involving two-dimensional and three-dimensional domains, different ionic models, and employing real unstructured geometries. Future work will focus on efficient coarse-operator evaluation, which is currently under investigation. Integration with existing high-performance AMG packages and application to more realistic settings and problems, including agglomeration of adaptively refined meshes, are also topics for future research.

Data availability

The source code implementing the methodology and the experiments showcased in this paper are publicly available at the GitHub page of POLYDEAL [48]. Detailed instructions for compilation and execution are provided in the repository.

CRedit author statement

M. Feder: Formal analysis, Investigation, Methodology, Software, Visualization, Writing – Original draft, Writing – review and editing. **P.C. Africa:** Conceptualization, Formal analysis, Methodology, Supervision, Writing – review and editing.

Acknowledgments

MF and PCA acknowledge support of the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (call HORIZON-EUROHPC-JU-2023-COE-03, grant agreement No. 101172493 "dealii-X"). PCA acknowledges support by the European Union - NextGenerationEU, in the framework of the iNEST - Interconnected Nord-Est Innovation Ecosystem (iNEST ECS00000043 - CUP G93C22000610007) and its CC5 Young Researchers initiative. The authors thank the Barcelona Supercomputing Center for the computational resources made available through the EuroHPC project EHPC-DEV-2026D02-275. The authors are members of Gruppo Nazionale per il Calcolo Scientifico (GNCS), Istituto Nazionale di Alta Matematica (INdAM). The authors are thankful to Prof. Andrea Cangiani, Prof. Luca Heltai, and Prof. Fabio Durastante for useful discussions.

Appendix A. Multigrid and model parameters

This appendix reports the parameters employed in the multigrid preconditioners and ionic models used in the numerical tests.

Parameter	Value
Smoother	Chebyshev
Smoother degree	3
Smoother sweeps	3
V-cycle applications	1
Aggregation type	Smoothed aggregation
Aggregation threshold	0.2
Max size coarse level	2000
Coarse solver	Amesos-KLU

Table A.6: Parameters for TrilinosML preconditioner [56] (Trilinos 14.4.0). When using polynomial degrees higher than one, `higher_order_elements` is set to true. In that case, the `Uncoupled` option is automatically selected as aggregation type.

Parameter	Value	Parameter	Value
Cycle type	V-cycle	Cycle type	V-cycle
Number of cycles	1	Number of cycles	1
Smoother	Chebyshev (degree 3)	Smoother	Chebyshev (degree 3)
Pre-smoothing steps	3	Pre-smoothing steps	3
Post-smoothing steps	3	Post-smoothing steps	3
Coarse solver	MUMPS	Coarse solver	PCG+AMG

Table A.7: Agglomerated multigrid configuration for the two-dimensional test case.

Table A.8: Agglomerated multigrid configuration for the three-dimensional test case.

Parameter	Value
C_m	$1 \times 10^{-2} \text{ [m}^{-1}\text{]}$
χ_m	$1 \times 10^5 \text{ [F m}^{-2}\text{]}$
Σ	$1.2 \times 10^{-1} \text{ [m}^2 \text{ s}^{-2}\text{]}$
$\mathbb{D}$	$\Sigma I_{2 \times 2} \text{ [m}^2 \text{ s}^{-2}\text{]}$
κ	1.95×10^1
ϵ	4×10^1
Γ	1×10^{-1}
a	1.3×10^{-2}

Table A.9: FitzHugh-Nagumo parameters used in the 2D numerical test.

Parameter	Value	Parameter	Value	Parameter	Value
σ_l	$1.0 \times 10^{-4} \text{ [m}^2 \text{ s}^{-2}]$	τ_1'	$6 \times 10^{-2} \text{ [s}^{-1}]$	k_2	6.5×10^1
σ_t	$0.5 \times 10^{-4} \text{ [m}^2 \text{ s}^{-2}]$	τ_1''	$1.15 \text{ [s}^{-1}]$	k_3	2.0994
σ_n	$0.1 \times 10^{-4} \text{ [m}^2 \text{ s}^{-2}]$	τ_2'	$7 \times 10^{-2} \text{ [s}^{-1}]$	k_{so}	2.0
τ_o'	$6 \times 10^{-3} \text{ [s}^{-1}]$	τ_2''	$2 \times 10^{-2} \text{ [s}^{-1}]$	w_∞^*	9.4×10^{-1}
τ_o''	$6 \times 10^{-3} \text{ [s}^{-1}]$	τ_3'	$2.7342 \times 10^{-3} \text{ [s}^{-1}]$	V_1	3×10^{-1}
τ_{so}'	$4.3 \times 10^{-2} \text{ [s}^{-1}]$	τ_3''	$2 \times 10^{-2} \text{ [s}^{-1}]$	w_∞^*	9.4×10^{-1}
τ_{so}''	$2 \times 10^{-4} \text{ [s}^{-1}]$	τ_3'	$2.7342 \times 10^{-3} \text{ [s}^{-1}]$	V_1	3×10^{-1}
τ_{si}	$2.8723 \times 10^{-3} \text{ [s}^{-1}]$	τ_3''	3×10^{-3}	V_{1m}	1.5×10^{-2}
τ_{fi}	$1.1 \times 10^{-4} \text{ [s}^{-1}]$	τ_2^+	$2.8 \times 10^{-1} \text{ [s}^{-1}]$	V_2	1.5×10^{-2}
τ_1^+	$1.4506 \times 10^{-3} \text{ [s}^{-1}]$	τ_2^∞	$7 \times 10^{-2} \text{ [s}^{-1}]$	V_{2m}	3×10^{-2}
				V_3	9.087×10^{-1}
				$\hat{V}$	1.58
				V_o	6×10^{-3}
				V_{so}	6.5×10^{-1}

Table A.10: Parameters for Bueno-Orovio model used in the 3D numerical test with the idealized ventricle mesh. In this case $\chi_m \equiv C_m = 1$. In the realistic ventricle test, the parameters are the same except for the conductivities, which are set to $\sigma_l = 10 \times 10^{-5} \text{ [m}^2 \text{ s}^{-2}]$, $\sigma_t = 2 \times 10^{-5} \text{ [m}^2 \text{ s}^{-2}]$, and $\sigma_n = 2 \times 10^{-5} \text{ [m}^2 \text{ s}^{-2}]$. Values have been chosen according to [4].

p	DoFs	$\Delta t \text{ [s]}$	PCG Iterations (AggloMG)				PCG Iterations (AMG)				PCG Iterations (Block-Jacobi)			
			Mean	Std Dev	Min	Max	Mean	Std Dev	Min	Max	Mean	Std Dev	Min	Max
1	3,263,232	5×10^{-4}	10.48	0.51	9	11	19.89	0.98	17	22	24.26	2.18	21	28
		1×10^{-4}	7.78	0.58	6	9	14.70	0.94	11	17	13.53	1.27	11	16
		5×10^{-5}	6.98	0.38	6	8	13.00	0.62	10	15	10.43	0.97	9	12
2	11,013,408	5×10^{-4}	10.90	0.69	9	12	19.92	1.21	17	22	49.91	3.70	43	56
		1×10^{-4}	8.62	0.63	7	10	15.94	0.89	14	18	28.95	2.33	24	33
		5×10^{-5}	7.39	0.59	5	9	14.07	0.91	12	16	21.89	1.97	18	25
3	26,105,856	5×10^{-4}	12.21	1.11	11	14	17.48	1.23	15	19	71.86	4.88	63	81
		1×10^{-4}	8.89	0.67	7	10	12.10	0.89	10	14	44.71	3.53	38	50
		5×10^{-5}	7.88	0.61	6	9	10.48	0.84	8	12	35.04	2.95	29	40

Table A.11: Statistics (mean value, standard deviation, and min-max) of PCG iterations per time step for the idealized left ventricle test (up to $T = 0.1 \text{ s}$) case across different time steps Δt and polynomial degrees p , for the three preconditioners AggloMG, AMG, and Block-Jacobi.

References

- [1] T. Abdelhamid, N. M. M. Huynh, S. Zampini, R. Chen, L. F. Pavarino, and S. Scacchi. Adaptive BDDC preconditioners for the bidomain model on unstructured ventricular finite element meshes. *Computer Methods in Applied Mechanics and Engineering*, 447:118366, 2025.
- [2] M. Adams, M. Brezina, J. Hu, and R. Tuminaro. Parallel multigrid smoothing: polynomial versus Gauss-Seidel. *Journal of Computational Physics*, 188(2):593–610, 2003.

- [3] P. C. Africa, R. Piersanti, M. Fedele, L. Dede', and A. Quarteroni. lifex-fiber: an open tool for myofibers generation in cardiac computational models. *BMC Bioinformatics*, 24(1):143, April 2023.
- [4] P. C. Africa, R. Piersanti, F. Regazzoni, M. Bucelli, M. Salvador, M. Fedele, S. Pagan, L. Dede, and A. Quarteroni. lifex-ep: a robust and efficient software for cardiac electrophysiology simulations. *BMC Bioinformatics*, 24, 10 2023.
- [5] P. C. Africa, M. Salvador, P. Gervasio, L. Dede', and A. Quarteroni. A matrix-free high-order solver for the numerical solution of cardiac electrophysiology. *Journal of Computational Physics*, 478:111984, 2023.
- [6] P. R. Amestoy, A. Guermouche, J.-Y. L'Excellent, and S. Pralet. Hybrid scheduling for the parallel solution of linear systems. *Parallel Computing*, 32(2):136–156, 2006.
- [7] P. F. Antonietti, S. Berrone, M. Busetto, and M. Verani. Agglomeration-based geometric multigrid schemes for the Virtual Element Method. *SIAM Journal on Numerical Analysis*, 61(1):223–249, 2023.
- [8] P. F. Antonietti, S. Bertoluzza, and F. Credali. The Reduced Basis Multigrid scheme for the Virtual Element Method. *arXiv:2511.22219*, 2026.
- [9] P. F. Antonietti, M. Caldana, I. Mazzieri, and A. R. Fraschini. MAGNET: an open-source library for mesh agglomeration by graph neural networks. *Engineering with Computers*, pages 1–26, 2025.
- [10] P. F. Antonietti, M. Corti, and G. Martinelli. Polytopal mesh agglomeration via geometrical deep learning for three-dimensional heterogeneous domains. *Mathematics and Computers in Simulation*, 241:335–353, 2026.
- [11] P. F. Antonietti, L. Dede', G. Loli, M. Montardini, G. Sangalli, and P. Tesini. Space-time isogeometric analysis of cardiac electrophysiology. *Computer Methods in Applied Mechanics and Engineering*, 441:117957, 2025.
- [12] P. F. Antonietti, P. Houston, and G. Pennesi. Fast numerical integration on polytopic meshes with applications to discontinuous Galerkin finite element methods. *Journal of Scientific Computing*, 77:1339–1370, 2019.
- [13] P. F. Antonietti, P. Houston, G. Pennesi, and E. Süli. An agglomeration-based massively parallel non-overlapping additive Schwarz preconditioner for high-order discontinuous Galerkin methods on polytopic grids. *Mathematics of Computation*, 89(325):2047–2083, 2020.
- [14] P. F. Antonietti, P. Houston, M. Sarti, and M. Verani. Multigrid algorithms for hp -version interior penalty discontinuous Galerkin methods on polygonal and polyhedral meshes. *Calcolo*, 54:1169–1198, 2017.

- [15] P. F. Antonietti, L. Mascotto, and M. Verani. A multigrid algorithm for the p-version of the Virtual Element Method. *ESAIM: M2AN*, 52(1):337–364, 2018.
- [16] P. F. Antonietti and L. Melas. Algebraic multigrid schemes for high-order nodal discontinuous Galerkin methods. *SIAM Journal on Scientific Computing*, 42(2):A1147–A1173, 2020.
- [17] P. F. Antonietti and G. Pennesi. V-cycle multigrid algorithms for discontinuous Galerkin methods on non-nested polytopic meshes. *Journal of Scientific Computing*, 78:625–652, 2019.
- [18] P. F. Antonietti, M. Sarti, and M. Verani. Multigrid algorithms for *hp*-discontinuous Galerkin discretizations of elliptic problems. *SIAM Journal on Numerical Analysis*, 53(1):598–618, 2015.
- [19] D. Arndt, W. Bangerth, M. Bergbauer, M. Feder, M. Fehling, J. Heinz, T. Heister, L. Heltai, M. Kronbichler, M. Maier, P. Munch, J.-P. Pelteret, B. Turcksin, D. Wells, and S. Zampini. The `deal.II` library, version 9.5. *Journal of Numerical Mathematics*, 31(3):231–246, 2023.
- [20] D. Arndt, W. Bangerth, D. Davydov, T. Heister, L. Heltai, M. Kronbichler, M. Maier, J.-P. Pelteret, B. Turcksin, and D. Wells. The `deal.II` finite element library: Design, features, and insights. *Computers & Mathematics with Applications*, 81:407–422, Jan. 2021.
- [21] D. N. Arnold, F. Brezzi, B. Cockburn, and L. D. Marini. Unified analysis of discontinuous Galerkin methods for elliptic problems. *SIAM Journal on Numerical Analysis*, 39(5):1749–1779, 2002.
- [22] W. Bangerth, C. Burstedde, T. Heister, and M. Kronbichler. Algorithms and data structures for massively parallel generic adaptive finite element codes. *ACM Trans. Math. Softw.*, 38(2), Jan. 2012.
- [23] F. Bassi, L. Botti, A. Colombo, D. Di Pietro, and P. Tesini. On the flexibility of agglomeration based physical space discontinuous Galerkin discretizations. *Journal of Computational Physics*, 231(1):45–65, 2012.
- [24] P. Bastian, M. Blatt, and R. Scheichl. Algebraic multigrid for discontinuous Galerkin discretizations of heterogeneous elliptic problems. *Numerical Linear Algebra with Applications*, 19(2):367–388, 2012.
- [25] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R^* -tree: an efficient and robust access method for points and rectangles. In *ACM SIGMOD Conference*, 1990.
- [26] M. Benzi, G. H. Golub, and J. Liesen. Numerical solution of saddle point problems. *Acta Numerica*, 14:1–137, 2005.

- [27] M. L. Bittencourt, C. C. Douglas, and R. A. Feijóo. Nonnested multigrid methods for linear problems. *Numerical Methods for Partial Differential Equations*, 17(4):313–331, 2001.
- [28] Boost. Boost C++ Libraries. <http://www.boost.org/>, 2015.
- [29] F. Botta, M. Calafà, P. C. Africa, C. Vergara, and P. F. Antonietti. High-order discontinuous Galerkin methods for the monodomain and bidomain models. *Mathematics in Engineering*, 6(6):726–741, 2024.
- [30] L. Botti, A. Colombo, and F. Bassi. h-multigrid agglomeration based solution strategies for discontinuous Galerkin discretizations of incompressible flow problems. *Journal of Computational Physics*, 347:382–415, 2017.
- [31] A. Bueno-Orovio, E. M. Cherry, and F. H. Fenton. Minimal model for human ventricular action potentials in tissue. *Journal of Theoretical Biology*, 253(3):544–560, 2008.
- [32] C. Burstedde, L. C. Wilcox, and O. Ghattas. **p4est**: Scalable algorithms for parallel adaptive mesh refinement on forests of octrees. *SIAM Journal on Scientific Computing*, 33(3):1103–1133, 2011.
- [33] A. Cangiani, Z. Dong, and E. Georgoulis. *hp*-version discontinuous Galerkin methods on essentially arbitrarily-shaped elements. *Mathematics of Computation*, 91(333):1–35, 2022.
- [34] A. Cangiani, Z. Dong, E. H. Georgoulis, and P. Houston. *hp-version discontinuous Galerkin methods on polygonal and polyhedral meshes*. SpringerBriefs in Mathematics. Springer, Cham, 2017.
- [35] T. F. Chan, J. Xu, and L. T. Zikatanov. An agglomeration multigrid method for unstructured grids. *Contemporary mathematics*, 1998.
- [36] Y. Chen and G. N. Wells. Multigrid on unstructured meshes with regions of low quality cells. *arXiv preprint arXiv:2402.12947*, 2024.
- [37] P. Colli Franzone, L. Pavarino, and B. Taccardi. Simulating patterns of excitation, repolarization and action potential duration with cardiac bidomain and monodomain models. *Mathematical Biosciences*, 197(1):35–66, 2005.
- [38] P. Colli Franzone and L. F. Pavarino. A parallel solver for reaction-diffusion systems in computational electrocardiology. *Mathematical Models and Methods in Applied Sciences*, 14(06):883–911, 2004.
- [39] P. D’Ambra, F. Durastante, and S. Filippone. AMG preconditioners for linear solvers towards extreme scale. *SIAM Journal on Scientific Computing*, 43(5):S679–S703, 2021.

- [40] P. D’Ambra, F. Durastante, and S. Filippone. Parallel Sparse Computation Toolkit. *Software Impacts*, 15:100463, 2023.
- [41] S. Dargaville, A. Buchan, R. Smedley-Stevenson, P. Smith, and C. Pain. A comparison of element agglomeration algorithms for unstructured geometric multigrid. *Journal of Computational and Applied Mathematics*, 390:113379, 2021.
- [42] D. A. Di Pietro, F. Hülsemann, P. Matalon, P. Mycek, U. Rüde, and D. Ruiz. Towards robust, fast solutions of elliptic equations on complex domains through hybrid high-order discretizations and non-nested multigrid methods. *International Journal for Numerical Methods in Engineering*, 122(22):6576–6595, 2021.
- [43] D. A. Di Pietro, P. Matalon, P. Mycek, and U. Rüde. High-order multigrid strategies for HHO discretizations of elliptic equations. *Numerical Linear Algebra with Applications*, 30(1):e2456, 2023.
- [44] C. R. Dohrmann. A preconditioner for substructuring based on constrained energy minimization. *SIAM Journal on Scientific Computing*, 25(1):246–258, 2003.
- [45] R. D. Falgout. An Introduction to Algebraic Multigrid. *Computing in Science and Engg.*, 8(6):24–33, Nov. 2006.
- [46] R. D. Falgout and U. M. Yang. hypre: A library of high performance preconditioners. In *Proceedings of the International Conference on Computational Science-Part III, ICCS ’02*, pages 632–641, Berlin, Heidelberg, 2002. Springer-Verlag.
- [47] M. Feder, A. Cangiani, and L. Heltai. R3MG: R-tree based agglomeration of polytopal grids with applications to multilevel methods. *Journal of Computational Physics*, 526:113773, 2025.
- [48] M. Feder, L. Heltai, and A. Cangiani. polyDEAL. <https://github.com/fdrmerc/Polydeal>.
- [49] M. Feder, L. Heltai, P. Munch, and M. Kronbichler. Matrix-free implementation of the non-nested multigrid method. *arXiv:2412.10910*, 2024.
- [50] R. FitzHugh. Impulses and physiological states in theoretical models of nerve membrane. *Biophysical Journal*, 1(6):445–466, 1961.
- [51] P. C. Franzone, L. F. Pavarino, and S. Scacchi. *Mathematical Cardiac Electrophysiology*. MS&A – Modeling, Simulation and Applications. Springer International Publishing, 2014.
- [52] M. Gee, C. Siefert, J. Hu, R. Tuminaro, and M. Sala. Ml 5.0 smoothed aggregation user’s guide. 01 2006.
- [53] A. Gerbi, L. Dedè, and A. Quarteroni. A monolithic algorithm for the simulation of cardiac electromechanics in the human left ventricle. *Mathematics in Engineering*, 1(1):1–37, 2019.

- [54] P. Gervasio, F. Saleri, and A. Veneziani. Algebraic fractional-step schemes with spectral methods for the incompressible Navier-Stokes equations. *Journal of Computational Physics*, 214(1):347–365, 2006.
- [55] A. Guttman. R-trees: a dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data*, SIGMOD '84, page 47–57, New York, NY, USA, 1984. Association for Computing Machinery.
- [56] M. A. Heroux, R. A. Bartlett, V. E. Howle, R. J. Hoekstra, J. J. Hu, T. G. Kolda, R. B. Lehoucq, K. R. Long, R. P. Pawlowski, E. T. Phipps, A. G. Salinger, H. K. Thornquist, R. S. Tuminaro, J. M. Willenbring, A. Williams, and K. S. Stanley. An overview of the trilinos project. *ACM Transactions on Mathematical Software*, 31(3):397–423, sep 2005.
- [57] J. M. Hoermann, C. Bertoglio, M. Kronbichler, M. R. Pfaller, R. Chabiniok, and W. A. Wall. An adaptive hybridizable discontinuous Galerkin approach for cardiac electrophysiology. *International Journal for Numerical Methods in Biomedical Engineering*, 34(5):e2959, 2018.
- [58] N. M. M. Huynh, L. F. Pavarino, and S. Scacchi. Parallel Newton–Krylov BDDC and FETI-DP deluxe solvers for implicit time discretizations of the cardiac bidomain equations. *SIAM Journal on Scientific Computing*, 44(2):B224–B249, 2022.
- [59] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998.
- [60] G. Karypis, K. Schloegel, and V. Kumar. Parmetis: Parallel graph partitioning and sparse matrix ordering library. 1997.
- [61] A. Klawonn, O. B. Widlund, and M. Dryja. Dual-primal FETI methods for three-dimensional elliptic problems with heterogeneous coefficients. *SIAM Journal on Numerical Analysis*, 40(1):159–179, 2002.
- [62] M. Kronbichler and K. Kormann. A generic interface for parallel cell-based finite element operator application. *Computers & Fluids*, 63:135–147, 2012.
- [63] M. Kronbichler and K. Kormann. Fast matrix-free evaluation of discontinuous Galerkin finite element operators. *ACM Trans. Math. Softw.*, 45(3), Aug. 2019.
- [64] C. B. Leimer Saglio, S. Pagani, M. Corti, and P. F. Antonietti. A high-order discontinuous Galerkin method for the numerical modeling of epileptic seizures. *Computers & Mathematics with Applications*, 205:112–131, 2026.
- [65] C. B. Leimer Saglio, S. Pagani, and P. F. Antonietti. A p-adaptive polytopal discontinuous Galerkin method for high-order approximation of brain electrophysiology. *Computer Methods in Applied Mechanics and Engineering*, 446:118249, 2025.

- [66] D. J. Mavriplis and V. Venkatakrishnan. A 3D agglomeration multigrid solver for the Reynolds-Averaged Navier-Stokes equations on unstructured meshes. *International Journal for Numerical Methods in Fluids*, 23(6):527–544, 1996.
- [67] S. A. Orszag. Spectral methods for problems in complex geometries. *Journal of Computational Physics*, 37(1):70–92, 1980.
- [68] Y. Pan and P.-O. Persson. Agglomeration-based geometric multigrid solvers for compact discontinuous Galerkin discretizations on unstructured meshes. *Journal of Computational Physics*, 449:110775, 2022.
- [69] P. Pathmanathan, G. Mirams, J. Southern, and J. Whiteley. The significant effect of the choice of ionic current integration method in cardiac electro-physiological simulations. *International Journal for Numerical Methods in Biomedical Engineering*, 27:1751 – 1770, 11 2011.
- [70] L. F. Pavarino and S. Scacchi. Multilevel additive Schwarz preconditioners for the bidomain reaction-diffusion system. *SIAM Journal on Scientific Computing*, 31(1):420–443, 2008.
- [71] R. Piersanti, P. C. Africa, M. Fedele, C. Vergara, L. Dede’, A. F. Corno, and A. Quarteroni. Modeling cardiac muscle fibers in ventricular and atrial electrophysiology simulations. *Computer Methods in Applied Mechanics and Engineering*, 373:113468, 2021.
- [72] G. Plank, M. Liebmann, R. Weber dos Santos, E. J. Vigmond, and G. Haase. Algebraic multigrid preconditioner for the cardiac bidomain model. *IEEE transactions on bio-medical engineering*, 54(4):585–596, April 2007.
- [73] D. Prada and M. Pennacchio. Algebraic multigrid methods for virtual element discretizations: A numerical study. *arXiv preprint arXiv:1812.02161*, 2018.
- [74] A. Quarteroni, A. Manzoni, and C. Vergara. The cardiovascular system: Mathematical modelling, numerical algorithms and clinical applications. *Acta Numerica*, 26:365–590, 2017.
- [75] B. Riviere. *Discontinuous Galerkin Methods For Solving Elliptic And Parabolic Equations: Theory and Implementation*, volume 35. 01 2008.
- [76] J. Rogers and A. McCulloch. A collocation-Galerkin finite element model of cardiac action potential propagation. *IEEE Transactions on Biomedical Engineering*, 41(8):743–757, 1994.
- [77] C. B. L. Saglio, S. Pagani, and P. F. Antonietti. A massively parallel non-overlapping Schwarz preconditioner for PolyDG methods in brain electrophysiology. *arXiv preprint arXiv:2512.19536*, 2025.

- [78] J. Schreiner and K.-A. Mardal. Simulating epileptic seizures using the bidomain model. *Scientific Reports*, 12(1):10065, 2022.
- [79] M. Strocchi, C. M. Augustin, M. A. F. Gsell, E. Karabelas, A. Neic, K. Gillette, O. Razeghi, A. J. Prassl, E. J. Vigmond, J. M. Behar, J. Gould, B. Sidhu, C. A. Rinaldi, M. J. Bishop, G. Plank, and S. A. Niederer. A publicly available virtual cohort of four-chamber heart meshes for cardiac electro-mechanics simulations. *PLOS ONE*, 15(6):1–26, 06 2020.
- [80] H. Sundar, G. Stadler, and G. Biros. Comparison of multigrid algorithms for high-order continuous finite element discretizations. *Numerical Linear Algebra with Applications*, 22(4):664–680, 2015.
- [81] K. H. W. J. Ten Tusscher, D. Noble, P. J. Noble, and A. V. Panfilov. A model for human ventricular tissue. *American Journal of Physiology-Heart and Circulatory Physiology*, 286(4):H1573–H1589, 2004.
- [82] K. H. W. J. Ten Tusscher and A. V. Panfilov. Alternans and spiral breakup in a human ventricular tissue model. *American Journal of Physiology-Heart and Circulatory Physiology*, 291(3):H1088–H1100, 2006.
- [83] U. Trottenberg, C. Oosterlee, and A. Schüller. *Multigrid*. Elsevier Academic Press, London, 2001.
- [84] S. Zampini. Dual-primal methods for the cardiac bidomain model. *Mathematical Models and Methods in Applied Sciences*, 24(04):667–696, 2014.