

lymph: discontinuous poLYtopal methods for Multi-PHysics differential problems

Paola F. Antonietti¹, Stefano Bonetti¹, Michele Botti¹, Mattia Corti¹, Ivan Fumagalli¹, and Ilario Mazzieri¹

¹MOX-Dipartimento di Matematica, Politecnico di Milano, Piazza Leonardo da Vinci 32, Milan, 20133, Italy

October 23, 2024

Abstract

We present the library `lymph` for the finite element numerical discretization of coupled multi-physics problems. `lymph` is a Matlab library for the discretization of partial differential equations based on high-order discontinuous Galerkin methods on polytopal grids (PolyDG) for spatial discretization coupled with suitable finite-difference time marching schemes. The objective of the paper is to introduce the library by describing it in terms of installation, input/output data, and code structure, highlighting – when necessary – key implementation aspects related to the method. A user guide, proceeding step-by-step in the implementation and solution of a Poisson problem, is also provided. In the last part of the paper, we show the results obtained for several differential problems, namely the Poisson problem, the heat equation, the elastodynamics system, and a multiphysics problem coupling poroelasticity and acoustic equations. Through these examples, we show the convergence properties and highlight some of the main features of the proposed method, i.e. geometric flexibility, high-order accuracy, and robustness with respect to heterogeneous physical parameters.

1 Introduction

The numerical solution of coupled multi-physics problems is of crucial importance nowadays, spanning different computational areas and applications. We find coupled problems in several engineering fields, e.g. in the context of life sciences for the modeling of soft tissues such as the heart or the brain, or in computational geosciences for studying seismicity, greenhouse gas sequestration, or geothermal energy production. The numerical simulation of these problems is challenging due to their complex nature: phenomena with different spatial and/or temporal scales, the interaction of several physical laws, and (possibly moving) objects with different materials and properties. Along with their intrinsic complexity, the accurate approximation of such problems through numerical methods often requires different constraints on geometric details, scale resolution, or local refinement of the computational mesh.

Over the past few years, polygonal and polyhedral (polytopal, for short) meshes have become increasingly popular as a solution for these numerical challenges due to their flexibility in representing intricate, possibly moving, geometries and interfaces, and heterogeneous media. Thus, a particular interest has been devoted to the development of numerical methods that can handle general grids, such as Discontinuous Galerkin (see e.g., [1, 2, 3, 4, 5]), Virtual Element Method (see e.g.,[6, 7, 8, 9, 10, 11]), Hybrid High-Order (see e.g.,[12, 13, 14]), Hybridizable Discontinuous Galerkin (see e.g.,[15, 16, 17]), and, more recently, Staggered Discontinuous Galerkin (see e.g.,[18, 19, 20]). On top of geometric flexibility offered by arbitrarily-shaped elements in describing complicated geometries, polytopal meshes exhibit some distinguished advantages over classical meshes, including for example the process of mesh generation and

handling, see e.g., [21, 22, 23, 24, 25]. On the other hand mesh agglomeration is also a key feature, that does not have an analogous counterpart in classical finite element methods. Agglomeration finds important applications both in the construction of scalable multilevel algebraic solvers and within adaptive algorithms (see e.g. [26, 27] where machine-learning driven refinement and agglomeration strategies for polytopal meshes have been developed). Moreover, agglomerated grids can be effectively employed to accurately represent complicated domain features, such as complex boundaries and/or embedded structures or microstructures, without the need for excessively fine grids. In this work, we focus on the high-order discontinuous Galerkin finite element method on polytopal grids (PolyDG), see e.g., [1, 2, 3, 5]). The use of the PolyDG method offers numerous benefits when dealing with coupled problems: *(i)* flexible representation of complex geometries, *(ii)* flexibility in refinement and agglomeration strategies, *(iii)* ability to cope with non-conforming interfaces, *(iv)* robustness concerning heterogeneities of physical properties, *(v)* arbitrary polynomial approximation order. Related to the geometrical flexibility, another attractive aspect concerns the treatment of transmission conditions that are usually localized on sub-regions of the computational domain and must be represented without compromising efficiency. The PolyDG method possesses some distinguishing features, that make it very appealing for multi-physics differential problems. First, it can be easily combined with agglomeration strategies for adaptivity which usually leads to many small faces per element, since the dimension of local approximation space does not depend on the number of faces. Concerning refinement procedures, its hierarchical basis structure can also be exploited. Second, PolyDG methods ensure very good performance in terms of parallelization and scalability – especially for high polynomial degrees and higher dimensions. Another advantage consists in its dimension-independent formulation, which is particularly useful when moving from 2D problems to 3D ones. Examples of PolyDG schemes can be found, e.g., in [2, 1] for elliptic problems, in [28] for advection-diffusion-reaction problems, in [29] for parabolic problems, in [30, 31] for poroelasticity, and in [32] for elastic wave propagation problems. PolyDG discretizations of multi-physics coupled problems can be found, e.g., in [33, 34, 35] for brain modeling and in [36, 37, 38] for computational geosciences.

This paper aims to introduce **1ymph** (discontinuous poLYthopal methods for Multi-PHysics), an open-source MATLAB library, released under the GNU Lesser General Public License, version 3 (LGPLv3), for the PolyDG approximation of multi-physics problems in two-dimensions. We focus on the two-dimensional case to ensure that the syntax remains clear and user-friendly, particularly when employing MATLAB as the software environment for the library. We point out that there is already some software on the market for the numerical approximation of multi-physics problems (e.g. **Basix** [39], **FEniCS** [40], **lifex** [41, 42, 43], **MFEM** [44], **MOOSE** [45]) and very few for the solution of problems on polytopal meshes (e.g., **VEM++** [46], **HArDCore** [47], **MRST** [48]). However, **1ymph** presents several features that, to the best of the authors' knowledge, make it unique by coupling all the advantages that come from the use of (possibly agglomerated) polytopal meshes and, more specifically, by discretizing the problem via PolyDG schemes. The available implementation hinges on Interior Penalty discontinuous Galerkin formulations, but it can be adapted to other DG schemes, such as the local discontinuous Galerkin or the Bassi-Rebay methods, by suitably modifying the assembly and introducing lifting operators. Moreover, the library is very flexible in terms of coupling different existing physics and implementation of new ones. For time dependent problems, time integration is realized by means of finite-difference time integration schemes (Crank-Nicolson for first-order differential systems, Newmark- β for second-order ones).

The rest of the article is structured as follows: in section 2 we provide a brief introduction to the PolyDG method, concerning its assumptions and basic elements. In section 3 we describe the library in terms of input/output data and code structure, while in section 4 we focus on the core functionalities at the basis of the PolyDG discretization. Then, in section 5 we provide a user guide, proceeding step-by-step in the solution of a Poisson problem and we show some results regarding time-dependent problems, in which the main features of the PolyDG method (e.g. geometric flexibility, high-order accuracy, and robustness concerning heterogeneous media) are exploited.

2 Main ingredients of high-order Polytopal discontinuous Galerkin methods

The purpose of this section is to present the mesh assumptions, the discrete spaces, and some preliminary results. We introduce a polygonal subdivision \mathcal{T}_h of the computational domain $\Omega \subset \mathbb{R}^2$. Next, we define the internal edges as the intersection of any two neighboring elements of \mathcal{T}_h . We define \mathcal{F}_I to be the set of all internal edges. The boundary edges are collected in the set \mathcal{F}_B which yields a subdivision of $\partial\Omega$. Accordingly, the set of all the edges is given by $\mathcal{F}_h = \mathcal{F}_B \cup \mathcal{F}_I$. In what follows, we introduce the main assumptions on the mesh \mathcal{T}_h (cf. [3, 5, 49]).

Definition 2.1 (Polytopic-regular mesh) A mesh \mathcal{T}_h is polytopic-regular if for any $\kappa \in \mathcal{T}_h$, there exist a set of non-overlapping simplices contained in κ , denoted by $\{S_\kappa^F\}_{F \subset \partial\kappa}$, such that, for any face $F \subset \partial\kappa$, the following condition holds: $h_\kappa \lesssim |S_\kappa^F| |F|^{-1}$, with h_κ denoting the diameter of κ (i.e. the maximum distance between two points belonging to the element) and with $|\cdot|$ denoting the Hausdorff measure.

In the above definition and the following, the symbol \lesssim is used to denote the inequality $x \leq Cy$ for a positive constant C not depending on the mesh size and the polynomial approximation order. As a basis for the construction of the PolyDG approximation, we define fully discontinuous polynomial spaces on \mathcal{T}_h . Given an element-wise constant polynomial degree $\ell : \mathcal{T}_h \rightarrow \mathbb{N}_{>0}$ determining the order of the approximation, the discrete spaces are defined such as

$$V_h^\ell = \left\{ v_h \in L^2(\Omega) : v_h|_\kappa \in \mathcal{P}^{\ell_\kappa}(\kappa) \quad \forall \kappa \in \mathcal{T}_h \right\}, \quad \mathbf{V}_h^\ell = \left[V_h^\ell \right]^2,$$

where, for each $\kappa \in \mathcal{T}_h$, the space $\mathcal{P}^{\ell_\kappa}(\kappa)$ is spanned by polynomials of maximum degree $\ell_\kappa = \ell|_\kappa$. We consider a mesh sequence $\{\mathcal{T}_h\}_{h \rightarrow 0}$ satisfying the following properties:

Assumption 2.1 The mesh sequence $\{\mathcal{T}_h\}_{h \rightarrow 0}$ and the polynomial degree ℓ are such that

A.1 $\{\mathcal{T}_h\}_{h \rightarrow 0}$ is uniformly polytopic-regular;

A.2 For each $\mathcal{T}_h \in \{\mathcal{T}_h\}_{h \rightarrow 0}$ and for any pair of neighbouring elements $\kappa^+, \kappa^- \in \mathcal{T}_h$, the following hp-local bounded variation properties hold: $h_{\kappa^+} \lesssim h_{\kappa^-} \lesssim h_{\kappa^+}$ and $\ell_{\kappa^+} \lesssim \ell_{\kappa^-} \lesssim \ell_{\kappa^+}$.

Finally, we also need to introduce the average and jump operators. We start by defining them on each interior edge $F \in \mathcal{F}_I$ shared by the elements κ^\pm as in [50]:

$$\begin{aligned} [[a]] &= a^+ \mathbf{n}^+ + a^- \mathbf{n}^-, \quad [\![\mathbf{a}]\!] = \mathbf{a}^+ \otimes \mathbf{n}^+ + \mathbf{a}^- \otimes \mathbf{n}^-, \quad [[\mathbf{a}]]_n = \mathbf{a}^+ \cdot \mathbf{n}^+ + \mathbf{a}^- \cdot \mathbf{n}^-, \\ \{\!\{a\}\!\} &= \frac{a^+ + a^-}{2}, \quad \{\!\{\mathbf{a}\}\!\} = \frac{\mathbf{a}^+ + \mathbf{a}^-}{2}, \quad \{\!\{\mathbf{A}\}\!\} = \frac{\mathbf{A}^+ + \mathbf{A}^-}{2}, \end{aligned}$$

where $\mathbf{a} \otimes \mathbf{n} = \mathbf{a}\mathbf{n}^T$, and a , \mathbf{a} , \mathbf{A} are (regular enough) scalar-, vector-, and tensor-valued functions, respectively. The notation $(\cdot)^\pm$ is used to denote the trace on F taken within the interior of κ^\pm and \mathbf{n}^\pm is the outer unit normal vector to $\partial\kappa^\pm$. Accordingly, on boundary faces $F \in \mathcal{F}_B$, we set

$$[[a]] = a\mathbf{n}, \quad \{\!\{a\}\!\} = a, \quad [\![\mathbf{a}]\!] = \mathbf{a} \otimes \mathbf{n}, \quad \{\!\{\mathbf{a}\}\!\} = \mathbf{a}, \quad [[\mathbf{a}]]_n = \mathbf{a} \cdot \mathbf{n}, \quad \{\!\{\mathbf{A}\}\!\} = \mathbf{A}.$$

The ingredients presented above are crucial for deriving the PolyDG semi-discrete formulation of a given partial differential problem. To obtain the fully-discrete formulation of time-dependent problems, the PolyDG discretization in space is coupled with a suitable time-integration scheme (e.g. Crank-Nicolson for first-order problems, Newmark- β for second-order problems).

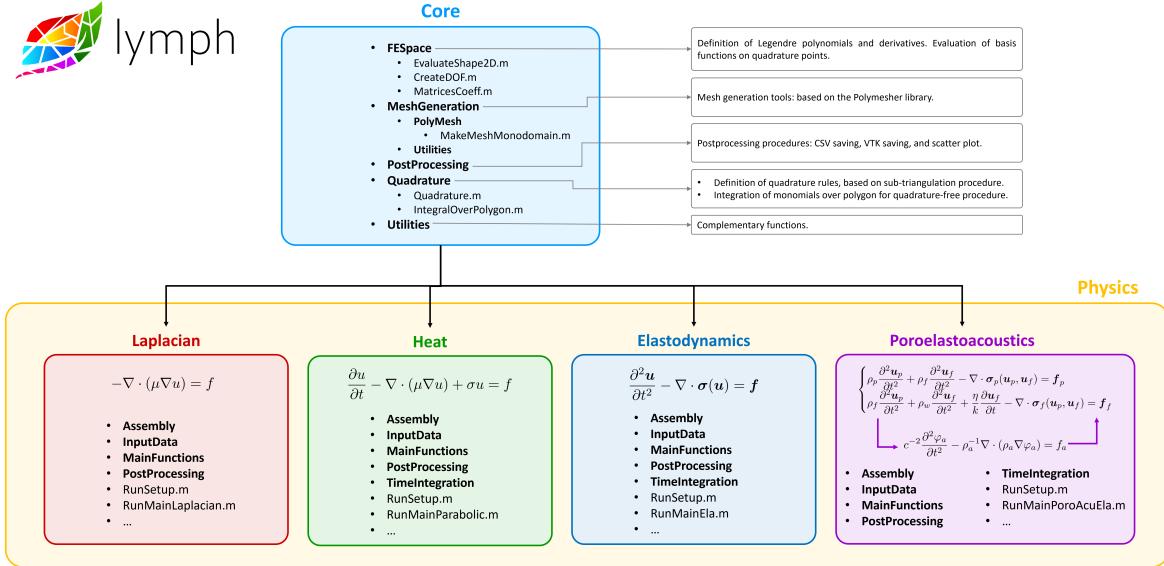


Figure 1: lymph code structure and logo (top left).

3 The lymph library

This section introduces the main structure and the components of lymph library. We give a high-level overview of the library by describing its main functionalities and folder structure. We postpone a more procedural user guide in section 5, going along the numerical approximation and implementation of a specific differential problem.

Overview and installation

lymph is designed to solve either single or multi-physics differential problems employing the PolyDG method, as explained in section 2. It has been developed in Matlab version R2022b, but its functionalities have been successfully tested in all versions from R2020b to R2023b including the **Mapping Toolbox**. The installation of lymph simply consists of downloading the software from the repository <https://bitbucket.org/lymph/lymph> into a directory that is accessible to Matlab. The library is organized into different folders: **Core** which contains the main routines such as mesh generation, polynomial space construction, and quadrature formulas; **Physics**, which contains specific routines for particular problems, organized in subfolders (e.g., **Laplacian**, **Heat**, **Elastodynamics**, **PoroElastoAcoustics**). A flowchart illustrating the code structure and workflow is given in fig. 1.

Design and Data Structures: Principles

The PolyDG discretization is built around several core design principles aimed at flexibility, efficiency, and accuracy in solving PDEs using polygonal meshes. The code leverages specific data structures to efficiently handle mesh generation, finite element computations, and numerical integration to assemble the algebraic system. Below, we detail the key data structures used and the reasons behind their design choices.

- Mesh Representation.** Polygonal meshes are represented using a cell array that stores vertex coordinates, element connectivity, and boundary conditions. This structure allows for flexible han-

dling of complex geometries and supports adaptive refinement.

- **Finite Element Space (FEspace).** The FEspace is defined through a structure that includes information about the basis functions, their derivatives, and the degrees of freedom (DOF). The high-level interface of this centralized structure simplifies the construction of the finite element basis over each mesh element, making the code highly modular and extendable to different element types. This allows users to easily adapt the FEspace for different polynomial degrees, supporting both low- and high-order approximations.
- **Quadrature Rules.** Integration over elements and boundaries is handled using a predefined quadrature array that stores the quadrature points and weights. This design allows for precise control over the integration process, which is crucial for accurately assembling the stiffness matrix and load vectors. Using different quadrature rules enables the optimization of computational cost based on the specific problem requirements.
- **Sparse Matrices for System Assembly.** The assembly of the linear systems, including the stiffness matrix and the right-hand side, is performed using sparse matrix structures. This is particularly beneficial for large-scale simulations where memory efficiency is crucial. By leveraging sparse storage formats, we minimize memory usage and improve the speed of algebraic solvers applied to the resulting linear systems.
- **Post-Processing and Visualization.** To facilitate result analysis, the solution and mesh data are exported using dedicated structures that can be read by Matlab and Paraview. These structures include solution values at each node and element-level information, which enables seamless post-processing and visualization. This decision was made to ensure compatibility with widely used visualization tools, allowing users to easily interpret and present their simulation results.

Customization and User Interaction

The flexibility of the code is enhanced through user-friendly scripts, such as `RunSetup.m`, which allow users to configure simulation parameters, select output formats, and define solver settings. This enables a high level of customization without requiring deep modifications to the underlying code. Users can adapt the library to their specific needs, whether they are solving steady-state or time-dependent problems, or validating results through convergence tests.

Documentation and Support

The full documentation, along with step-by-step tutorials, is provided to guide users through the library's features and applications. This includes practical examples and detailed explanations of the library's setup, aiming to make the onboarding process as smooth as possible for new users. Further information is available at <https://lymph.bitbucket.io/>, where users can access tutorials and examples to deepen their understanding of the library's capabilities.

4 Common aspects of PolyDG discretization: the Core of lymph

In this section, we describe the core functionalities of the `lymph` library, namely, the routines contained in the `Core` folder that are proper of a PolyDG discretization. Moreover, we provide indications on coding strategies that we implemented and can be useful in general finite element software development.

4.1 Mesh Generation

The mesh generation tools included in `lymph` rely upon `PolyMesher v1.1` [22], an open-source library for the generation of 2D polygonal grids, not directly included in the repository but automatically downloaded at the first run of `lymph`. It is important to note that it is possible to use within `lymph` meshes generated by external software whose output format is compatible with that of `PolyMesher`. Specifically, agglomerated grids can also be imported and managed.

4.2 Finite Element Spaces

The construction of an algebraic formulation for the discrete problem requires the construction of a basis for the PolyDG space V_h^ℓ . In the general formulation, the basis is $(\varphi_j)_{j=1}^N$, where $N = \dim\{V_h^\ell\} = |\mathcal{T}_h| \dim\{P^\ell(k)\}$ is the number of degrees of freedom in `femregion.ndof` with $|\mathcal{T}_h|$ as the number of elements `femregion.nel` of the partition and $\dim\{P^\ell(k)\} = \frac{1}{2}(\ell+1)(\ell+2) = \text{femregion.nbases}$.

In the `lymph` library, the basis functions for each physical element κ are constructed starting from the Legendre polynomials of order ℓ [51] in one dimension $\mathcal{L}_i = \mathcal{L}_i(x)$ with $i = 0, \dots, \ell$. Then, by using a tensor product of basis functions in the two directions, we obtain that:

$$\varphi_i(x, y) = \mathcal{L}_j(x)\mathcal{L}_k(y), \quad i = 1, \dots, \frac{1}{2}(\ell+1)(\ell+2), \text{ and } j, k = 1, \dots, \ell, \text{ and } j+k \leq \ell.$$

The reference Legendre polynomials $\hat{\mathcal{L}}$ are constructed initially on the square $[-1, 1] \times [1, 1]$; then the final solution is reported on each polygon using an affine transformation to the bounding box of the element, stored in the matrix `femregion.bbox` [5].

By construction, the basis of the PolyDG space is modal. Therefore, we cannot associate the coefficients of the linear combination expansion to a physical value at a point in the space. For this reason, the final solution will be reconstructed in the quadrature nodes for visualization purposes.

4.3 Evaluation of integrals and use of quadrature formulas

Any dG discretization requires the computation of volume and boundary integrals. As a general example, we consider the following integral (cf. the Poisson problem in Section 5.1) and explain how it is computed in `lymph`. We start by considering the general volume term:

$$\mathbf{A}.loc \rightarrow (\mu \nabla \varphi_j, \nabla \varphi_i)_\kappa, \quad \text{for } i, j = 1, \dots, \frac{1}{2}(\ell+1)(\ell+2), \quad (1)$$

where μ is a given element-wise constant function, $\kappa \in \mathcal{T}_h$ is a generic mesh element, and $(\cdot, \cdot)_\kappa$ denotes the $L^2(\kappa)$ inner product over $\kappa \in \mathcal{T}_h$. We remark that the syntax `A.loc` denotes the volume integral computed for every element κ in the mesh \mathcal{T}_h ; then, for assembling the linear system stemming from the discretization of our differential equation, we need to combine the local matrices to assemble a global volume matrix \mathbf{A} . The evaluation of these integrals is based on the quadrature-free approach proposed in [52]. The idea of the method is to apply Euler's homogeneous function theorem alongside Stokes' theorem to exactly integrate polynomial functions over polytopal elements. This method allows integrals to be computed without the need for a sub-tessellation of the domain. In Algorithm 1, we report the main steps to compute the values of the volume integrals following the quadrature-free approach (cf. `lymph/Core/Quadrature` and `lymph/Laplacian/Assembly` folders for the actual implementation of Algorithm 1 in `lymph`). We compute the volume integrals relying on the decomposition $\int_\kappa \varphi_j \varphi_i = \sum_{k,q} a_{kq}^{ij} \int_\kappa x^k y^q$ in terms of bivariate monomials $x^k y^q$. Once and for all at the beginning, we compute the reference coefficients \hat{a}_{kq}^{ij} corresponding to the case of κ being a rectangle. Then, on each element κ , we scale the reference coefficients to obtain the current a_{kq}^{ij} and we combine them with the volume integrals of bivariate monomials. For computing the volume integrals we simply need the maximum degree of the monomials in x -component, the maximum degree of the monomials in y -component, and the geometrical information about the vertices of the

Algorithm 1 Assemble volume matrices using the quadrature-free approach of [52]

- 1: Compute the coefficients of the polynomial basis functions
 - 2: Compute the reference 2D-monomial coefficients for integral decomposition
 - 3: **for** every element κ in the mesh \mathcal{T}_h **do**
 - 4: Extract the position of the matrix \mathbf{A}_{loc} , cf. (1), associated to the current element κ w.r.t. the global matrix \mathbf{A}
 - 5: Scale the reference 2D-monomial coefficients according to the shape of the polygon κ
 - 6: Compute the bivariate monomial integrals in the polygon κ
 - 7: Evaluate the physical parameters
 - 8: Assemble the local matrix \mathbf{A}_{loc}
 - 9: **end for**
 - 10: Assemble the global matrix \mathbf{A} by combining the local matrices \mathbf{A}_{loc} along with their positions in the global matrix (cf. Step 4 of this Algorithm).
-

polygon (then, of the current element $\kappa \in \mathcal{T}_h$). The efficiency of this approach, which does not require the construction of any sub-tessellation of κ is discussed in detail in [52]. Notice that the integrals of the monomials are assembled exploiting matrix-vector multiplication, so that the assembly of the local matrices does not need additional **for** loops that could hinder computational efficiency. The only **for** loop needed in the algorithm is the one over mesh elements: this choice allows the code to be flexible w.r.t. the heterogeneity of physical parameters and ready for using different polynomial degrees ℓ on each element (p -adaptivity). We also remark that the aforementioned algorithm works also when differential operators (e.g. gradient, divergence) appear in the bilinear forms. Another implementation strategy worth pointing out is the construction of each global matrix from the corresponding local ones. We store the local matrices and the corresponding DOF indices in element-indexed arrays, and then pass them to the **sparse** command to create the global matrices. This strategy yields a significant gain in computational time w.r.t. preallocating the sparse global matrix and then filling it with local information.

To compute the volume integral of the (possibly non-polynomial) forcing term on the element κ , we employ a sub-tessellation of κ and use a Gauss-Legendre quadrature rules therein. More precisely, we consider a sub-tesselation of κ made of triangles **Tria** and, on each triangle, we compute the quantities of interest through a quadrature rule with **femregion.nqn** = $(\ell + 1)^2$ points [53]. We denote the corresponding quadrature nodes **ref_qNodes_2D** and weights **w_2D**. The quadrature rule is exact for the quantities of interest that involve the trial functions, while the possibility of extending this approach for non-polynomial functions (e.g. forcing terms of generic form) is still under investigation. We point out that the quadrature-free formula over each triangle would entail substantially the same computational cost of the chosen quadrature rule, and that other methods to integrate functions on each (sub) triangle can be applied, also using fewer quadrature nodes [53, 54, 55]. Moreover, we point out that this sub-tessellation strategy can be seen as an interpolatory quadrature formula over the original mesh element κ , with an interpolation error of the same order of the numerical scheme (or lower): other interpolatory formulas would require the reconstruction of the polynomial interpolation of the forcing term, and since there is no straightforward way to define such interpolation directly on a polygon, this would encompass a computational effort that is comparable to sub-tessellation. Although the quadrature-free strategy is preferable for matrix assembly whenever possible, **lymph** also allows for the implementation of the sub-tessellation strategy. We refer the reader to **Laplacian/Assembly/MatrixLaplacianST.m** for an example where the use of **for** loops is minimized, as discussed above.

We next discuss the implementation of the face terms. Denoting by $(\cdot, \cdot)_e$ the L^2 inner product over $e \in \mathcal{F}_I$, we consider the following surface integrals $(\{\mu \nabla \varphi_j\}, [\![\varphi_i]\!])_e$ and $(\alpha_e [\![\varphi_j]\!], [\![\varphi_i]\!])_e$ over an edge e shared by two neighboring elements κ^+ and κ^- in \mathcal{T}_h . Using the definitions of $\{\!\!\{\cdot\}\!\!\}$ and $[\![\cdot]\!]$ operators, we get:

Algorithm 2 Assemble face matrices

- 1: Compute reference 1D nodes and weights for quadrature on the faces
- 2: **for** every element κ^+ in the mesh \mathcal{T}_h **do**
- 3: Compute the geometric contribution to the penalty coefficient α_e for every edge of the element κ^+
- 4: **for** every edge e of the element κ^+ **do**
- 5: Extract the id of the neighboring element κ^- (across the current edge e)
- 6: Extract the DOF indexes for assembling the contribution of the current element κ^+
- 7: Scale the reference 1D nodes and weights according to the edge e (cf. Step 1)
- 8: Construct and evaluate the basis functions on the quadrature nodes
- 9: Evaluate the physical parameters
- 10: **if** e is a boundary face **then**
- 11: **if** weakly impose boundary conditions on e (e.g. Dirichlet and Robin b.c.) **then**
- 12: Assemble the local matrices **IA_loc** and **SA_loc** (contributions coming from the element κ^+)
- 13: **end if**
- 14: **else if** e is an internal face **then**
- 15: Assemble the local matrices **IA_loc** and **SA_loc** (contributions coming from the element κ^+)
- 16: Extract the indexes for assembling the contribution of the neighboring element κ^-
- 17: Update the local matrices **IA_loc** and **SA_loc** with the contributions coming from the element κ^- , cf. (2) and (3), respectively
- 18: **end if**
- 19: **end for**
- 20: **end for**
- 21: Assemble the global matrices **IA**, **SA** by combining the local matrices **IA_loc**, **SA_loc** along with their positions in the global matrix (cf. Step 6, Step 16 in the Algorithm).

- $(\{\mu \nabla \varphi_j\}, [\varphi_i])_e = \frac{1}{2}((\mu^+ \nabla \varphi_j^+ - \mu^- \nabla \varphi_j^-) \cdot \mathbf{n}^+, \varphi_i^+)_e + \frac{1}{2}((\mu^- \nabla \varphi_j^- - \mu^+ \nabla \varphi_j^+) \cdot \mathbf{n}^+, \varphi_i^-)_e,$
- $(\alpha_e [\varphi_j], [\varphi_i])_e = (\alpha_e(\varphi_j^+ - \varphi_j^-), \varphi_i^+)_e + (\alpha_e(\varphi_j^- - \varphi_j^+), \varphi_i^+)_e,$

for $i, j = 1, \dots, \frac{1}{2}(\ell+1)(\ell+2)$. When the current element is κ^+ , only the following integrals are computed by means of one-dimensional Gauss-Legendre quadrature rules

$$\text{IA_loc}\{\kappa^+\} \rightarrow \frac{1}{2}(\mu^+ \nabla \varphi_j^+ \cdot \mathbf{n}^+, \varphi_i^+)_e \quad \text{and} \quad \text{IA_loc}\{\kappa^+\} \rightarrow -\left(\frac{1}{2} \mu^- \nabla \varphi_j^- \cdot \mathbf{n}^+, \varphi_i^+\right)_e, \quad (2)$$

and

$$\text{SA_loc}\{\kappa^+\} \rightarrow \alpha_e(\varphi_j^+, \varphi_i^+)_e \quad \text{and} \quad \text{SA_loc}\{\kappa^+\} \rightarrow -\alpha_e(\varphi_j^-, \varphi_i^+)_e, \quad (3)$$

for $i, j = 1, \dots, \frac{1}{2}(\ell+1)(\ell+2)$. Boundary integrals for weakly imposing essential boundary conditions are treated in the same way. We omit the discussion for the sake of brevity. In Algorithm 2, we report the main steps to compute the values of the face integrals (cf. `lymph/Core/Quadrature` and `lymph/Laplacian/Assembly` folders for the actual implementation of Algorithm 2 in `lymph`).

5 Examples

In the following, we show how to solve differential problems with `lymph`. In the subsequent discussion, we present four examples of increasing complexity: the Poisson equation, the heat equation, the elastodynamics equation, and a multiphysics system modeling wave propagation in poroelasto-acoustic media. These examples are paradigmatic in illustrating the capabilities of `lymph` in handling single-physics problems—encompassing both steady-state and time-dependent, as well as scalar and vector-valued differential equations—alongside multiphysics models.

5.1 The Poisson problem

We start, by considering the following problem in a polygonal domain $\Omega \subset \mathbb{R}^2$:

$$\begin{cases} -\nabla \cdot (\mu \nabla u)(\boldsymbol{x}) = f(\boldsymbol{x}), & \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) = g(\boldsymbol{x}), & \boldsymbol{x} \in \partial\Omega, \end{cases} \quad (4)$$

where μ , f and g are given regular functions. We reformulate it using the PolyDG discretization described in section 2 obtaining: find $u_h \in V_h^\ell$ s.t.

$$a_{dG}(u_h, v_h) = F(v_h) \quad \forall v_h \in V_h^\ell, \quad (5)$$

where

$$a_{dG}(u, v) = \sum_{\kappa \in \mathcal{T}_h} (\mu \nabla u, \nabla v)_\kappa - \sum_{e \in \mathcal{F}_h} \left((\{\mu \nabla u\}, [v])_e + ([u], \{\mu \nabla v\})_e - (\alpha_e [u], [v])_e \right) \quad \forall u, v \in V_h^\ell,$$

with the penalization parameter $\alpha : \mathcal{F} \rightarrow \mathbb{R}_+$ defined as [5]:

$$\alpha_e(\boldsymbol{x}) = \begin{cases} C_\alpha \max_{\kappa \in \{\kappa^+, \kappa^-\}} \left(\mu_\kappa \frac{\ell_\kappa^2}{h_\kappa} \right), & \boldsymbol{x} \in e, e \in \mathcal{F}_I, e \subset \partial\kappa^+ \cap \partial\kappa^-, \\ C_\alpha \mu_\kappa \frac{\ell_\kappa^2}{h_\kappa}, & \boldsymbol{x} \in e, e \in \mathcal{F}_B, e \subset \partial\kappa^+ \cap \partial\Omega, \end{cases} \quad (6)$$

with $C_\alpha > 0$ denoting the penalty coefficient to be properly set, and

$$F(v) = \sum_{\kappa \in \mathcal{T}_h} (f, v)_\kappa - \sum_{e \in \mathcal{F}_B} \left((g, \mu \nabla v)_e - (\alpha_e g, v)_e \right) \quad \forall v \in V_h^\ell.$$

By introducing a set of basis functions $\{\varphi_j\}_{j=1}^{N_h}$ for the space V_h^ℓ we can write (5) as the following algebraic problem: find $\boldsymbol{U}_h \in \mathbb{R}^{N_h}$ s.t.

$$A_{dG} \boldsymbol{U}_h = \boldsymbol{F}, \quad (7)$$

with $A_{dG} \in \mathbb{R}^{N_h} \times \mathbb{R}^{N_h}$ defined for any $i, j = 1, \dots, N_h$ as $(A_{dG})_{ij} = a_{dG}(\varphi_j, \varphi_i)$ and $\boldsymbol{F} \in \mathbb{R}^{N_h}$ is given by $\boldsymbol{F}_i = F(\varphi_i)$ for all $i = 1, \dots, N_h$. The entries of the matrix A_{dG} as well as the right-hand side \boldsymbol{F} in (7) are computed as explained in section 4. For the sake of completeness, we introduce the dG-norm as $\|u\|_{dG}^2 = \|\sqrt{\mu} \nabla u\|_{L^2(\Omega)}^2 + \|\sqrt{\alpha_e} [u]\|_{L^2(\mathcal{F}_h)}^2$ for any $u \in V_h^\ell$, and we recall a well-known convergence result of the PolyDG discretization, [5, 56, Theorem 36] from which we have the following convergence rates:

$$\|u - u_h\|_{dG}^2 \lesssim \sum_{\kappa \in \mathcal{T}_h} \frac{h_\kappa^{2(s_\kappa-1)}}{\ell_\kappa^{2(m_\kappa-1)}} \|u\|_{H^{m_\kappa}(\kappa)}^2, \quad \|u - u_h\|_{L^2(\Omega)}^2 \lesssim \sum_{\kappa \in \mathcal{T}_h} \frac{h_\kappa^{2s_\kappa}}{\ell_\kappa^{2m_\kappa}} \|u\|_{H^{m_\kappa}(\kappa)}^2, \quad (8)$$

with $s_\kappa = \min(\ell_\kappa + 1, m_\kappa)$ for all $\kappa \in \mathcal{T}_h$, $m_\kappa > 0$ denoting the local Sobolev regularity of the solution u , and h_κ the element diameter.

5.1.1 Verification test

We consider problem (4) in $\Omega = (0, 1)^2$ with the following data: $\mu(\boldsymbol{x}) = 1$, $f(\boldsymbol{x}) = 8\pi^2 \sin(2\pi x) \cos(2\pi y)$, and $g(\boldsymbol{x}) = \sin(2\pi x) \cos(2\pi y)$, whose exact solution is $u(\boldsymbol{x}) = \sin(2\pi x) \cos(2\pi y)$. To solve this problem we use the functions contained in `Laplacian`. We set up these data in `InputData/DataTestLap.m`, and fix the number of the element mesh $N_{el} = 30$, the polynomial approximation degree $\ell_\kappa = 3$ for any $\kappa \in \mathcal{T}_h$, and the penalty constant $C_\alpha = 10$ in (6). Next, we run the simulation using the script `RunMainLaplacian.m`, which calls the main algorithm `MainFunctions/MainLaplacian.m`. As the output of the run we obtain the plots in Figure 2

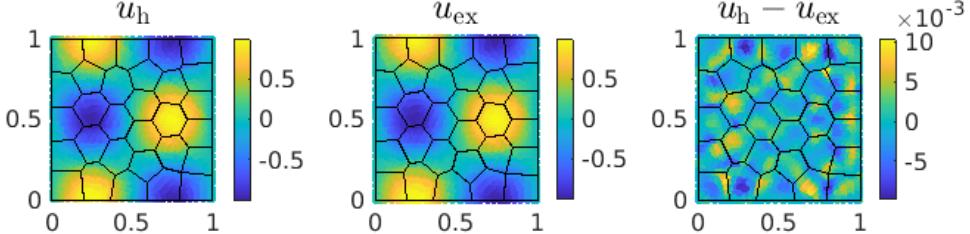


Figure 2: Test case of Section 5.1. Left: computed PolyDG solution u_h using a polygonal mesh with $N_{el} = 30$ elements and a polynomial degree $\ell = 3$. Center: analytical solution u_{ex} . Right: the difference between numerical and analytical solutions.

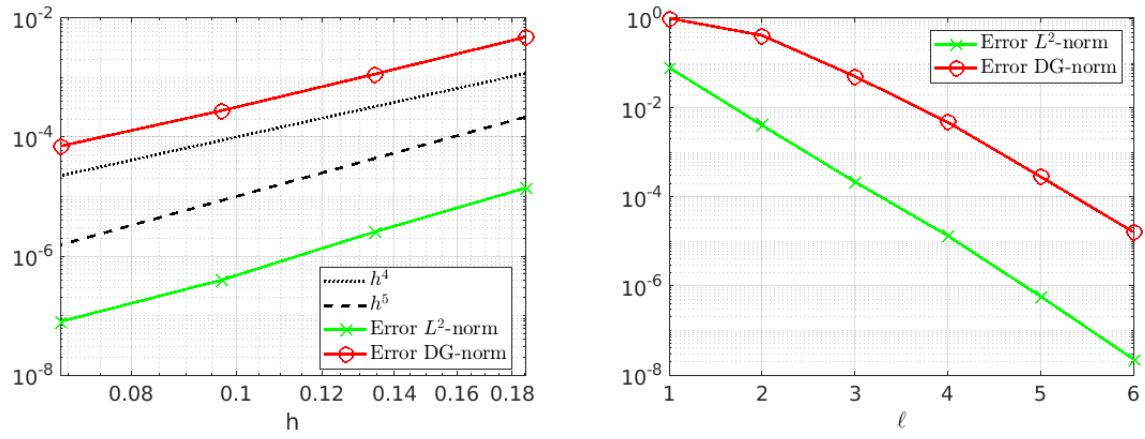


Figure 3: Test case of Section 5.1. Left: computed errors $\|u - u_h\|_{dG}$ and $\|u - u_h\|_{L^2(\Omega)}$ as a function of the mesh size h by fixing the polynomial degree $\ell = 4$. Right: computed errors $\|u - u_h\|_{dG}$ and $\|u - u_h\|_{L^2(\Omega)}$ as a function of the polynomial degree $\ell = 4$ by fixing the number of mesh element $N_{el} = 100$.

showing the computed solution u_h (left), the analytical solution u_{ex} (center), and the arithmetic difference between the two (right). Moreover, the output structure **Error** contains the following fields: **Nel** = 30 (number of mesh elements), **h** = 0.3235 (mesh size), **p** = 3 (polynomial approximation degree), **L2** = 0.0027 (L^2 -norm of the error), **dG** = 0.3349 (dG -norm of the error). To verify the convergence rates of the PolyDG solution u_h in (5) we use two different scripts: `RunhConvergenceLaplacian.m` and `RunpConvergenceLaplacian.m` accounting for the h -convergence (mesh size) and the ℓ -convergence (polynomial degree) respectively. This time, we set up the data in the external script

`InputData/DataConvTestLap.m`. Concerning the previous one, four mesh with decreasing granularity h are provided in input within the field `Data.meshfileseq`. As the output of the aforementioned scripts we obtain the plot in Figure 3. In particular, on the left, we can observe the convergence of the PolyDG solution obtained with $\ell_\kappa = 4$ for any $\kappa \in \mathcal{T}_h$ for the L^2 - and dG -norms, confirming the theoretical results in (8). On the right, the exponential convergence with respect to the polynomial degree ℓ is also shown, cf. (8), by fixing $N_{el} = 100$. For assessing the performance of `lymph` in terms of computational time, we consider the Poisson problem solved on a Cartesian grid made of $N = 19600$ elements using a polynomial degree $\ell = 5$. These choices of discretization parameters lead to 411,600 degrees of freedom. Table 1 presents the computational times, averaged over five simulations. Specifically, we compare the performance of the matrix assembly phase using the quadrature-free and subtriangulation strategies for numerical integration. We observe that using the quadrature-free approach saves approximately 20% of computational time during the matrix assembly phase. Additionally, this approach becomes increasingly

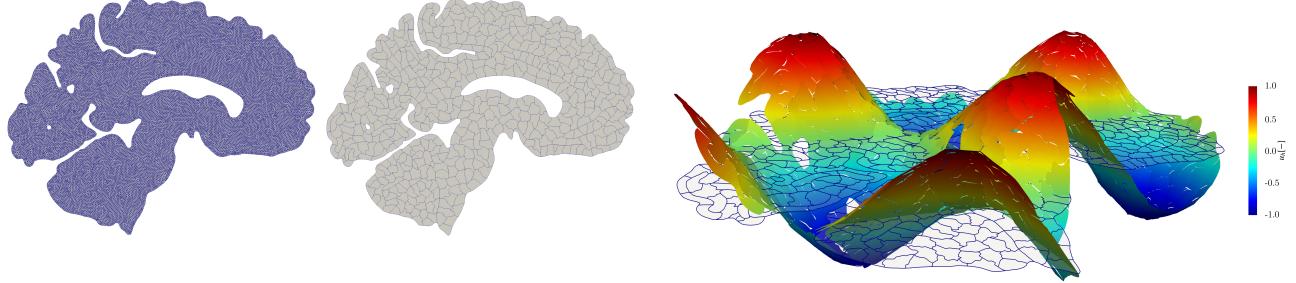


Figure 4: Test case of Section 5.1. Triangular mesh of brain section (left), agglomerated polygonal mesh (center), and computed PolyDG solution on the polygonal grid with $\ell = 1$ (right).

advantageous for general meshes, as it becomes more efficient than the subtriangulation strategy as the number of vertices in the polygonal elements increases, which is the case with agglomerated grids. Table 1 also reports the averaged computational times for assembling the right-hand side and solving the corresponding algebraic system. It is important to note that for the right-hand side of the problem, the quadrature-free approach cannot be used as-is, because we need to integrate functions that, in general, are not polynomials. Extending the quadrature-free approach to non-polynomial functions is an open question and will be the subject of further investigation. The numerical simulations have been performed as a *serial* job, using the Kami cluster (40 computing nodes configured as follows: **CPU** 2x AMD EPYC 7413 24-Core Processor, **RAM** 512Gb) at the Department of Mathematics, Politecnico di Milano.

Mat. assembly (QF)	Mat. assembly (ST)	RHS assembly	Linear system	.csv-file saving	.vtk-file saving
25.6500 s	31.7024 s	5.8172 s	24.7120 s	9.3620 s	13.2213 s

Table 1: Test case of Section 5.1. Computational times for a test case with 411,600 **dofs** ($N = 19600$, $\ell = 5$). We compare the quadrature-free (QF) and subtriangulation (ST) strategies for the numerical evaluation of integrals during matrix assembly. We also report the computational time for the assembly of the right-hand side (RHS) as well as for solving the algebraic system. For the output, we consider two different formats: **.csv** and **.vtk** files.

5.1.2 Advantages of polygonal meshes

As an application of the method, we compare the solution of the Poisson problem on a complicated domain, to show the advantages of the proposed solver on general (agglomerated) polygons. In particular, we consider problem (4) with the following data: $\mu(\mathbf{x}) = 1$, $f(\mathbf{x}) = 400\pi^2 \sin(20\pi x) \cos(20\pi y)$, and $g(\mathbf{x}) = \sin(20\pi x) \cos(20\pi y)$, whose exact solution is $u(\mathbf{x}) = \sin(20\pi x) \cos(20\pi y)$. The domain Ω has been derived from a magnetic resonance image of the brain (from the OASIS-3 database [57]) and it is firstly meshed with 42 891 triangular elements (see Figure 4-left). Then we agglomerate the mesh into 534 polygons (see Figure 4-center), which can capture all the geometric features of the geometry, but with a reduced number of elements in the underlying computational mesh.

We solve the problem using the triangular mesh with different polynomial degrees $\ell = 1, 2, 3$ and the agglomerated one with $\ell = 1, \dots, 7$. In Figure 4 (right), we show the solution obtained on the agglomerated grid with $\ell = 1$. Moreover, we compare the performance of the solution algorithm, by visualizing the approximation errors versus the number of DOFs (see Figure 5 left) and versus the computational time including the matrix and forcing term assembly and linear system solution (see Figure 5 right). By fixing the level of accuracy, the approximation on the agglomerated grid outperforms the one on the triangular grid. Generally, using a high-order polynomial on an agglomerated mesh is more advantageous than

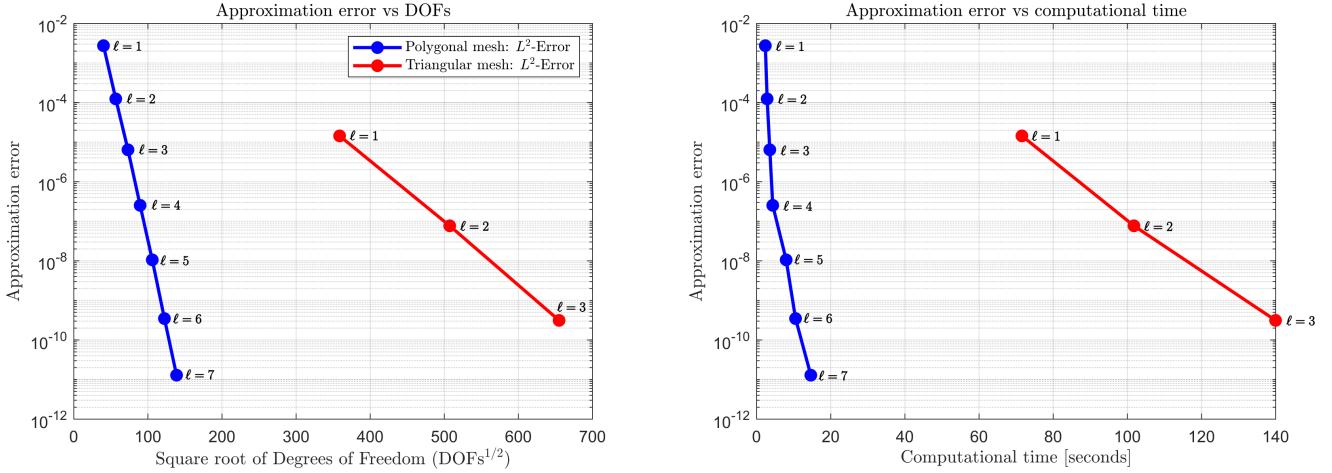


Figure 5: Test case of Section 5.1. Computed approximation errors in the L^2 -norm on the triangular (red) and agglomerated (blue) meshes in Figure 4. Computed errors versus the total number of degrees of freedom (left), and versus the computational time (right).

using a low-order approximation on the underlying triangular grid, at least whenever the exact solution is sufficiently regular.

5.2 The heat equation

Let us consider a polygonal domain $\Omega \subset \mathbb{R}^2$, we denote by $\Gamma = \partial\Omega$ its boundary with an outward normal unit vector \mathbf{n} . On the boundary, we assume to impose Dirichlet boundary conditions ($u = g$). Given a sufficiently regular external load f and initial data u_0 , the heat equation in $\Omega \times (0, T]$ is given by

$$\frac{\partial u}{\partial t} - \nabla \cdot (\mu \nabla u) = f, \quad \text{in } \Omega \times (0, T],$$

with initial condition $u = u_0$, in $\Omega \times \{0\}$. The PolyDG formulation [29] reads: for any time $t \in (0, T]$ find $u_h = u_h(t) \in V_h^\ell$ such that

$$\sum_{\kappa \in \mathcal{T}_h} (\dot{u}_h, v_h)_\kappa + a_{dG}(u_h, v_h) = \sum_{\kappa \in \mathcal{T}_h} (f, v_h)_\kappa \quad \forall v_h \in V_h^\ell, \quad (9)$$

with initial conditions $u_h = u_{0,h}$, where $u_{0,h}$ is the L^2 -projection of the initial data on V_h^ℓ . In (9) the bilinear form $a_{dG}(\cdot, \cdot)$ is defined as in the previous section. Now, by introducing a set of basis functions $\{\varphi_j\}_{j=1}^{N_h}$ for V_h^ℓ we can easily get the following system of first-order differential equations:

$$\begin{cases} M \dot{U}_h(t) + A U_h(t) = F(t) & t \in (0, T], \\ U_h(0) = U_{0,h}, \end{cases} \quad (10)$$

To integrate system (10) in time we apply the θ -method scheme. The numerical simulation is performed using the Crank-Nicolson scheme ($\theta = 1/2$), see, e.g., [58, 59]. We remark that it is possible to couple the `lymph` library with time integration schemes already present in MATLAB (e.g., Runge-Kutta schemes [59]).

Test case with discontinuous boundary conditions

As an application of the presented PolyDG method, we solve with `lymph` the heat equation problem presented in [60]. It considers the parabolic problem with $\mu = 0.1$ and a homogeneous forcing term

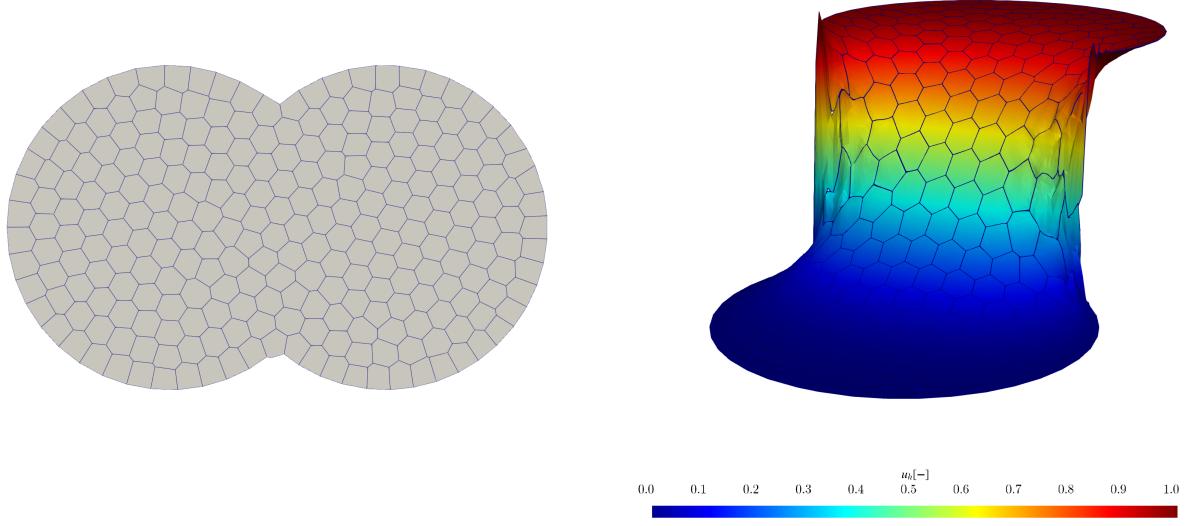


Figure 6: Test case of Section 5.2. Polygonal mesh of the two circles (left) and computed PolyDG solution at time $t = 1.0$.

$f = 0$. The domain Ω is composed of two overlapping circles of radius 0.5 and center $(-0.5, 0)$ and $(0.5, 0)$ respectively. For the numerical discretization of this problem, we construct a polygonal mesh through the PolyMesher software [22] (see Figure 6 left). The mesh we adopt in the simulation is composed of 250 elements.

We consider discontinuous Dirichlet boundary conditions on the top of the model such that $u(\mathbf{x}) = 0$ for $x \leq 0$ and $u(\mathbf{x}) = 1$ for $x > 0$, where $\mathbf{x} = (x, y)$. As shown in the test case reference [60], we obtain that the solution at time $t = 1$ is smoothed inside the computational domain, due to the diffusion process. The discretization in space is performed employing polynomials of degree $\ell = 5$, while the time discretization uses a timestep $\Delta t = 0.002$, and a final time $T = 1$. In Figure 6 we report the snapshot of the solution at the final time $T = 1$. The solution is coherent to what is expected from the literature [60].

5.3 The elastodynamics system

We consider a bounded convex polygonal domain $\Omega \subset \mathbb{R}^2$, we denote by $\Gamma = \partial\Omega$ its boundary with outward normal unit vector \mathbf{n} . The boundary is assumed to be composed of two disjoint portions Γ_D and Γ_N , where Dirichlet ($\mathbf{u} = \mathbf{0}$), and Neumann ($\boldsymbol{\sigma}(\mathbf{u})\mathbf{n} = \mathbf{0}$), conditions are imposed, respectively. Given sufficiently regular external loads \mathbf{f} and \mathbf{g} and initial data \mathbf{u}_0 and \mathbf{v}_0 , cf. [61], the equations of (linear) elastodynamics in $\Omega \times (0, T]$ are given by

$$\rho \frac{\partial^2 \mathbf{u}}{\partial t^2} - \nabla \cdot \boldsymbol{\sigma}(\mathbf{u}) = \mathbf{f}, \quad \text{in } \Omega \times (0, T],$$

with initial conditions $(\mathbf{u}, \frac{\partial \mathbf{u}}{\partial t})(0) = (\mathbf{u}_0, \mathbf{v}_0)$, in Ω . We denote by $\mathbf{u} : \Omega \times [0, T] \rightarrow \mathbb{R}^2$ the displacement vector and by $\boldsymbol{\sigma} : \Omega \times [0, T] \rightarrow \mathbb{S}$ the stress tensor where \mathbb{S} is the space of symmetric, 2×2 , real-valued tensor fields. We assume the an isotropic-linear elasticity constitutive model $\boldsymbol{\sigma}(\mathbf{u}) = 2\mu\boldsymbol{\epsilon}(\mathbf{u}) + \lambda \operatorname{tr}(\boldsymbol{\epsilon}(\mathbf{u}))I$, where $\boldsymbol{\epsilon}(\mathbf{u})$ is the symmetric gradient of \mathbf{u} , I is the identity tensor, $\operatorname{tr}(\cdot)$ is the trace operator, and $\lambda, \mu \in L^\infty(\Omega)$ are Lamé's parameters. The compressional (P) and shear (S) wave velocities of the medium are obtained

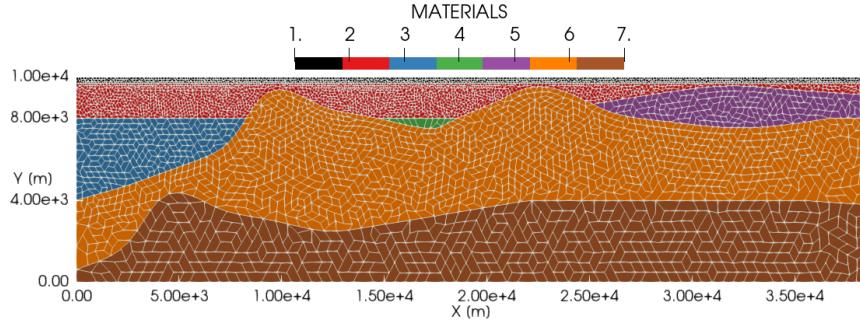


Figure 7: Test case of Section 5.3. Unstructured polygonal grid with mesh spacing of about $h \approx 160$ m for material 1 to $h \approx 1500$ m for material 7; cf. Table 2. Mesh file is available in `Elastodynamics/InputMeshPhysics/MeshEmilia.mat`.

Materials	1	2	3	4	5	6	7
ρ [kg/m ³]	1800	1800	2050	2050	2050	2400	2450
c_S [m/s]	294	450	600	600	600	1515	1600
c_P [m/s]	1321	2024	1920	1920	1920	3030	3200

Table 2: Test case of Section 5.3. Material properties used for the computational domain in Figure 7, cf. also `Elastodynamics/InputData/Elastic/DataTestPhysicsEla.m`

through the relations $c_P = \sqrt{(\lambda + 2\mu)/\rho}$ and $c_S = \sqrt{\mu/\rho}$, respectively. By following [61], we obtain the PolyDG formulation: for any time $t \in (0, T]$ find $\mathbf{u}_h = \mathbf{u}_h(t) \in \mathbf{V}_h^\ell$ such that

$$\sum_{\kappa \in \mathcal{T}_h} (\rho \mathbf{u}, \mathbf{v})_\kappa + a_{dG}^e(\mathbf{u}, \mathbf{v}) = \sum_{\kappa \in \mathcal{T}_h} (\mathbf{f}, \mathbf{v})_\kappa \quad \forall \mathbf{v} \in \mathbf{V}_h^\ell, \quad (11)$$

with initial conditions $(\mathbf{u}_h, \frac{\partial \mathbf{u}_h}{\partial t}) = (\mathbf{u}_{0h}, \mathbf{v}_{0h})$, where $\mathbf{u}_{0,h}$ and $\mathbf{v}_{0,h}$ are the L^2 -projection of the initial data on \mathbf{V}_h^ℓ . In (11) the bilinear forms $a_{dG}^e(\cdot, \cdot)$ is defined as

$$a_{dG}^e(\mathbf{u}, \mathbf{v}) = \sum_{\kappa \in \mathcal{T}_h} (\boldsymbol{\sigma}(\mathbf{u}), \boldsymbol{\epsilon}(\mathbf{v}))_\kappa - \sum_{e \in \mathcal{F}_h \setminus \Gamma_N} \left((\{\boldsymbol{\sigma}(\mathbf{u})\}, [\mathbf{v}])_e + (\{\boldsymbol{\sigma}(\mathbf{v})\}, [\mathbf{u}])_e - (\eta_e [[(\mathbf{u})]], [[\mathbf{v}]])_e \right) \quad \forall \mathbf{u}, \mathbf{v} \in \mathbf{V}_h^\ell,$$

with η_e as in [61, eq. (9)]. Now, by introducing a set of basis functions $\{\boldsymbol{\varphi}_j^1, \boldsymbol{\varphi}_j^2\}_{j=1}^{N_h}$ for \mathbf{V}_h^ℓ we can easily get the following system of second order differential equations:

$$\begin{cases} M \ddot{\mathbf{U}}_h(t) + A \mathbf{U}_h(t) = \mathbf{F}(t) & t \in (0, T], \\ \dot{\mathbf{U}}_h(0) = \mathbf{V}_{0h}, \quad \mathbf{U}_h(0) = \mathbf{U}_{0h}, \end{cases} \quad (12)$$

To integrate system (12) in time we apply the Newmark β -scheme, with $\beta = \frac{1}{4}$ and $\gamma = \frac{1}{2}$.

As an application of the presented PolyDG method, we solve with `lymph` the wave propagation problem presented in [49, Section 5.4.3.2]. It considers the elastic wave propagation $\Omega = (0, 38.4)$ km $\times (0, 10)$ km representing an idealized bidimensional Earth's cross-section, see Figure 7.

We consider homogeneous Neumann conditions on the top of the model ($\boldsymbol{\sigma} \mathbf{n} = \mathbf{0}$) whereas homogeneous Dirichlet conditions ($\mathbf{u} = \mathbf{0}$) are set on the remaining boundaries. The bottom and the lateral boundaries are set far enough from the point source to prevent any reflections from the boundaries of the waves of interest. We simulate a double-couple moment source load of the form $\mathbf{f}(\mathbf{x}, t) = -I \cdot \nabla \delta(\mathbf{x} - \mathbf{x}_s) S(t)$, where $\delta(\mathbf{x} - \mathbf{x}_s)$ is the Dirac delta distribution centered in $\mathbf{x}_s = (19432, 7800)$ m and $S(t) = (1 - 8\pi^2(t - 0.5)^2)e^{-4\pi^2(t-0.5)^2}$ is the source time function. We assign constant material properties within each region as described in Table 2. The computational domain is discretized using an unstructured grid consisting of 4870 (agglomerated) polygonal elements, with a mesh size varying from

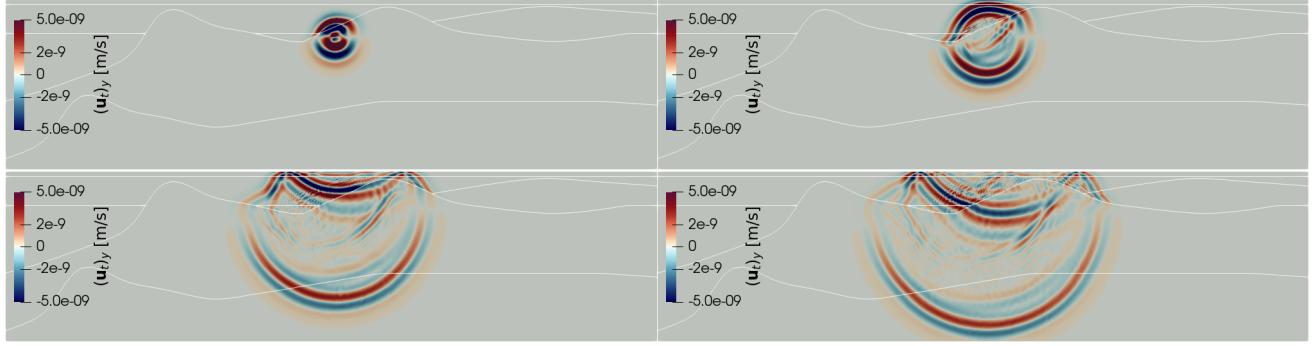


Figure 8: Test case of Section 5.3. Snapshots of the computed vertical velocity $(\mathbf{u}_t)_y$ at different times $t = 0.75$ (top-left), $t = 1.25$ (top-right), $t = 2.25$ (bottom-left), $t = 2.75$ (bottom-right). Due to the material heterogeneity's, high oscillations and perturbations of the wavefront can be observed, as well as the effect of the free surface on top.

$h \approx 160$ m for material 1 to $h \approx 1500$ m for material 7; cf. Table 2. We consider also a polynomial degree $\ell = 5$, $\Delta t = 0.001$ s, and a final time $T = 4$ s. In Figure 8 we report a set of snapshots of the computed vertical velocity field $(\mathbf{u}_t)_y$. The discontinuities between the mechanical properties of the materials produce oscillations and perturbations on the wavefront; surface waves are visible. Finally, we report in Table 3 the computing time of the presented test (averaged over 5 different simulations). The numerical simulation has been performed as a *serial* job, using the **Kami** cluster described before.

Mat. assembly (QF)	Mat. assembly (ST)	RHS assembly	Linear system	.csv-file saving	.vtk-file saving
10.9928 s	13.0421 s	2.3805 s	62.4727 s	3.4967 s	4.3340 s

Table 3: Test case of Section 5.3. Computational times for a test case with 409,080 **dofs** ($N = 4870$, $\ell = 5$). We compare the quadrature-free (QF) and subtriangulation (ST) strategies for the numerical evaluation of integrals during matrix assembly. We also report the computational time for the assembly of the right-hand side (RHS) as well as for solving the algebraic system. For the output, we consider two different formats: **.csv** and **.vtk** files.

5.4 A multi-physics problem: the poroelasto-acoustic system

In this last application, we present a wave propagation problem in a coupled poroelastic-acoustic domain $\Omega = \Omega_p \cup \Omega_a$, where Ω_p and Ω_a are the poroelastic and acoustic medium, respectively. These domains share an interface denoted as Γ_I .

In the fluid domain Ω_a we consider the following model

$$\begin{cases} c^{-2} \frac{\partial^2 \varphi_a}{\partial t^2} - \rho_a^{-1} \nabla \cdot (\rho_a \nabla \varphi_a) = f_a, & \text{in } \Omega_a \times (0, T], \\ \varphi_a = g_a, & \text{on } \Gamma_{aD} \times (0, T], \\ \nabla \varphi_a \cdot \mathbf{n}_a = 0, & \text{on } \Gamma_{aN} \times (0, T], \\ \varphi_a(0) = \varphi_0, \quad \frac{\partial \varphi_a}{\partial t}(0) = \psi_0, & \text{in } \Omega_a, \end{cases} \quad (13)$$

where φ_a is the acoustic potential, $c > 0$ is the wave speed, $\rho_a > 0$ is the medium density, and f_a, g_a, φ_0 and ψ_0 are regular enough data. We consider the boundary $\Gamma_a = \Gamma_{aD} \cup \Gamma_{aN}$ of Ω_a to be sufficiently regular, with the normal \mathbf{n}_a outward pointing, and to be decomposed into two distinct portions denoted by Γ_{aD} (resp. Γ_{aN}) where Dirichlet (resp. Neumann) conditions are imposed.

In the porous domain Ω_p we consider the system [62]:

$$\begin{cases} \rho_p \frac{\partial^2 \mathbf{u}_p}{\partial t^2} + \rho_f \frac{\partial^2 \mathbf{u}_f}{\partial t^2} - \nabla \cdot \boldsymbol{\sigma}_p(\mathbf{u}_p, \mathbf{u}_f) = \mathbf{f}_p, & \text{in } \Omega_p \times (0, T], \\ \rho_f \frac{\partial^2 \mathbf{u}_p}{\partial t^2} + \rho_w \frac{\partial^2 \mathbf{u}_f}{\partial t^2} + \frac{\eta}{k} \frac{\partial \mathbf{u}_f}{\partial t} + \nabla p_p(\mathbf{u}_p, \mathbf{u}_f) = \mathbf{f}_f, & \text{in } \Omega_p \times (0, T], \\ \boldsymbol{\sigma}_p(\mathbf{u}_p, \mathbf{u}_f) = \boldsymbol{\sigma}_e(\mathbf{u}_p) - \beta p_p(\mathbf{u}_p, \mathbf{u}_f) I, & \text{in } \Omega_p \times (0, T], \\ p_p(\mathbf{u}_p, \mathbf{u}_f) = -m(\beta \nabla \cdot \mathbf{u}_p + \nabla \cdot \mathbf{u}_f), & \text{in } \Omega_p \times (0, T], \\ (\mathbf{u}_p, \mathbf{u}_f) = (\mathbf{g}_p, \mathbf{g}_f), & \text{on } \Gamma_p \times (0, T], \\ \mathbf{u}_p(0) = \mathbf{u}_{p0}, \quad \frac{\partial \mathbf{u}_p}{\partial t}(0) = \mathbf{v}_{p0}, & \text{in } \Omega_p, \\ \mathbf{u}_f(0) = \mathbf{u}_{f0}, \quad \frac{\partial \mathbf{u}_f}{\partial t}(0) = \mathbf{v}_{f0}, & \text{in } \Omega_p. \end{cases} \quad (14)$$

Here, \mathbf{u}_p and \mathbf{u}_f represent the displacements of the solids and the filtration, respectively, ρ_p is the average density given by $\rho_p = \phi \rho_f + (1 - \phi) \rho_s$, where $\rho_s > 0$ is the solid density, $\rho_f > 0$ is the density of the saturated fluid, ρ_w is defined as $\rho_w = \frac{a}{\phi} \rho_f$, with porosity ϕ satisfying $0 < \phi_0 \leq \phi \leq \phi_1 < 1$, and the tortuosity $a > 1$ measures the deviation of the fluid paths from straight streamlines. The dynamic viscosity of the fluid is given by $\eta > 0$, the absolute permeability by $k > 0$, while the Biot–Willis coefficient β and the Biot modulus m are such that $\phi < \beta \leq 1$ and $m \geq m_0 > 0$. In (14), $\mathbf{f}_p, \mathbf{f}_f, \mathbf{g}_p, \mathbf{g}_f, \mathbf{u}_{p0}, \mathbf{v}_{p0}, \mathbf{u}_{f0}$, and \mathbf{v}_{f0} are given (regular enough) data. Moreover, the elastic tensor $\boldsymbol{\sigma}_e(\mathbf{u}_p)$ is defined by Hooke's law introduced in Section 5.3. Finally, we suppose the boundary Γ_p of Ω_p to be sufficiently regular, having outward pointing unit normal \mathbf{n}_p .

On the shared interface Γ_I , the following coupling conditions are prescribed

$$\begin{cases} -\boldsymbol{\sigma}_p(\mathbf{u}_p, \mathbf{u}_f) \mathbf{n}_p = \rho_a \frac{\partial \varphi_a}{\partial t} \mathbf{n}_p, & \text{on } \Gamma_I \times (0, T], \\ p_p(\mathbf{u}_p, \mathbf{u}_f) = \rho_a \frac{\partial \varphi_a}{\partial t}, & \text{on } \Gamma_I \times (0, T], \\ -\left(\frac{\partial \mathbf{u}_p}{\partial t} + \frac{\partial \mathbf{u}_f}{\partial t}\right) \cdot \mathbf{n}_p = \nabla \varphi_a \cdot \mathbf{n}_p, & \text{on } \Gamma_I \times (0, T], \end{cases} \quad (15)$$

expressing the continuity of normal stresses, continuity of pressure, and conservation of mass, respectively. We refer the reader to [36] for a detailed description of the problem as well its numerical discretization and analysis. For this problem the *polytopic* mesh $\mathcal{T}_h = \mathcal{T}_h^p \cup \mathcal{T}_h^a$, where $\mathcal{T}_h^\# = \{\kappa \in \mathcal{T}_h : \bar{\kappa} \subseteq \bar{\Omega}_\#\}$, with $\# = \{p, a\}$. Implicit in this decomposition is the assumption that the meshes \mathcal{T}_h^a and \mathcal{T}_h^p are aligned with Ω_a and Ω_p , respectively. Moreover, we denote by $V_{\#h}^\ell = V_h^\ell|_{\Omega_\#}$, with $\# = \{p, a\}$. As explained in [36], the space discretization of (13)-(14)-(15) with a PolydG method leads to the following problem: for $t \in (0, T]$, find $(\mathbf{u}_{ph}, \mathbf{u}_{fh}, \varphi_{ah})(t) \in \mathbf{V}_{ph}^\ell \times \mathbf{V}_{ph}^\ell \times V_{ah}^\ell$, s.t.

$$\begin{aligned} & \sum_{\kappa \in \mathcal{T}_{ph}} (\rho \ddot{\mathbf{u}}_{ph} + \rho_f \ddot{\mathbf{u}}_{fh}, \mathbf{v}_h)_\kappa + (\rho_f \ddot{\mathbf{u}}_{ph} + \rho_w \ddot{\mathbf{u}}_{fh}, \mathbf{z}_h)_\kappa + \sum_{\kappa \in \mathcal{T}_{ah}} (c^{-2} \ddot{\varphi}_{ah}, \psi_h)_\kappa + \sum_{\kappa \in \mathcal{T}_{ph}} (\eta k^{-1} \dot{\mathbf{u}}_{fh}, \mathbf{z}_h)_\kappa \\ & \quad + \mathcal{A}_h^e(\mathbf{u}_{ph}, \mathbf{v}_h) + \mathcal{A}_h^p(\beta \mathbf{u}_{ph} + \mathbf{u}_{fh}, \beta \mathbf{v}_h + \mathbf{z}_h) + \mathcal{A}_h^a(\varphi_{ah}, \psi_h), \\ & + \sum_{e \in \Gamma_I} (\rho_a \dot{\varphi}_{ah}, (\mathbf{v}_h + \mathbf{z}_h) \cdot \mathbf{n}_p)_e - \sum_{e \in \Gamma_I} ((\dot{\mathbf{u}}_{ph} + \dot{\mathbf{u}}_{fh}) \cdot \mathbf{n}_p, \rho_a \psi_h)_e = \sum_{\kappa \in \mathcal{T}_{ph}} (\mathbf{f}_p, \mathbf{v}_h)_\kappa + (\mathbf{f}_f, \mathbf{z}_h)_\kappa + \sum_{\kappa \in \mathcal{T}_{ah}} (f_a, \psi_h)_\kappa \end{aligned} \quad (16)$$

for any $(\mathbf{v}_h, \mathbf{z}_h, \psi_h) \in \mathbf{V}_{ph}^\ell \times \mathbf{V}_{ph}^\ell \times V_{ah}^\ell$. Note that in (16) we have used the notation \dot{u} (resp. \ddot{u}) to indicate the first (resp. second) time derivative of a generic function u . The bilinear forms $\mathcal{A}_h^e(\cdot, \cdot)$, $\mathcal{A}_h^p(\cdot, \cdot)$, and

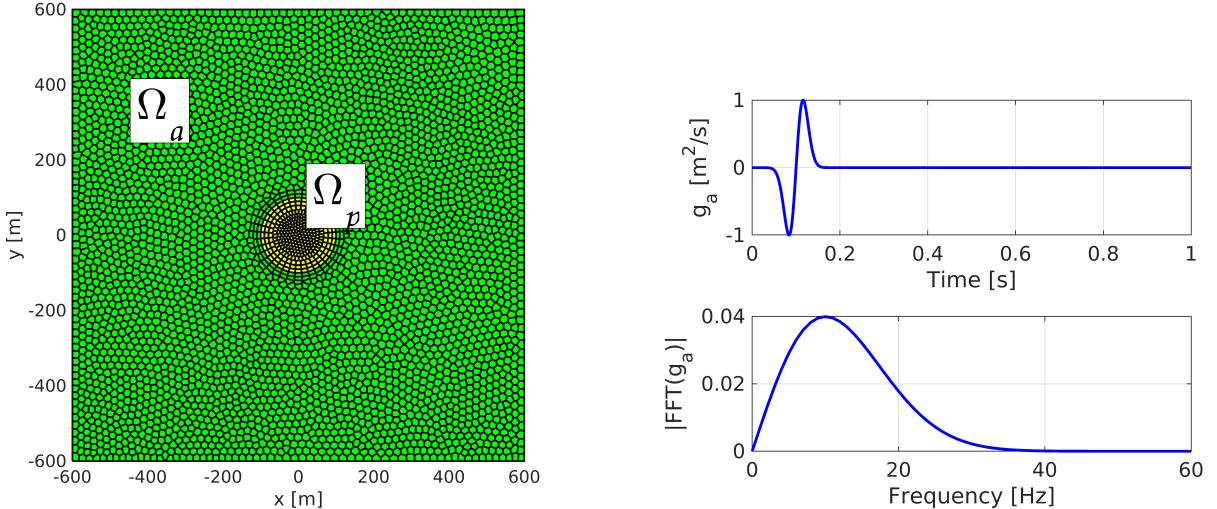


Figure 9: Test case of Section 5.4. Left: circular porous domain Ω_p (yellow) surrounded by an acoustic medium Ω_a (green). The mesh is made by 4287 polygonal elements divided into 3979 for Ω_a ($h \approx 36.5$ m) and 308 for Ω_p ($h \approx 11.8$ m). Right: Dirichlet boundary condition applied on the domain's bottom edge (top), the absolute value of its Fourier transform (bottom).

$\mathcal{A}_h^a(\cdot, \cdot)$ appearing in (16) are defined as in [36, eq. (3.4)]. Now, by introducing a set of basis functions for \mathbf{V}_{ph}^ℓ and \mathbf{V}_{ah}^ℓ and denoting by $(U_{pf}, U_{fh}, \Phi_{ah})$ the vector of the expansion coefficients in the chosen basis of the unknowns \mathbf{u}_{ph} , \mathbf{u}_{fh} and φ_{ah} , respectively, we obtain the following system of ordinary differential equations:

$$\begin{bmatrix} \mathbf{M}_\rho^p & \mathbf{M}_{\rho_f}^p & 0 \\ \mathbf{M}_{\rho_f}^p & \mathbf{M}_{\rho_w}^p & 0 \\ 0 & 0 & \mathbf{M}_{c^{-2}}^a \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{U}}_{ph} \\ \ddot{\mathbf{U}}_{fh} \\ \ddot{\Phi}_{ah} \end{bmatrix} + \begin{bmatrix} 0 & 0 & \mathbf{C}^p \\ 0 & \mathbf{B} & \mathbf{C}^p \\ \mathbf{C}^a & \mathbf{C}^a & 0 \end{bmatrix} \begin{bmatrix} \dot{\mathbf{U}}_{ph} \\ \dot{\mathbf{U}}_{fh} \\ \dot{\Phi}_{ah} \end{bmatrix} + \begin{bmatrix} \mathbf{A}^e + \mathbf{A}_{\beta^2}^p & \mathbf{A}_\beta^p & 0 \\ \mathbf{A}_\beta^p & \mathbf{A}^p & 0 \\ 0 & 0 & \mathbf{A}^a \end{bmatrix} \begin{bmatrix} \mathbf{U}_{ph} \\ \mathbf{U}_{fh} \\ \Phi_{ah} \end{bmatrix} = \begin{bmatrix} \mathbf{F}^p \\ \mathbf{F}^f \\ \mathbf{F}^a \end{bmatrix} \quad (17)$$

with initial conditions $(U_{ph}, \dot{U}_{ph})(0) = (U_{p0}, V_{p0})$, $(U_{fh}, \dot{U}_{fh})(0) = (U_{f0}, V_{f0})$, and $(\Phi_{ah}, \dot{\Phi}_{ah})(0) = (\Phi_0, \Psi_0)$. In the above system \mathbf{F}^p , \mathbf{F}^f and \mathbf{F}^a are the vector representations of the right-hand side of (16). To integrate in time system (17) we use the Newmark scheme as in Section 5.3. We refer the interested reader to [36] for all the details.

We point out that the assembly strategy of `lymph` allows for system (17) to be assembled with almost no additional complexity with respect to assembling each of the involved physics separately. Indeed, the `Physics/PoroElastoAcoustics/Assembly` folder contains a sub-folder with the routines to assemble the terms corresponding to each of the involved physics, an additional one to assemble the coupling terms, and a function to combine all the terms in the monolithic matrices.

We now consider the problem of acoustic scattering by a porous cylinder. The domain is given by $\Omega = (-600, 600)$ m², see Figure 9, cf. also [63], where a circular porous cylinder Ω_p is surrounded by the acoustic medium Ω_a . More precisely, we consider a circular interface Γ_I of radius 100 m centered at (0,0), dividing Ω into a circular porous domain Ω_p (yellow, in Figure 9) surrounded by an acoustic medium Ω_a (green, in Figure 9). The computational domain is discretized by 4287 polygonal elements having a characteristic size $h \approx 36$ m, and we consider a space polynomial degree equal to 4. A locally refined mesh is employed around the interface to compute precisely the various wave conversions, cf. Figure 9. The source is an acoustic plane wave given as a Dirichlet condition on the bottom boundary,

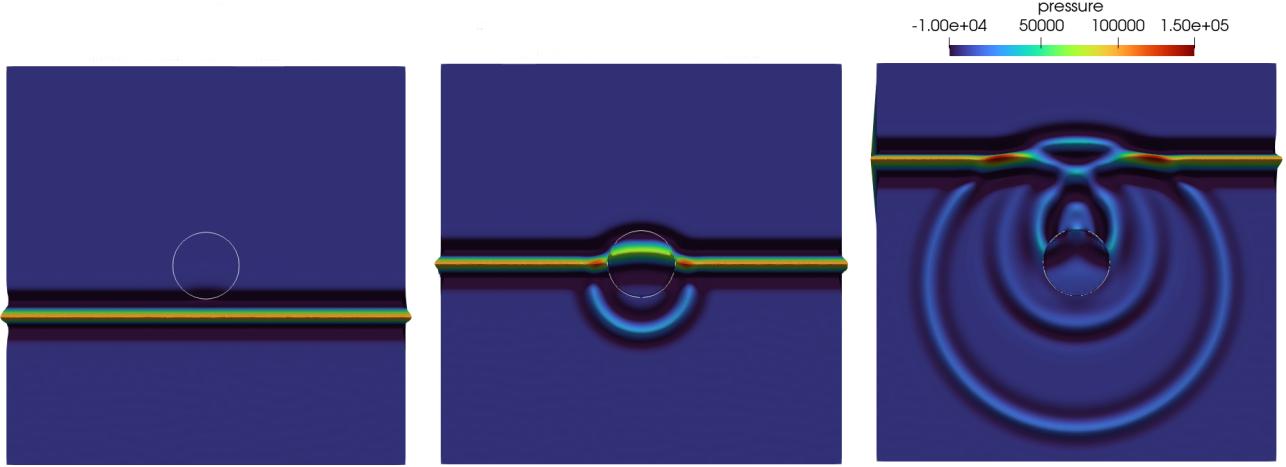


Figure 10: Test case of Section 5.4. Snapshots of the computed pressure field at different time instants: $t = 0.4 \text{ s}$ (top-left), $t = 0.5 \text{ s}$ (top-right), $t = 0.6 \text{ s}$ (bottom-left), and $t = 0.7 \text{ s}$ (bottom-right).

i.e., $\Gamma_{aD} = (-600, 600) \times \{-600\}$, cf. Figure 9 (right). The pulse is a Gaussian wavelet $g_a(t) = 2\pi f_p \sqrt{e}(t - 1/f_p)e^{-2(\pi f_p)^2(t-1/f_p)^2}$, having a peak frequency $f_p = 10 \text{ Hz}$. A sound soft condition is enforced on the remaining boundaries, i.e., $\nabla \varphi_a \cdot \mathbf{n}_a = 0$ on Γ_{aN} . The physical parameters for this test case are listed in Table 4. In Figure 10 we plot the computed pressure field $p_a = \rho_a \dot{\varphi}_a(t)$ in Ω_a and

Fluid	Fluid density	ρ_f	1000	kg/m^3
		ρ_a	1000	kg/m^3
	Wave velocity	c	1500	m/s
	Dynamic viscosity	η	$1.05 \cdot 10^{-3}$	$\text{Pa} \cdot \text{s}$
Grain	Solid density	ρ_s	2690	kg/m^3
	Shear modulus	μ	$1.86 \cdot 10^9$	Pa
Matrix	Porosity	ϕ	0.38	
	Tortuosity	a	1.8	
	Permeability	k	$2.79 \cdot 10^{-11}$	m^2
	Lamé coefficient	λ	$1.20 \cdot 10^8$	Pa
	Biot's coefficient	m	$5.34 \cdot 10^9$	Pa
	Biot's coefficient	β	0.95	

Table 4: Test case of Section 5.4. Physical parameters for the poroelastic-acoustic test case.

$p_p = -m(\beta \nabla \cdot \mathbf{u}_p + \nabla \cdot \mathbf{u}_f)$ in Ω_p at different time instants, considering a final time $T = 1 \text{ s}$, and a time step $\Delta t = 0.001 \text{ s}$. It is possible to observe that the wave is moving from the bottom to the top of the domain, impacting the porous cylinder, and producing a scattered circular wavefield directed backwards. The numerical results are aligned with the corresponding ones presented in [63].

6 Conclusions

This paper presents **lymph**, a general-purpose Matlab library for the approximate solution of multi-physics differential problems in two-dimensions. For the spatial discretization of the underlying differential systems, **lymph** library is based on high-order discontinuous Galerkin methods on polytopal grids, making its use attractive for several areas of engineering and applied sciences applications. The target of this paper is to introduce the library step-by-step and to show the potential of the software, starting from the solution of classical differential problems. As **lymph** is a user-friendly, general-purpose library, the

authors think that its use can be widely extended to other engineering applications. Interesting future developments of this work include the design of more robust and flexible (agglomeration-driven) mesh generation algorithms and the introduction of hp -refinement approaches, and the extension to three-dimension.

7 Acknowledgements

We thank the Editors Lin-bo Zhang and Wolfgang Bangerth, and the anonymous Reviewers for their valuable insights and comments, which greatly contributed to improving the content of the present work. We acknowledge Prof. Paul Houston and Dr. Giorgio Pennesi for the original implementation of the quadrature-free approach [52] adopted in the library. The brain MRI images were provided by OASIS-3: Longitudinal Multimodal Neuroimaging: Principal Investigators: T. Benzinger, D. Marcus, J. Morris; NIH P30 AG066444, P50 AG00561, P30 NS09857781, P01 AG026276, P01 AG003991, R01 AG043434, UL1 TR000448, R01 EB009352. AV-45 doses were provided by Avid Radiopharmaceuticals, a wholly-owned subsidiary of Eli Lilly. This work received funding from the European Union (ERC SyG, NEMESIS, project number 101115663). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them. PFA, IF, IM, have been partially supported by ICSC–Centro Nazionale di Ricerca in High Performance Computing, Big Data, and Quantum Computing funded by European Union–NextGenerationEU. PFA and SB have been partially funded by MUR, PRIN 2020 research grant n. 20204LN5N5. All the authors are members of INdAM-GNCS. The work of IM has been partially supported by the INdAM-GNCS project CUP E53C22001930001.

References

- [1] 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*, vol. 231, no. 1, pp. 45–65, 2012.
- [2] P. F. Antonietti, S. Giani, and P. Houston, “ hp -version composite discontinuous Galerkin methods for elliptic problems on complicated domains,” *SIAM Journal on Scientific Computing*, vol. 35, no. 3, pp. A1417–A1439, 2013.
- [3] A. Cangiani, E. H. Georgoulis, and P. Houston, “ hp -version discontinuous Galerkin methods on polygonal and polyhedral meshes,” *Mathematical Models and Methods in Applied Sciences*, vol. 24, no. 10, pp. 2009–2041, 2014.
- [4] A. Cangiani, Z. Dong, E. H. Georgoulis, and P. Houston, “ hp -version discontinuous Galerkin methods for advection-diffusion-reaction problems on polytopic meshes,” *ESAIM: Mathematical Modelling and Numerical Analysis*, vol. 50, no. 3, pp. 699–725, 2016.
- [5] A. Cangiani, Z. Dong, E. H. Georgoulis, and P. Houston, *hp-version discontinuous Galerkin methods on polytopic meshes*. SpringerBriefs in Mathematics, Cham: Springer International Publishing, 2017.
- [6] L. Beirão Da Veiga, F. Brezzi, A. Cangiani, G. Manzini, L. D. Marini, and A. Russo, “Basic principles of virtual element methods,” *Mathematical Models and Methods in Applied Sciences*, vol. 23, no. 01, pp. 199–214, 2013.
- [7] L. Beirão da Veiga, F. Brezzi, and L. D. Marini, “Virtual elements for linear elasticity problems,” *SIAM J. Numer. Anal.*, vol. 51, no. 2, pp. 794–812, 2013.

- [8] L. Beirão da Veiga, F. Brezzi, L. D. Marini, and A. Russo, “The hitchhiker’s guide to the virtual element method,” *Math. Models Methods Appl. Sci.*, vol. 24, no. 8, pp. 1541–1573, 2014.
- [9] L. Beirão da Veiga, F. Brezzi, L. D. Marini, and A. Russo, “Virtual element method for general second-order elliptic problems on polygonal meshes,” *Math. Models Methods Appl. Sci.*, vol. 26, no. 4, pp. 729–750, 2016.
- [10] L. Beirão da Veiga, F. Brezzi, F. Dassi, L. D. Marini, and A. Russo, “A family of three-dimensional virtual elements with applications to magnetostatics,” *SIAM Journal on Numerical Analysis*, vol. 56, no. 5, pp. 2940–2962, 2018.
- [11] L. Beirão da Veiga, A. Russo, and G. Vacca, “The virtual element method with curved edges,” *ESAIM: Mathematical Modelling and Numerical Analysis*, vol. 53, no. 2, pp. 375–404, 2019.
- [12] D. A. Di Pietro, A. Ern, and S. Lemaire, “An arbitrary-order and compact-stencil discretization of diffusion on general meshes based on local reconstruction operators,” *Computational Methods in Applied Mathematics*, vol. 14, no. 4, pp. 461–472, 2014.
- [13] D. A. Di Pietro and J. Droniou, “A hybrid high-order method for Leray–Lions elliptic equations on general meshes,” *Mathematics of Computation*, vol. 86, no. 307, pp. 2159–2191, 2017.
- [14] D. A. Di Pietro and A. Ern, “Hybrid high-order methods for variable-diffusion problems on general meshes,” *Comptes Rendus Mathematique*, vol. 353, no. 1, pp. 31–34, 2015.
- [15] B. Cockburn, B. Dong, and J. Guzmán, “A superconvergent LDG-hybridizable Galerkin method for second-order elliptic problems,” *Mathematics of Computation*, vol. 77, no. 264, pp. 1887–1916, 2008.
- [16] B. Cockburn, J. Gopalakrishnan, and R. Lazarov, “Unified hybridization of discontinuous galerkin, mixed, and continuous galerkin methods for second order elliptic problems,” *SIAM Journal on Numerical Analysis*, vol. 47, no. 2, pp. 1319–1365, 2009.
- [17] B. Cockburn, J. Guzmán, and H. Wang, “Superconvergent discontinuous Galerkin methods for second-order elliptic problems,” *Mathematics of Computation*, vol. 78, no. 265, pp. 1–24, 2009.
- [18] L. Zhao and E. J. Park, “A staggered discontinuous galerkin method of minimal dimension on quadrilateral and polygonal meshes,” *SIAM Journal on Scientific Computing*, vol. 40, no. 4, pp. A2543–A2567, 2018.
- [19] L. Zhao, E. J. Park, and D. W. Shin, “A staggered dg method of minimal dimension for the stokes equations on general meshes,” *Computer Methods in Applied Mechanics and Engineering*, vol. 345, pp. 854–875, 2019.
- [20] L. Zhao and E. J. Park, “A new hybrid staggered discontinuous galerkin method on general meshes,” *Journal of Scientific Computing*, vol. 82, no. 1, p. 12, 2020.
- [21] M. Botsch, S. Steinberg, S. Bischoff, L. Kobbelt, and R. Aachen, “Openmesh - a generic and efficient polygon mesh data structure,” 2002.
- [22] C. Talischi, G. H. Paulino, A. Pereira, and I. F. M. Menezes, “Polymesher: a general-purpose mesh generator for polygonal elements written in matlab,” *Structural and Multidisciplinary Optimization*, vol. 45, pp. 309–328, 2012.
- [23] B. Lévy and A. Filho, “Geogram: a library for geometric algorithms,” 2015.
- [24] A. Vaxman *et al.*, “libhedra: geometric processing and optimization of polygonal meshes,” 2017.

- [25] M. Livesu, *Cinolib: A Generic Programming Header Only C++ Library for Processing Polygonal and Polyhedral Meshes*, pp. 64–76. Berlin, Heidelberg: Springer Berlin Heidelberg, 2019.
- [26] P. Antonietti and E. Manuzzi, “Refinement of polygonal grids using convolutional neural networks with applications to polygonal discontinuous galerkin and virtual element methods,” *Journal of Computational Physics*, vol. 452, p. 110900, 2022.
- [27] P. F. Antonietti, N. Farenga, E. Manuzzi, G. Martinelli, and L. Saverio, “Agglomeration of polygonal grids using graph neural networks with applications to multigrid solvers,” *Computers & Mathematics with Applications*, vol. 154, pp. 45–57, 2024.
- [28] P. Houston, C. Schwab, and E. Süli, “Discontinuous hp -finite element methods for advection-diffusion-reaction problems,” *SIAM Journal on Numerical Analysis*, vol. 39, no. 6, pp. 2133–2163, 2002.
- [29] A. Cangiani, Z. Dong, and E. H. Georgoulis, “ hp -version space-time discontinuous Galerkin methods for parabolic problems on prismatic meshes,” *SIAM Journal on Scientific Computing*, vol. 39, no. 4, pp. A1251–A1279, 2017.
- [30] M. Botti, D. A. D. Pietro, and P. Sochala, “A hybrid high-order discretization method for nonlinear poroelasticity,” *Computational Methods in Applied Mathematics*, vol. 20, no. 2, pp. 227–249, 2020.
- [31] L. Botti, M. Botti, and D. A. Di Pietro, “An abstract analysis framework for monolithic discretisations of poroelasticity with application to hybrid high-order methods,” *Computers & Mathematics with Applications*, vol. 91, pp. 150–175, 2021. Robust and Reliable Finite Element Methods in Poromechanics.
- [32] P. F. Antonietti and I. Mazzieri, “High-order discontinuous Galerkin methods for the elastodynamics equation on polygonal and polyhedral meshes,” *Computer Methods in Applied Mechanics and Engineering*, vol. 342, pp. 414–437, 2018.
- [33] M. Corti, P. F. Antonietti, L. Dede’, and A. M. Quarteroni, “Numerical modeling of the brain poromechanics by high-order discontinuous galerkin methods,” *Mathematical Models and Methods in Applied Sciences*, vol. 33, no. 8, pp. 1577–1609, 2023.
- [34] M. Corti, F. Bonizzoni, L. Dede’, A. M. Quarteroni, and P. F. Antonietti, “Discontinuous Galerkin methods for Fisher–Kolmogorov equation with application to α -synuclein spreading in Parkinson’s disease,” *Computer Methods in Applied Mechanics and Engineering*, vol. 417, p. 116450, 2023.
- [35] I. Fumagalli, M. Corti, N. Parolini, and P. F. Antonietti, “Polytopal discontinuous galerkin discretization of brain multiphysics flow dynamics,” *Journal of Computational Physics*, vol. 513, p. 113115, 2024.
- [36] P. F. Antonietti, M. Botti, I. Mazzieri, and S. Nati Poltri, “A high-order discontinuous Galerkin method for the poro-elasto-acoustic problem on polygonal and polyhedral grids,” *SIAM Journal of Scientific Computing*, vol. 44, no. 1, pp. B1–B28, 2021.
- [37] P. F. Antonietti, S. Bonetti, and M. Botti, “Discontinuous galerkin approximation of the fully coupled thermo-poroelastic problem,” *SIAM Journal on Scientific Computing*, vol. 45, no. 2, pp. A621–A645, 2023.
- [38] S. Bonetti, M. Botti, I. Mazzieri, and P. F. Antonietti, “Numerical modelling of wave propagation phenomena in thermo-poroelastic media via discontinuous galerkin methods,” *Journal of Computational Physics*, vol. 489, p. 112275, 2023.

- [39] M. W. Scroggs, I. A. Baratta, C. N. Richardson, and G. N. Wells, “Basix: a runtime finite element basis evaluation library,” *Journal of Open Source Software*, vol. 7, no. 73, p. 3982, 2022.
- [40] M. Alnæs, J. Blechta, J. Hake, A. Johansson, B. Kehlet, A. Logg, C. Richardson, J. Ring, M. E. Rognes, and G. N. Wells, “The fenics project version 1.5,” *Archive of numerical software*, vol. 3, no. 100, pp. 9–23, 2015.
- [41] P. C. Africa, “lifex: A flexible, high performance library for the numerical solution of complex finite element problems,” *SoftwareX*, vol. 20, p. 101252, 2022.
- [42] P. C. Africa, R. Piersanti, F. Regazzoni, M. Bucelli, M. Salvador, M. Fedele, S. Pagani, L. Dede’, and A. Quarteroni, “lifex-ep: a robust and efficient software for cardiac electrophysiology simulations,” *BMC bioinformatics*, vol. 24, no. 1, p. 389, 2023.
- [43] P. C. Africa, I. Fumagalli, M. Bucelli, A. Zingaro, M. Fedele, L. Dede’, and A. Quarteroni, “lifex-cfd: An open-source computational fluid dynamics solver for cardiovascular applications,” *Computer Physics Communications*, vol. 296, p. 109039, 2024.
- [44] R. Anderson, J. Andrej, A. Barker, J. Bramwell, J. S. Camier, J. Cerveny, V. Dobrev, Y. Dudouit, A. Fisher, T. Kolev, W. Pazner, M. Stowell, V. Tomov, I. Akkerman, J. Dahm, D. Medina, and S. Zampini, “Mfem: A modular finite element methods library,” *Computers & Mathematics with Applications*, vol. 81, pp. 42–74, 2021.
- [45] C. J. Permann, D. R. Gaston, D. Andrš, R. W. Carlsen, F. Kong, A. D. Lindsay, J. M. Miller, J. W. Peterson, A. E. Slaughter, R. H. Stogner, and R. C. Martineau, “Moose: Enabling massively parallel multiphysics simulation,” *SoftwareX*, vol. 11, p. 100430, 2020.
- [46] F. Dassi, “Vem++, a c++ library to handle and play with the Virtual Element Method,” 2023.
- [47] J. Droniou, “HArDCore.” <https://github.com/jdroniou/HArDCore>, 2019.
- [48] K. A. Lie, *An Introduction to Reservoir Simulation Using MATLAB/GNU Octave: User Guide for the MATLAB Reservoir Simulation Toolbox (MRST)*. Cambridge: Cambridge University Press, 2019.
- [49] P. F. Antonietti, C. Facciolà, P. Houston, I. Mazzieri, G. Pennesi, and M. Verani, *High-order Discontinuous Galerkin Methods on Polyhedral Grids for Geophysical Applications: Seismic Wave Propagation and Fractured Reservoir Simulations*, pp. 159–225. Cham: Springer International Publishing, 2021.
- [50] 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*, vol. 39, no. 5, pp. 1749–1779, 2002.
- [51] A. D. Pouliakis, *Handbook of Formulas and Tables for Signal Processing*. Berlin, Heidelberg: Springer, 1 ed., 1999.
- [52] P. F. Antonietti, P. Houston, and G. Pennesi, “Fast numerical integration on polytopic meshes with applications to discontinuous Galerkin finite element methods,” *J. Sci. Comput.*, vol. 77, no. 3, pp. 1339–1370, 2018.
- [53] L. Zhang, T. Cui, and H. Liu, “A set of symmetric quadrature rules on triangles and tetrahedra,” *Journal of Computational Mathematics*, vol. 27, no. 1, pp. 89–96, 2009.
- [54] S.-A. Papanicolopoulos, “Computation of moderate-degree fully-symmetric cubature rules on the triangle using symmetric polynomials and algebraic solving,” *Computers & Mathematics with Applications*, vol. 69, no. 7, pp. 650–666, 2015.

- [55] S. E. Mousavi, H. Xiao, and N. Sukumar, “Generalized Gaussian quadrature rules on arbitrary polygons,” *International Journal for Numerical Methods in Engineering*, vol. 82, no. 1, pp. 99–113, 2010.
- [56] P. F. Antonietti, A. Cangiani, J. Collis, Z. Dong, E. H. Georgoulis, S. Giani, and P. Houston, *Review of Discontinuous Galerkin Finite Element Methods for Partial Differential Equations on Complicated Domains*, pp. 281–310. Cham: Springer International Publishing, 2016.
- [57] P. J. LaMontagne, T. L. Benzinger, J. C. Morris, S. Keefe, R. Hornbeck, C. Xiong, E. Grant, J. Hassenstab, K. Moulder, A. G. Vlassenko, M. E. Raichle, C. Cruchaga, and D. Marcus, “OASIS-3: Longitudinal neuroimaging, clinical, and cognitive dataset for normal aging and Alzheimer disease,” *medRxiv*, 2019.
- [58] J. Crank and P. Nicolson, “A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type,” *Advances in Computational Mathematics*, vol. 6, no. 1, pp. 207–226, 1996.
- [59] J. C. Butcher, *Numerical Methods for Ordinary Differential Equations 3rd Edition*. Wiley, 2016.
- [60] A. Quarteroni, *Numerical Models for Differential Problems*. Milano: Springer, 3 ed., 2017.
- [61] P. F. Antonietti and I. Mazzieri, “High-order discontinuous Galerkin methods for the elastodynamics equation on polygonal and polyhedral meshes,” *Comput. Methods Appl. Mech. Engrg.*, vol. 342, pp. 414–437, 2018.
- [62] M. A. Biot, “Theory of Propagation of Elastic Waves in a Fluid-Saturated Porous Solid. I. Low-Frequency Range,” *J. Acoust. Soc. Am.*, vol. 28, pp. 168–178, 1956.
- [63] G. Chiavassa and B. Lombard, “Wave propagation across acoustic/Biot’s media: A finite-difference method,” *Communications in Computational Physics*, vol. 13, no. 4, pp. 985–1012, 2013.