MOX-Report No. 06/2024

lymph: discontinuous poLYtopal methods for Multi-PHysics differential problems

Antonietti, P.F., Bonetti, S., Botti, M., Corti, M., Fumagalli, I., Mazzieri, I.

lymph: discontinuous poLYtopal methods for Multi-PHysics differential problems

Paola F. Antonietti[1], Stefano Bonetti[1], Michele Botti[1], Mattia Corti[1], Ivan Fumagalli[1], and Ilario Mazzieri[1]

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

January 24, 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, and the elastodynamics system. 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.

Keywords: polygonal mesh, discontinuous Galerkin method, high-order methods, multi-physics problems, numerical simulations, open-source software

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, polytopal meshes have become increasingly popular as a solution for these numerical challenges due to their flexibility in representing intricate geometries, 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]). An important aspect of the development of polygonal methods is mesh generation. In the following, we report

some works in the processing of 2D and 3D polygonal/polytopal meshes [21, 22, 23, 24, 25]. In addition, we highlight that there are some recent works in which Machine Learning techniques are exploited for mesh refinement and coarsening [26, 27].

In this work, we focus on the high-order discontinuous Galerkin finite element method on polytopal grids (PolyDG [1, 2, 3, 5]). The use of the PolyDG method offers numerous benefits when dealing with coupled problems: *(i)* accurate 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-order accuracy. Related to the geometrical flexibility, another attractive aspect concerns the treatment of transmission conditions; indeed, these conditions are usually localized on sub-regions of the computational domain and must be accurately represented without compromising efficiency. The PolyDG method possesses some distinguishing features, that make it very appealing for multi-physics differential problems. First, the PolyDG scheme can be easily combined with agglomeration strategies for adaptivity which usually leads to many small faces per element. Indeed, the dimension of local approximation space does not depend on the number of faces. Concerning refinement procedures, its hierarchical basis structure can 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 of PolyDG is that it is dimension-independent; this is particularly useful when moving from 2D problems to 3D ones. Examples of PolyDG schemes can be found in [2, 1] for elliptic problems, in [28] for advection-diffusion-reaction problems, in [29] for parabolic problems, in [30, 31, 32] for poroelasticity, in [33] for Stokes problem, and in [34, 35] for wave propagation problems. In the following works, you can find the PolyDG discretization of the coupled problem we mentioned at the beginning of the introduction: brain modeling [36, 37, 38], seismicity [39, 40], and computational geosciences [41, 42].

This paper aims to introduce `lymph` (discontinuous poLYthopal methods for Multi-PHysics), an open-source MATLAB library for the PolyDG approximation of multi-physics problems. There is already some software on the market for the numerical approximation of multi-physics problems (e.g. `Basix` [43], `FEniCS` [44], `life`$^\text{x}$ [45, 46, 47], `MFEM` [48], `MOOSE` [49]) and very few for the solution of problems on polytopal meshes (e.g. `MRST` [50]). However, `lymph` 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 polytopal meshes and, more specifically, by discretizing the problem via PolyDG schemes. Moreover, the library is very flexible in terms of coupling different existing physics and implementation of new ones. The semidiscrete formulation is then suitably coupled with 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, Section 4 we describe the library in terms of installation, input/output data, and code structure. 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 technical results for the design of PolyDG schemes.

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, 51]).

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: $h_\kappa \lesssim d\,|S_\kappa^F|\,|F|^{-1}$, with h_κ denoting the diameter of the element κ 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 that might be different at different occurrences but does not depend on the discretization parameters (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 the mesh $\mathcal{T}_h$. Given an element-wise constant polynomial degree $\ell : \mathcal{T}_h \to \mathbb{N}_{>0}$ which determines the order of the approximation, the discrete spaces are defined such as

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

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

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

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

A.2 *For each $\mathcal{T}_h \in \{\mathcal{T}_h\}_{h \to 0}$ 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 $\kappa^\pm$ as in [52]:

$$[\![a]\!] = a^+ \mathbf{n}^+ + a^- \mathbf{n}^-, \quad [\![\mathbf{a}]\!] = \mathbf{a}^+ \otimes \mathbf{n}^+ + \mathbf{a}^- \otimes \mathbf{n}^-, \quad [\![\mathbf{a}]\!]_n = \mathbf{a}^+ \cdot \mathbf{n}^+ + \mathbf{a}^- \cdot \mathbf{n}^-,$$

$$\{\!\{a\}\!\} = \frac{a^+ + a^-}{2}, \qquad \{\!\{\mathbf{a}\}\!\} = \frac{\mathbf{a}^+ + \mathbf{a}^-}{2}, \qquad \{\!\{\mathbf{A}\}\!\} = \frac{\mathbf{A}^+ + \mathbf{A}^-}{2},$$

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 $\kappa^\pm$ 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).

3 The lymph library

This section introduces the main structure and the components of `lymph` library. We start by giving a high-level overview of the library, describing its main functionalities as well as its folder structure. Next, we present a flowchart illustrating the overall workflow of the code, as well as code snippets and possible configuration and customization of `lymph` for specific needs. We postpone a more procedural user guide in Section 5, going along the numerical approximation and implementation of a specific differential problem.

Overview

`lymph` is designed to solve either single or multi-physics differential problems employing the high-order PolyDG method, as explained in Section 2. The library is organized into different folders: `Core` which contains the main routines such as the mesh generation, the polynomial space construction, and quadrature formulas; `Physics`, which contains specific routines for particular problems, organized in subfolders. In the current release, `Physics` contains solvers for Poisson problem (`Laplacian`), time-dependent diffusion-reaction equations (`Heat`), and elastodynamics system (`Elastodynamics`).

Installation

`lymph` has been developed in Matlab version R2022b, but its functionalities have been successfully tested in all versions from R2020b to R2023b, which include 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 repository contains also `Polymesher v1.1` [22], on which `lymph` relies for the generation of 2D polygonal grids of arbitrary shape. The addition of `lymph`'s paths and subpaths to the working environment is directly accounted for by the main file, as explained in the following paragraphs.

`Core` folder

The `Core` folder contains special functions that are the building blocks of the PolyDG discretization. In particular, we have split these routines into the following directories:

- `MeshGeneration`: routines for the polygonal mesh generation, cf. Section 4.1,

- `FEspace`: finite element definition, computation of the basis functions and their derivatives, cf. Section 4.2,

- `Quadrature`: quadrature formulas used to compute the elemental and boundary integrals, cf. Section 4.3,

- `PostProcessing`: functions to export the solution and mesh in Matlab and Paraview formats,

- `Utilities`: additional routines.

`Physics` folders

Inside each physics folder, one can find the following directories:

- `MainFunctions`: main routine containing (i) the mesh reading, (ii) the finite element space construction, (iii) the assembly of the linear system (ode system for time-dependent problems), (iv) algebraic solvers, (v) post-processing of the solutions, and (vi) error computation (if needed),

- `InputData`: input data files,

- `Assembly`: routines for the assembly of the linear system and of the right-hand side,

- `TimeIntegration` (time-dependent physics only): time-advancing schemes;

- `PostProcessing`: routines integrating `Core/PostProcessing` for a specific physics,

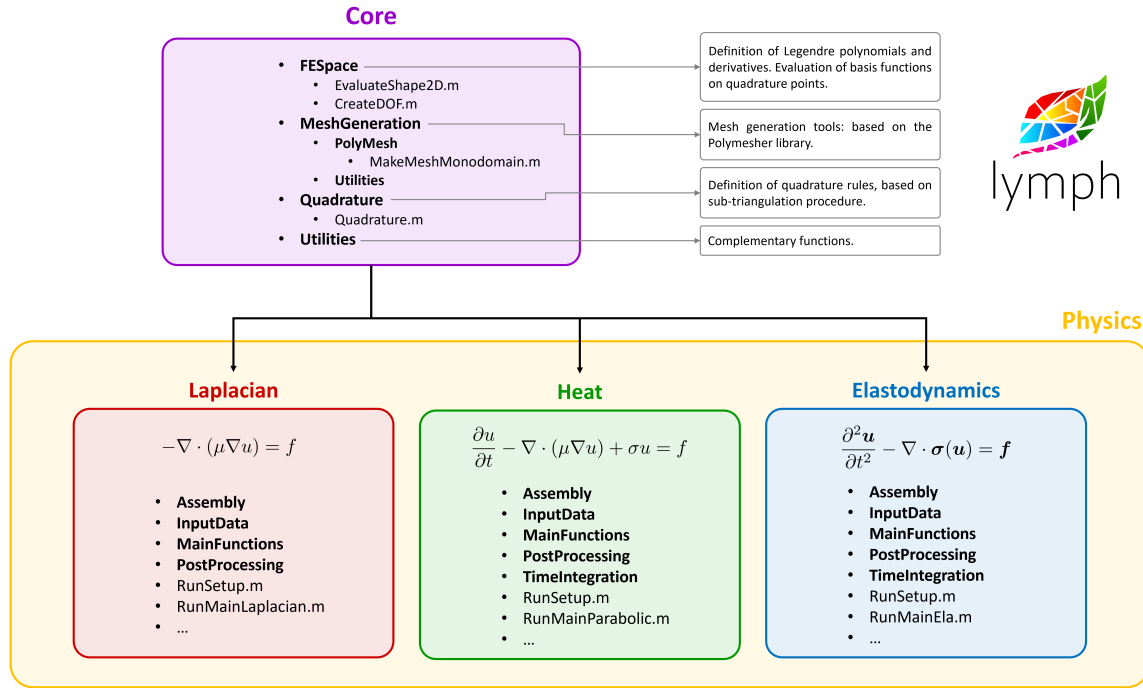- `Error`: routines for the errors computation,

and functions:

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

- `ImportLymphPath.m:` to set up the path of the linked folders,

- `RunSetup.m:` setup of the simulation, selection of output formats and output folders, visualization of the solution, and the polygonal mesh,

- `RunMainPHYS.m:` script used to run the specific simulation,

- `RunhConvergencePHYS:` script used for verification purposes (h-convergence),

- `RunpConvergencePHYS:` script used for verification purposes (p-convergence).

Here, the suffix `PHYS` stands for the given physics name, e.g. `Laplacian, Heat, Elastodynamics`.

Dependencies and External Libraries

To solve specific problems each `Physics` folder, containing functions to discretize a differential problem through the PolyDG method, must be linked to the `lymph` library as it will be explained in the Section 5. The `lymph` library makes use of the `Polymesher` software [22] that is provided within the `Core/MeshGeneration/PolyMesh` directory or alternatively can be downloaded from `http://paulino.princeton.edu/software.html`.

Additional Folder Structure

As described above, the main folder structure of the code is divided into two parts: the `Core` and the `Physics` folders. Data files for specific runs must be included in the `Physics/InputData` folder. Additionally, mesh files (in `.mat` format) can be stored in the `Physics/InputMesh` directory for further runs. Output results are saved in directories specified in the `InputData` files. A flowchart illustrating the code structure and workflow is given in Fig. 1.

Configuration, Customization and Documentation

The user can configure or customize the code for their specific needs by using the `Physics/RunSetup.m` script. The use of this script, the full documentation of the code, as well as simple tutorials to get practice with the `lymph` library (see also Section 5), are available at `https://lymph.bitbucket.io/`. For each physics under examination, step-by-step practical examples are given in the `Tutorial` section of the user guide. Here, for the sake of presentation, we report a summary.

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.

4.1 Mesh Generation

We use the Polymesher software [22], embedded in the `lymph` library. 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 $|\mathcal{T}_h|$ is the number of elements of the partition `femregion.nel` and $\dim\{P^\ell(k)\} = \frac{1}{2}(\ell+1)(\ell+2)$ `femregion.nbases`.

In the `lymph` library, the basis functions for each physical element κ are constructed starting from the Legendre polynomials of order ℓ [53] in one dimension $\mathcal{L}_i = \mathcal{L}_i(x)$ with $i = 0, ..., \ell$. Then, by using a tensor product of basis functions in the two directions, we obtain that:

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

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 consists of the computation of volume and boundary integrals. As a general example, we consider the following integrals and explain how they are computed in `lymph`. We start by considering the general volume terms:

$$\texttt{Mprj\_loc}(i,j) = (\varphi_j, \varphi_i)_\kappa \quad \text{and} \quad \texttt{A\_loc}(i,j) = (\mu\nabla\varphi_j, \nabla\varphi_i)_\kappa, \quad \text{for } i, j = 1, ..., \frac{1}{2}(\ell+1)(\ell+2),$$

where μ is a given 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 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` $= (2\ell+1)^2$ points `ref_qNodes_2D` and weights `w_2D` which is exact for the quantities of interest. We point out that there exist more efficient quadrature rules for the numerical approximation of integrals of polynomial functions

over general polygonal/polyhedral elements that do not require explicit construction of a sub-tessellation into triangular/tetrahedral elements, e.g. the quadrature free-method of [54, 55]. Here the main idea is based on successive application of Stokes' theorem; thereby, the underlying integral may be evaluated using only the values of the integrand at the vertices of the polytopic domain, and hence leads to an exact quadrature rule whose quadrature points are the vertices of the polytopal element. For the sake of simplicity in this first release of the `lymph` library we rely on the sub-tessellation quadrature approach, soon we plan to include in `lymph` the possibility of choosing between different quadrature approaches, including also the quadrature-free approach.

```matlab
%% Laplacian/Assembly/MatrixLaplacian.m
%% Quadrature values
[ref_qNodes_1D, w_1D, ref_qNodes_2D, w_2D] = Quadrature(femregion.nqn);
...
for ie = 1:femregion.nel %Loop over the elements
    ...
    % Creation of the subtriangulation of the element
    ...
    for iTria = 1:size(Tria,1) % Loop over the subtriangulation
        % Construction of Jacobian and quadrature nodes
        [BJ, qNodes_2D] = GetJacobianPhysicalPoints(.,.);   xq  =
            qNodes_2D(:,1); yq  = qNodes_2D(:,2);
        dx = det(BJ) * w_2D; % Scaled weights
        mu = Data.mu{1}(xq,yq);% Evaluation of physical parameters
        ...
        %% Matrix assembling
        Mprj_loc = Mprj_loc + (dx.*phiq)'*phiq;
        A_loc = A_loc + (dx .* (mu .* gradqx))' * gradqx + (dx .* (mu .*
            gradqy))' * gradqy;
    end
...
```

Next, denoting by $(\cdot, \cdot)_e$ the L^2 inner product over $e \in \mathcal{F}_h^I$, 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, for $i, j = 1, ..., \frac{1}{2}(\ell+1)(\ell+2)$ we get:

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

- $(\alpha_e[\![\varphi_j]\!], [\![\varphi_i]\!])_e = (\alpha_e(\varphi_j^+ - \varphi_j^-), \varphi_i^+)_e + (\alpha_e(\varphi_j^- - \varphi_j^+), \varphi_i^+)_e.$

When the current element is κ^+ only the following integrals are computed:

$$\mathtt{IA\_loc}(i,j) = \frac{1}{2}(\mu^+ \nabla \varphi_j^+ \cdot \boldsymbol{n}^+, \varphi_i^+)_e \quad \text{and} \quad \mathtt{IAN\_loc}(i,j) = -(\frac{1}{2}\mu^- \nabla \varphi_j^- \cdot \boldsymbol{n}^+, \varphi_i^+)_e,$$

and

$$\mathtt{SA\_loc}(i,j) = \alpha_e(\varphi_j^+, \varphi_i^+)_e \quad \text{and} \quad \mathtt{SAN\_loc}(i,j) = -\alpha_e(\varphi_j^-, \varphi_i^+)_e,$$

for $i, j = 1, ..., \frac{1}{2}(\ell+1)(\ell+2)$. Their computation is achieved in the function `Laplacian/Assembly/MatrixLaplacian.m`

```matlab
%% Laplacian/Assembly/MatrixLaplacian.m
...
for ie = 1:femregion.nel %loop over the elements
    ...
```

```matlab
     [penalty_geom] = PenaltyCoefficient(.,.,.);
    for iedg = 1 : neighbor.nedges(ie)      % Loop over faces
        ...
        % Construction of quadrature nodes on the face
        [qNodes_1D] = GetPhysicalPointsFaces(.,.,.);
        xq = qNodes_1D(:,1); yq = qNodes_1D(:,2);
        ds = meshsize(iedg) * w_1D;  % Scaled weights
        ...
        % Construction of the basis functions
        [phiedgeq, gradedgeqx, gradedgeqy] = Evalshape2D(.,.,);
        %Element itself
        IA_loc = IA_loc + 0.5 * (ds .* mu .* ( nx * gradedgeqx + ny *
            gradedgeqy))' * phiedgeq;
        SA_loc = SA_loc + (ds .* (mu * penalty_geom(iedg)) .* phiedgeq)'
             * phiedgeq;
        % Construction of the basis functions for the neighbor
        phiedgeqneigh = Evalshape2D(.,.,.);
        % Neighboring element
        IAN_loc(:,:,iedg) = IAN_loc(:,:,iedg) - 0.5 * (ds .* (mu .* ( nx
             * gradedgeqx +  ny * gradedgeqy)))' * phiedgeqneigh;
        SAN_loc(:,:,iedg) = SAN_loc(:,:,iedg) - (ds .* (mu *
            penalty_geom(iedg)) .* phiedgeq)' * phiedgeqneigh;
    end
...
```

5 Examples

In the following, we show how to solve differential problems, with `lymph` .

The Poisson problem

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

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

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.

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

where

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

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

$$\alpha_e(\boldsymbol{x}) = \begin{cases} C_\alpha \max_{\kappa \in \{\kappa^+, \kappa^-\}} \left(\mu_\kappa \frac{\ell_\kappa^2}{h_\kappa} \right), & \boldsymbol{x} \in e, e \in \mathcal{F}_I, e \subset \partial\kappa^+ \cap \partial\kappa^-, \\ C_\alpha \mu_\kappa \frac{\ell_\kappa^2}{h_\kappa}, & \boldsymbol{x} \in e, e \in \mathcal{F}_B, e \subset \partial\kappa^+ \cap \partial\Omega, \end{cases}$$
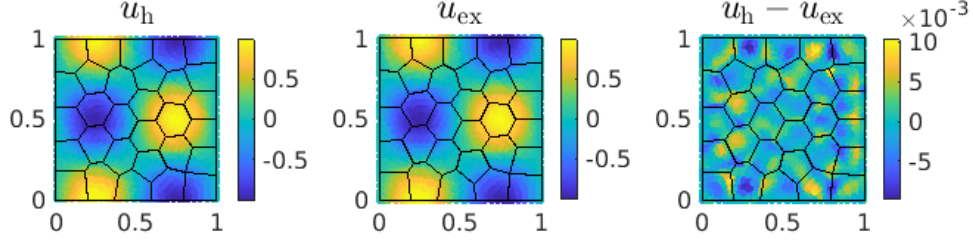
8

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

being $C_\alpha > 0$ 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.

$$A_{dG} \boldsymbol{U}_h = \boldsymbol{F},$$

with $A_{dG} \in \mathbb{R}^{N_h} \times \mathbb{R}^{N_h}$ defined for any $i, j = 1, ..., N_h$ as

$$(A_{dG})_{ij} = \sum_{\kappa \in \mathcal{T}_h} (\mu \nabla \varphi_j, \nabla \varphi_i)_\kappa - \sum_{e \in \mathcal{F}_h} \left((\{\!\{\mu \nabla \varphi_j\}\!\}, [\![\varphi_i]\!])_e + ([\![\varphi_j]\!], \{\!\{\mu \nabla \varphi_i\}\!\})_e - (\alpha_e [\![\varphi_j]\!], [\![\varphi_i]\!])_e \right)$$

while $\boldsymbol{F} \in \mathbb{R}^{N_h}$ is given by $\boldsymbol{F}_i = \sum_{\kappa \in \mathcal{T}_h} (f, \varphi_i)_\kappa - \sum_{e \in \mathcal{F}_B} \left((g, \mu \nabla \varphi_i)_e - (\alpha_e g, \varphi_i)_e \right)$ for all $i = 1, ..., N_h$. The entries of the matrix A_{dG} as well as the right-hand side $\boldsymbol{F}$ in (5) are computed as explained in Section 4. For the sake of completeness, we introduce the dG-norm as $\|u\|_{dG}^2 = \|\sqrt{\mu} \nabla u\|_{L^2(\Omega)}^2 + \|\sqrt{\alpha_e} [\![u]\!]\|_{L^2(\mathcal{F})}^2$ 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:

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

with $s_\kappa = min(\ell_\kappa + 1, m_\kappa)$ for all $\kappa \in \mathcal{T}_h$ and h_κ denoting the element diameter.

Verification test

We consider problem (5) 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 (5). 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 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
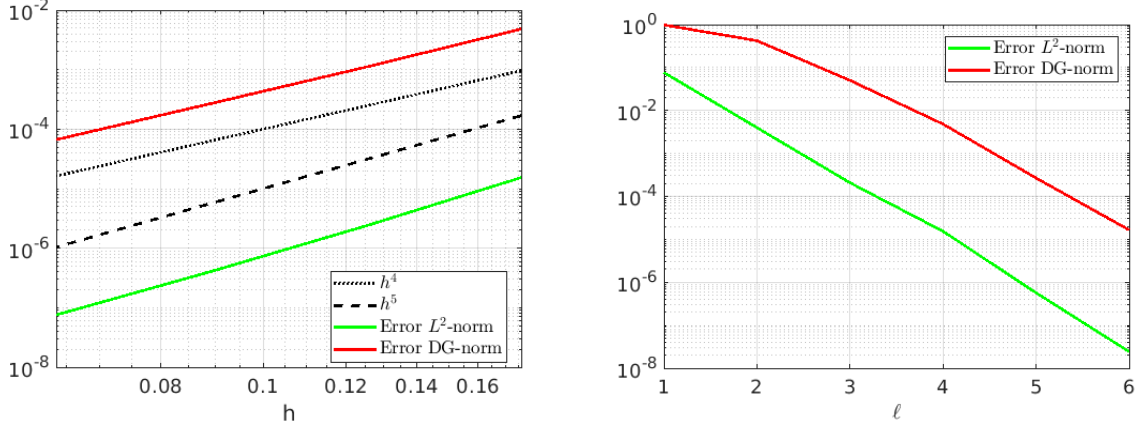
Figure 3: Left: computed errors $\|u - u_h\|_{dG}$ and $\|u - u_h\|_{L^2(\Omega)}$ as a function of the mesh size h by fixing the polynomial degree $\ell = 4$. Right: computed errors $\|u - u_h\|_{dG}$ and $\|u - u_h\|_{L^2(\Omega)}$ as a function of the polynomial degree $\ell = 4$ by fixing the number of mesh element $N_{el} = 100$.

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 (5). On the right, the exponential convergence with respect to the polynomial degree ℓ is also shown, cf. (5), by fixing $N_{el} = 100$.

The heat equation

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

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

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

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

with initial conditions $u_h = u_{0h}$, being $u_{0,h}$ the L^2-projection of the initial data on V_h^ℓ. In (5) 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:

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

To integrate system (5) in time we apply the $\theta-$method scheme. The numerical simulation is performed by using the Crank-Nicolson scheme ($\theta = 1/2$).
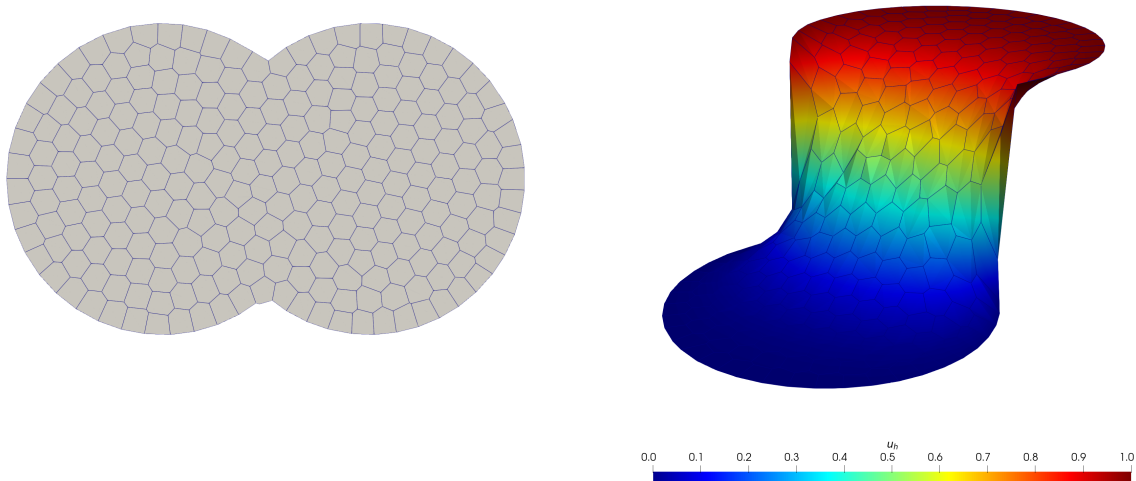
10

Figure 4: Example heat equation. Polygonal mesh of the two circles (left) and computed PolyDG solution at time $t = 1.0$.

Test case with discontinuous boundary conditions

As an application of the presented PolyDG method, we solve with `lymph` the heat equation problem presented in [57]. It considers the parabolic problem with $\mu = 0.1$, $\sigma = 0$, and homogeneous forcing term $f = 0$. 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 4 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 $u(\boldsymbol{x}) = 0$ for $x \leq 0$ and $u(\boldsymbol{x}) = 1$ for $x > 0$, where $\boldsymbol{x} = (x, y)$. As shown in the test case reference [57], 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 $\Delta t = 0.025$, and a final time $T = 1$. In Figure 4 we report the snapshot of the solution at the final time $T = 1$. The solution is coherent to what is expected from the literature [57].

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 $\boldsymbol{n}$. The boundary is assumed to be composed of two disjoint portions Γ_D and Γ_N, where Dirichlet ($\boldsymbol{u} = \boldsymbol{0}$), and Neumann ($\boldsymbol{\sigma}(\boldsymbol{u})\boldsymbol{n} = \boldsymbol{0}$), conditions are imposed, respectively, cf. [58]. Given sufficiently regular external loads $\boldsymbol{f}$ and $\boldsymbol{g}$ and initial data $\boldsymbol{u}_0$ and $\boldsymbol{v}_0$, cf. [59], the equations of (linear) elastodynamics in $\Omega \times (0, T]$ are given by

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

with initial conditions $(\boldsymbol{u}, \frac{\partial \boldsymbol{u}}{\partial t})(0) = (\boldsymbol{u}_0, \boldsymbol{v}_0)$, in Ω. We denote by $\boldsymbol{u} : \Omega \times [0, T] \to \mathbb{R}^2$ the displacement vector and by $\boldsymbol{\sigma} : \Omega \times [0, T] \to \mathbb{S}$ the stress tensor where $\mathbb{S}$ is the space of symmetric, 2×2, real-valued tensor fields. We assume the generalized Hooke's law $\boldsymbol{\sigma}(\boldsymbol{u}) = 2\mu\boldsymbol{\epsilon}(\boldsymbol{u}) + \lambda tr(\boldsymbol{\epsilon}(\boldsymbol{u}))I$, being $\boldsymbol{\epsilon}(\boldsymbol{u})$ the symmetric
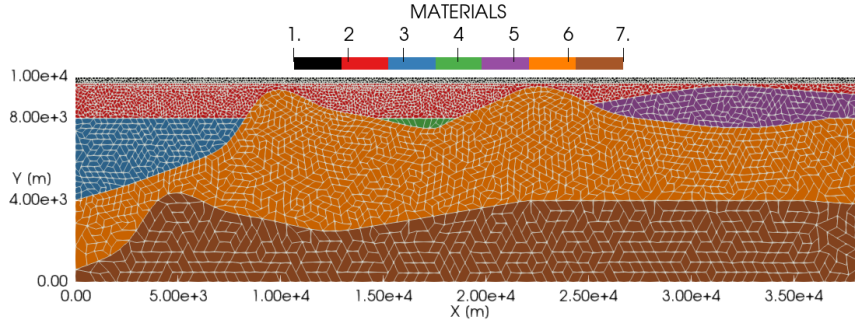
Figure 5: Example elastodynamics system. Unstructured polygonal grid with mesh spacing of about $h \approx 160$ m for material 1 to $h \approx 1500$ m for material 7; cf. Table 1. Mesh file is available in `Elastodynamics/InputMeshPhysics/MeshEmilia.mat`.

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 1: Example elastodynamics system. Material properties used for the computational domain in Figure 5, cf. also `Elastodynamics/InputData/Elastic/DataTestPhysicsEla.m`

gradient of $\boldsymbol{u}$, I the identity tensor, $tr(\cdot)$ the trace operator, and $\lambda, \mu \in L^\infty(\Omega)$ are the Lamé's parameters. The compressional (P) and shear (S) wave velocities of the medium are obtained through the relations $c_P = \sqrt{(\lambda + 2\mu)/\rho}$ and $c_S = \sqrt{\mu/\rho}$, respectively. By following [59], we obtain the PolyDG formulation: for any time $t \in (0, T]$ find $\boldsymbol{u}_h = \boldsymbol{u}_h(t) \in \boldsymbol{V}_h^\ell$ such that

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

with initial conditions $(\boldsymbol{u}_h, \frac{\partial \boldsymbol{u}_h}{\partial t}) = (\boldsymbol{u}_{0h}, \boldsymbol{v}_{0h})$, being $\boldsymbol{u}_{0,h}$ and $\boldsymbol{v}_{0,h}$ the L^2-projection of the initial data on $\boldsymbol{V}_h^\ell$. In (5) the bilinear forms $a_{dG}^e(\cdot, \cdot)$ is defined as

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

with η_e as in [59, eq. (9)]. Now, by introducing a set of basis functions $\{\boldsymbol{\varphi}_j^1, \boldsymbol{\varphi}_j^2\}_{j=1}^{N_h}$ for $\boldsymbol{V}_h^\ell$ we can easily get the following system of second order differential equations:

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

To integrate system (5) in time we apply the Newmark β-scheme, with $\beta = \frac{1}{4}$ and $\gamma = \frac{1}{2}$.

Wave propagation in a layered media

As an application of the presented PolyDG method, we solve with `lymph` the wave propagation problem presented in [51, 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 5.

We consider homogeneous Neumann conditions on the top of the model ($\boldsymbol{\sigma}\boldsymbol{n} = \boldsymbol{0}$) whereas homogeneous Dirichlet conditions ($\boldsymbol{u} = \boldsymbol{0}$) 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 $\boldsymbol{f}(\boldsymbol{x}, t) = $
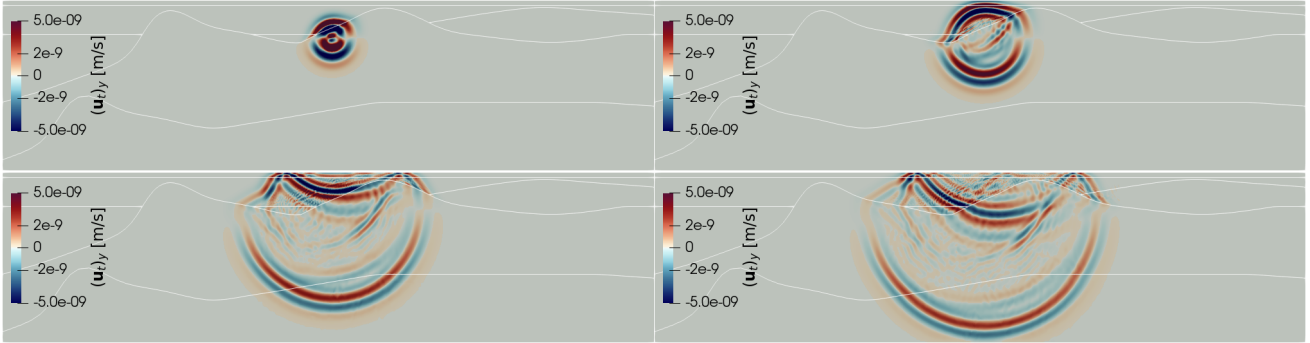
12

Figure 6: Example 5. Snapshots of the computed vertical velocity $(\boldsymbol{u}_t)_y$ 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 heterogeneities, high oscillations and perturbations of the wavefront can be observed, as well as the effect of the free surface on top.

dofs	Assembly of matrices	Assembly of RHS	Single time step	Single file saving
409.080	2056 s	13 s	≈ 300 s	250 s

Table 2: Example elastodynamics system. Computational time considering ≈ 400.000 degrees of freedom (`dofs`).

$-I \cdot \nabla \delta(\boldsymbol{x} - \boldsymbol{x}_s) S(t)$, being, $\delta(\boldsymbol{x} - \boldsymbol{x}_s)$ is the Dirac delta distribution centered in $\boldsymbol{x}_s = (19432, 7800)$ 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 1. The computational domain is discretized using an unstructured grid consisting of 4870 (agglomerated) polygonal elements, with a mesh size varying from $h \approx 160$ m for material 1 to $h \approx 1500$ m for material 7; cf. Table 1. We consider also a polynomial degree $\ell = 5$, $\Delta t = 0.01$ s, and a final time $T = 4$ s. In Figure 6 we report a set of snapshots of the computed vertical velocity field $(\boldsymbol{u}_t)_y$. 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 2 the computational cost of the presented test. The numerical simulation was performed using the Cerbero cluster (6 cores I7-3930K @3.20GHz + 20 cores Intel Xeon E5-2640v4 @ 2.40GHz, GPU NVidia GT520, 128GB RAM, O.S. CentOS 7 – Resource for sequential applications) at MOX, Department of Mathematics, Politecnico di Milano.

6   Conclusions

This paper presents `lymph`, a general-purpose Matlab library for the approximate solution of multi-physics differential problems. 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, the use of quadrature-free formulas, and the extension to three-dimensional settings.

7 Acknowledgements

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. Livesu, *Cinolib: A Generic Programming Header Only C++ Library for Processing Polygonal and Polyhedral Meshes*, pp. 64–76. Berlin, Heidelberg: Springer Berlin Heidelberg, 2019.

[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. Filbois, "Geogram: a library for geometric algorithms," 2015.

[24] M. Botsch, S. Steinberg, S. Bischoff, L. Kobbelt, and R. Aachen, "Openmesh - a generic and efficient polygon mesh data structure," 2002.

[25] A. Vaxman *et al.*, "libhedra: geometric processing and optimization of polygonal meshes," 2017.

[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. 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] P. F. Antonietti, C. Facciolà, A. Russo, and M. Verani, "Discontinuous Galerkin approximation of flows in fractured porous media on polytopic grids," *SIAM Journal on Scientific Computing*, vol. 41, no. 1, pp. A109–A138, 2019.

[31] 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.

[32] 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.

[33] P. F. Antonietti, L. Mascotto, M. Verani, and S. Zonca, "Stability analysis of polytopic Discontinuous Galerkin ppproximations of the Stokes problem with applications to Fluid-Structure Interaction Problems," *J. Sci. Comput.*, vol. 90, p. 23, Nov 2021.

[34] 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.

[35] J. de la Puente, M. Dumbser, M. Käser, and H. Igel, "Discontinuous Galerkin methods for wave propagation in poroelastic media," *Geophysics*, vol. 73, pp. T77–T97, 09 2008.

[36] 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.

[37] 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.

[38] I. Fumagalli, M. Corti, N. Parolini, and P. F. Antonietti, "Polytopal discontinuous galerkin discretization of brain multiphysics flow dynamics," 2023. arXiv preprint arXiv:2310.07651.

[39] P. F. Antonietti, M. Botti, and I. Mazzieri, "On mathematical and numerical modelling of multiphysics wave propagation with polytopal discontinuous galerkin methods: a review," *Vietnam Journal of Mathematics*, vol. 50, no. 4, pp. 997–1028, 2022.

[40] P. F. Antonietti, M. Botti, and I. Mazzieri, "A space-time discontinuous galerkin method for coupled poroelasticity-elasticity problems," 2023. arXiv preprint arXiv:2306.01140.

[41] 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.

[42] 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.

[43] 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.

[44] 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.

[45] P. C. Africa, "lifex: A flexible, high performance library for the numerical solution of complex finite element problems," *SoftwareX*, vol. 20, p. 101252, 2022.

[46] 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.

[47] 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.

[48] 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.

[49] 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.

[50] 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.

[51] 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.

[52] 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.

[53] A. D. Poularikas, *Handbook of Formulas and Tables for Signal Processing*. Berlin, Heidelberg: Springer, 1 ed., 1999.

[54] 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.

[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] A. Quarteroni, *Numerical Models for Differential Problems*. Milano: Springer, 3 ed., 2017.

[58] P. F. Antonietti, I. Mazzieri, A. Quarteroni, and F. Rapetti, "Non-conforming high order approximations of the elastodynamics equation," *Comput. Methods Appl. Mech. Engrg.*, vol. 209, pp. 212–238, 2012.

[59] 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.

MOX Technical Reports, last issues

Dipartimento di Matematica

Politecnico di Milano, Via Bonardi 9 - 20133 Milano (Italy)

05/2024 Conti, P.; Gobat, G.; Fresca, S.; Manzoni, A.; Frangi, A.

Reduced order modeling of parametrized systems through autoencoders and SINDy approach: continuation of periodic solutions

04/2024 Torzoni, M.; Tezzele, M.; Mariani, S.; Manzoni, A.; Willcox, K.E.

A digital twin framework for civil engineering structures

03/2024 Ciaramella, G.; Gander, M.J.; Vanzan, T.

A gentle introduction to interpolation on the Grassmann manifold

02/2024 Parolini, N.; Poiatti, A.; Vene', J.; Verani, M.

Structure-preserving neural networks in data-driven rheological models

Parolini, N.; Poiatti, A.; Vene', J.; Verani, M.

Structure-preserving neural networks in data-driven rheological models

01/2024 Criseo, M.; Fumagalli, I.; Quarteroni, A.; Marianeschi, S. M.; Vergara, C.

Computational haemodynamics for pulmonary valve replacement by means of a reduced Fluid-Structure Interaction model

109/2023 Clementi, L.; Arnone, E.; Santambrogio, M.D.; Franceschetti, S.; Panzica, F.; Sangalli, L.M.

Anatomically compliant modes of variations: new tools for brain connectivity

106/2023 Fontana, N.; Savaré, L.; Ieva, F.

Integrating state-sequence analysis to uncover dynamic drug-utilization patterns to profile heart failure patients

108/2023 Arnone, E.; Negri, L.; Panzica, F.; Sangalli, L.M.

Analyzing data in complicated 3D domains: smoothing, semiparametric regression and functional principal component analysis

105/2023 Cicci, L.; Fresca, S.; Guo, M.; Manzoni, A.; Zunino, P.

Uncertainty quantification for nonlinear solid mechanics using reduced order models with Gaussian process regression