Executive Summary of the Thesis

LYMPH3D: A new library to solve PDE problems with Discontinuous Galerkin methods on three-dimensional polytopic meshes

Laurea Magistrale in Mathematical Engineering - Ingegneria Matematica

Author: Nicoletta De Giosa

Advisor: Prof. Paola Antonietti

Co-advisor: Prof. Ilario Mazzieri

Academic year: 2022-2023

1. Introduction

Classical finite element methods typically only support computational grids composed of standard element shapes; tetrahedral, hexahedral, prismatic or pyramidal elements in three-dimensions. The use of these kind of elements necessitates the exploitation of very fine computational meshes when the geometry is complicated. In such situations, for a given mesh, a large number of elements is required to produce a mesh that adequately describes the underlying geometry. Thereby, the solution of the algebraic system of equations stemming from a finite element discretization of a generic differential problem on the underlying mesh, may be impractical due to the large number of degrees of freedom involved. To overcome this problem in the last decade numerical methods that support computational meshes composed of polytopic elements (polygonal or polyhedral) have gained a lot of relevance. One of the advantages of choosing polytopic element shapes over standard simplicial/tensor product elements is that the average number of elements needed to discretize complicated domains is substantially smaller and this allows to reduce the computational complexity of the given problem. This advantage becomes even more evident whenever the domain contains complex geometrical features. In this program framework, we implement and validate a new library called LYMPH3D, written in Fortran, based on discontinuous Galerkin (DG) finite element methods on polytopic grids (PolyDG) in three-dimensions, which is the natural extension of the classical discontinuous Galerkin methods on standard element shape grids to meshes composed of polytopic elements. We refer to [3, 4] and [1] for a comprehensive overview of the PolyDG method.

2. Discontinuous Galerkin Methods on Polytopic meshes

Let $\mathscr{T}_h$ be a subdivision of the computational domain $\Omega \subset \mathbb{R}^d, d = 3$, into disjoint open polyhedral elements E. For each element we denote by |E| its measure, h_E its diameter and we set $h = \max_{E\in\mathscr{T}_h} h_E$.
Next we introduce the concept of mesh interfaces, which consist of general polygons which we assume may be decomposed into sets of co-planar triangles. We refer to these $(d-1)$-dimensional simplices, whose union forms the in-

terfaces of $\mathscr{T}_h$ as the *faces* of the computational mesh. We denote the set of all the triangles by $\mathscr{F}_h$. We introduce a partition of the set $\mathscr{F}_h$ into two subsets $\mathscr{F}_h = \mathscr{F}_h^I \cup \mathscr{F}_h^B$, where $\mathscr{F}_h^I$ is the set of interior faces and $\mathscr{F}_h^B$ is the set of faces on the boundary of the domain $\partial\Omega$. Finally, given an element $E \in \mathscr{T}_h$, for any face $F \subset \partial E$, with $F \in \mathscr{F}_h$, we define $\mathbf{n}_F$ as the unit normal vector on F which points outwards from E.
For each element $E \in \mathscr{T}_h$ we associate a local polynomial degree $p_E \geq 1$. We collect the p_E in the vector $\mathbf{p} = \{p_E : E \in \mathscr{T}_h\}$. With this notation we introduce the following space.

$$V^{\mathbf{p}}(\mathscr{T}_h) = \{v_h \in L^2(\Omega) : v|_E \in \mathbb{P}_{p_E}(E) \quad \forall E \in \mathscr{T}_h\}$$

We recall that $\mathbb{P}_p(E)$ denotes the space of polynomials of total degree p on E.
Finally we introduce some assumptions on $\mathscr{T}_h$. We denote by S_E^F a d-dimensional simplex contained in E which shares with E a specific face $F \subset \partial E$, $F \in \mathscr{F}_h$. We need this notation to introduce the following definition.

Definition 2.1 (Polytopic Regular Mesh). *A mesh $\mathscr{T}_h$ is said to be polytopic-regular if:*

$$\forall E \in \mathscr{T}_h \quad \exists \{S_E^F\}_{F \subset \partial E} \text{ such that } \forall F \subset \partial E$$
$$h_E \lesssim d|S_E^F||F|^{-1},$$

where $\{S_E^F\}_{F\subset\partial E}$ is a set of non-overlapping d-dimensional simplices contained in E.

Definition 2.2 (Covering). *A covering $\mathscr{T}_\# = \{T_E\}$ related to the polytopic mesh $\mathscr{T}_h$ is a set of shape-regular d-dimensional simplices T_E, such that:*

$$\forall E \in \mathscr{T}_h \quad \exists T_E \in \mathscr{T}_\# \text{ such that } E \subsetneq T_E.$$

Assumption 2.2.1. *The mesh $\mathscr{T}_h$ satisfies the following properties:*

1. *$\mathscr{T}_h$ is polytopic-regular.*
2. *There exists a covering $\mathscr{T}_\#$ of $\mathscr{T}_h$ such that for each pair $E \in \mathscr{T}_h$ and $T_E \in \mathscr{T}_\#$, with $E \subset T_E$ it holds:*
   (a) $h_{T_E} \lesssim h_E$;
   (b) $\max_{E\in\mathscr{T}_h} |E'| \lesssim 1$ *where* $E' \in \mathscr{T}_h : E' \cap T_E \neq \emptyset, T_E \in \mathscr{T}_\#, E \subset T_E$.

We now introduce average and jump operators on a face. Let $F \in \mathscr{F}_h^I$ be an interior face shared by the elements $E^\pm$. We define $\mathbf{n}^\pm$ to be the unit normal vectors on F pointing exterior to $E^\pm$, respectively. Then, for sufficiently regular scalar-valued and vector-valued functions q,$\mathbf{v}$ respectively, we define the *average* $\{\cdot\}$ and *jump* $[\![\cdot]\!]$ operators on F as

$$\{q\} = \frac{1}{2}(q^+ + q^-), \quad [\![q]\!] = q^+\mathbf{n}^+ + q^-\mathbf{n}^-,$$
$$\{\mathbf{v}\} = \frac{1}{2}(\mathbf{v}^+ + \mathbf{v}^-), \quad [\![\mathbf{v}]\!] = \mathbf{v}^+ \cdot \mathbf{n}^+ + \mathbf{v}^- \cdot \mathbf{n}^-,$$

where the subscript $\pm$ on q denote the respective traces of the functions on F restricted to $E^\pm$, respectively.
On a boundary face $F \in \mathscr{F}_h^B$, we set analogously $\{q\} = q$, $[\![q]\!] = q\mathbf{n}$, $\{\mathbf{v}\} = \mathbf{v}$, $[\![\mathbf{v}]\!] = \mathbf{v}\cdot\mathbf{n}$, where $\mathbf{n}$ is the outward normal vector on $\partial\Omega$.

2.1. PolyDG Discrete Formulation

Given an open bounded Lipschitz domain Ω in $\mathbb{R}^d$, $d = 3$, with boundary $\partial\Omega$, we consider the following boundary-value problem subject to Dirichlet boundary conditions:
find u such that

$$\begin{cases} -\Delta u + cu = f & \text{in } \Omega, \\ u = g_D & \text{on } \partial\Omega. \end{cases} \tag{1}$$

Here $f \in L^2(\Omega)$ and $c \in L^\infty(\Omega)$ is a positive function. The well-posedness of the boundary value problem (1) can be deduced, based on employing the Lax-Milgram Theorem.
We write the following PolyDG discrete formulation:
find $u_h \in V^{\mathbf{p}}(\mathscr{T}_h)$ such that

$$B_d(u_h, v_h) + B_r(u_h, v_h) = F(v_h) \tag{2}$$

for all $v_h \in V^{\mathbf{p}}(\mathscr{T}_h)$, where

- $B_d : V^{\mathbf{p}} \times V^{\mathbf{p}} \to \mathbb{R}$ is defined as

$$\begin{aligned} B_d(w_h, v_h) = & \sum_{E\in\mathscr{T}_h} \int_E \nabla w_h \cdot \nabla v_h \, d\mathbf{x} \\ & - \int_{\mathscr{F}_h} (\{\nabla v_h\} \cdot [\![w_h]\!] + \{\nabla w_h\} \cdot [\![v_h]\!]) \, ds \\ & + \int_{\mathscr{F}_h} \sigma [\![w_h]\!] \cdot [\![v_h]\!] \, ds \end{aligned} \tag{3}$$

- $B_r : V^{\mathbf{p}} \times V^{\mathbf{p}} \to \mathbb{R}$ is defined as

$$B_r(w_h, v_h) = \sum_{E\in\mathscr{T}_h} \int_E c\, w_h v_h \, d\mathbf{x} \tag{4}$$

- $F : V^{\mathbf{p}} \to \mathbb{R}$ is defined as

$$\begin{aligned} F(v_h) = & \sum_{E\in\mathscr{T}_h} \int_E f v_h \, d\mathbf{x} \\ & - \int_{F_h^B} g_D (\nabla v_h \cdot \mathbf{n} - \sigma v_h) \, ds. \end{aligned} \tag{5}$$

The well-posedness and stability properties of the above method depend on the choice of the discontinuity-penalization σ.
The penalization function σ is face-wise defined as $\sigma : \mathscr{F}_h \to \mathbb{R}^+$ such that

$$\sigma = \alpha \begin{cases} \frac{p_E^2}{h_E} & \text{on } F \in \mathscr{F}_h^B \\ \frac{\max\{p_{E+}^2, p_{E-}^2\}}{\min\{h_{E+}, h_{E-}\}} & \text{on } F \in \mathscr{F}_h^I \end{cases} , \quad (6)$$

where α is a constant to be chosen large enough. Here, for semplicity we report the results of the analysis of the problem in the case of a polytopic-regular computational mesh and choosing $c = 0$ in problem (1). We define the space $\mathscr{V} = H^1(\Omega) \oplus V^{\mathbf{p}}(\mathscr{T}_h)$ and we introduce the associated DG norm given by:

$$||v||_{DG}^2 = ||\nabla v||^2_{L^2(\mathscr{T}_h)} + ||\sigma^{\frac{1}{2}} [\![v]\!]||^2_{L^2(\mathscr{F}_h)}, \quad \forall v \in \mathscr{V},$$

where we used the notation

$$||\cdot||_{L^2(\mathscr{F}_h)} = \sum_{F \in \mathscr{F}_h} ||\cdot||_{L^2(F)},$$

$$||\cdot||_{L^2(\mathscr{T}_h)} = \sum_{E \in \mathscr{T}_h} ||\cdot||_{L^2(E)}.$$

Given that Assumption 2.2.1 holds and that the constant α appearing in the Definition 6 of the penalization function is chosen sufficiently large then, problem (2) is well-posed over $\mathscr{V}$, c.f. [4] for the details.
Hence, we report the a priori error estimates in norm DG and in L^2 norm in the case of uniform orders, $p_E = p$ for all $E \in \mathscr{T}_h$, $p \geq 1$ and $h = \max_{E \in \mathscr{T}_h} h_E$, $s = \min\{p+1, r\}$. The generalization to element-wise variable order is straightforward provided that a local bounded variation property holds. Given that Assumption 2.2.1 holds we have the following estimates. If $u \in H^r(\Omega)$, $r > 1 + d/2$ we have that:

$$||u - u_h||_{DG} \lesssim \frac{h^{(s-1)}}{p^{(r-\frac{3}{2})}} ||u||_{H^r(\Omega)} . \quad (7)$$

If $u \in H^r(\Omega)$ for some $r \geq 2$ we have that:

$$||u - u_h||_{L^2(\Omega)} \lesssim \frac{h^s}{p^{r-1}} ||u||_{H^s(\Omega)} . \quad (8)$$

Note that the hidden constants depend on the shape-regularity of $\mathscr{T}_\#$, but are independent of h, p and the number of faces of the elements.

2.2. Basis Functions

To constuct the discrete space $V^p(\mathscr{T}_h)$, we exploit the approach presented in [4]. Given an element $E \in \mathscr{T}_h$, we first construct the Cartesian bounding box B_E, such that $\bar{E} \subseteq \bar{B}_E$. In Figure 1 there is an example of a polygonal element E in $\mathbb{R}^2$ with its Cartesian bounding box B_E. Given B_E we can define a linear map between B_E and the reference hypercube $\hat{B} = (-1,1)^d$ as follow: $G_k : \hat{B} \to B_E$ such that $F_E : \hat{x} \in \hat{B} \mapsto G_E(\hat{x}) = J_E\hat{x} + \mathbf{c}$. Now, on $\hat{B}$ we may employ tensor-product Legendre polynomials to construct the polynomial space $\mathbb{P}_{p_E}(\hat{B})$ spanned by a set of basis functions $\hat{\phi}_{i,E}$ $i, \ldots, \dim(\mathbb{P}_{p_E}(\hat{B}))$. Then, the basis functions $\{\phi_{i,E}\}$, $i = 1, \ldots, \dim(\mathbb{P}_{p_E}(B_E))$ for the polynomial space $\mathbb{P}_{p_E}(B_E)$ are defined by using the map G_E. Thereby, the polynomial basis over the general polytopic element E may be defined by simply restricting the support of these basis functions to E.

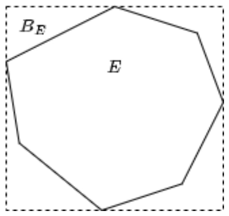


Figure 1: Cartesian bounding box for a polygon

2.3. Quadrature Rules

The simplest, and perhaps most natural approach to design the quadrature rules is to simply construct a sub-tessellation of each polytopic element into standard element shapes, upon which standard quadrature rules may be employed, c.f. [4]. Since we are considering agglomerated meshes, the sub-tessellation will already be available. This approach is computationally expensive and more sophisticated quadrature free approaches have been proposed, see [2], however this is not considered in this work and will be the subject of future research.

2.4. Assembling of the linear system

We write the algebraic formulation of problem (1) considering the case $p_E = p$ $\forall E \in \mathscr{T}_h$ and $\Omega \subset \mathbb{R}^3$.
For simplicity we consider $g_D = 0$. At the end of the section we will present the case of $g_D \neq 0$. By fixing a basis $\{\phi_i\}_{i=1}^{N_h}$, N_h denoting the dimension of the discrete space $V^{\mathbf{p}}(\mathscr{T}_h)$, (2) can be rewritten as:

find $\mathbf{u} \in \mathbb{R}^{N_h}$

$$(\mathbf{A} + c\,\mathbf{M})\mathbf{u} = \mathbf{f}, \tag{9}$$

where $\mathbf{u}$ contains the expansion coefficients of $u_h \in V^{\mathbf{p}}(\mathscr{T}_h)$,

- $\mathbf{f}$ is the right hand side vector given by

$$\mathbf{f}_i = \int_\Omega f \phi_i \, d\mathbf{x}, \quad i = 1, \ldots, N_h. \tag{10}$$

- $\mathbf{M}$ is the mass matrix given by

$$\mathbf{M}_{i,j} = \int_\Omega \phi_i \, \phi_j \, d\mathbf{x}, \quad i,j = 1, \ldots, N_h, \tag{11}$$

- $\mathbf{A}$ is the stiffness matrix given by

$$\mathbf{A} = \mathbf{V} - \mathbf{I}^\intercal - \mathbf{I} + \mathbf{S}\,, \tag{12}$$

  where

$$\mathbf{V}_{i,j} = \int_\Omega \nabla\phi_j \cdot \nabla\phi_i \, d\mathbf{x}, \tag{13}$$

  and

$$\begin{aligned} \mathbf{I}_{i,j} &= \sum_{F\in\mathscr{F}_h} \int_F [\![\phi_j]\!] \cdot \{\nabla_h \phi_i\} \, ds\,, \\ \mathbf{S}_{i,j} &= \sum_{F\in\mathscr{F}_h} \int_F \sigma [\![\phi_j]\!] \cdot [\![\phi_i]\!] \, ds\,, \end{aligned} \tag{14}$$

  for any $i, j = 1, \ldots, N_h$.

In the case of $g_D \neq 0$, Dirichlet boundary conditions can be enforced by penalization, i.e.,

$$\begin{aligned} \mathbf{f}_i = &\sum_{E\in\mathscr{T}_h} \int_E f \phi_{i,E} \, d\mathbf{x} \\ &- \sum_{F\in\mathscr{F}_h} \int_F g_D (\nabla_h \phi_i^+ \cdot \mathbf{n} + \sigma \phi_i^+) \, ds. \end{aligned} \tag{15}$$

In Section 2.2 we introduced the polynomial basis over a general polytopic element E, $\{\phi_{i,E}|_E\}$, $i = 1, \ldots, \dim(\mathbb{P}_{p_E}(B_E))$. Choosing $\mathbb{P}_p(B_E)$ as discrete space we have that $N_p = \dim(\mathbb{P}_p(B_E)) = (p+1)(p+2)(p+3)/6$ in 3D. We call N_{poly} the number of polyhedra in the mesh $\mathscr{T}_h$. Moreover, we choose the penalization function according to (6) where h_E is the diameter of the bounding box B_E of the element E. We employ this notation in the functions of the library LYMPH3D needed to solve the algebraic lineary system (9).

3. Description of the library

LYMPH3D is a library written in Fortran. In LYMPH3D we can find the implementation of PolyDG method to discretize problem (1) on a computational mesh made of polytopic elements. The code uses the open-source libraries, METIS for mesh agglomeration, MPI for message passing, and PETSc, for solving the algebraic linear systems. Figure 2 shows the basic structure of the main file **Lymph3D**. First, we declare PETSc and Fortran variables and we initialize the PETSc and MPI environment. Next, we call the subroutine READ_INPUT_FILES to read the input file and the subroutine MAKE_PARTITION_AND_MPI_FILES to make the partition and to store the mesh information, then we define and assemble the PETSc matrices and vectors by calling SET_PETSC_MATRICES, MAKE_MATRICES, SET_PETSC_VECTORS and MAKE_RHS. We set the algebraic solver and finally solve the linear system by calling the PETSc function KSPSolve and then export the solution.

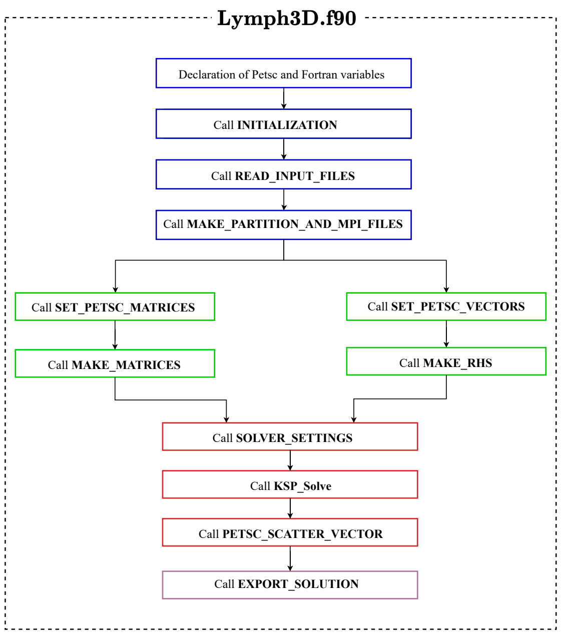


Figure 2: Basic structure of the main program **Lymph3D**.

In the following subsections we briefly describe the structure of the library.

3.1. Reading input files and store mesh structure

The starting point of a finite element discretization consists in reading the mesh file and store the key information about the geometry of the

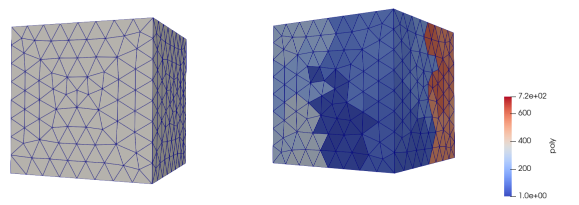
Name field	Description
num_tet_in_poly	Number of tetrahedra contained in the polyhedron
tet_in_poly	Global indexes of the tetrahedra contained in the polyhedron
b_box	Coordinates in the three different directions of the bounding box of the polyhedron
hk	Diameter of the bounding box of the polyhedron
neigh_bbox	Coordinates of the bounding box of a neighbouring polyhedron
neigh_hk	Diameter of the bounding box of a neighboring polyhedron

Table 1: Fields of the structure Polyhedron

problem. The main feature is given by the introduction of the structure **Polyhedron** storing the key properties of the polyhedral mesh. In Table 1 we can see the description of its fields. After reading the mesh file with the function READ_INPUT_FILES, we call the subroutine MAKE_PARTITION_AND_MPI_FILES that performs the partition of the mesh into the different processors and stores the local properties of the mesh in the correspondent fields of the **Mesh_Structure**. The subroutine MAKE_PARTITION_AND_MPI_FILES.f90 calls the following subroutines:

- MESH_PARTITIONING performs the partition of the mesh into different processors by using METIS.
- WRITE_PARTITION writes the partition in the mpi files storing the connettivity of tetrahedra and triangles.
- MESH_AGGLOMERATION generates the polyhedral mesh by agglomerating the tetrahedral mesh read from the .mesh file.
- MESH_CORRECTION ensures that each polyhedron contains at least one tetrahedron.
- CREATE_GLOBAL_POLY_MAP allocates and stores the field `elem_in_poly` that contains for each tetrahedron the index of polyhedron it belongs to.
- CREATE_LOCAL_MESH stores the local properties of the tetrahedral mesh into the correspondent fields of the **Mesh_Structure**.
- CREATE_VERT_LIST stores the coordinates of the vertices of the local tetrahedra.
- CREATE_POLY_LIST stores the map `poly_loc2glo` to go from local polyhedron to the global one and initializes the other fields of the structure **Polyhedron**.
- CREATE_NORMAL_FACE performs the computation of the coordinates of the normal and the area to each face.
- CREATE_BBOX_EL performs the computation of the bounding box for each polyhedron.
- CREATE_NEIGH_EL_TRIA stores the information of the neighbouring tetrahedra and polyhedra.
- WRITE_MESH_INFO writes the mesh information in suitable mpi files.

In this work we generate the tetrahedral mesh with the software CUBIT. Then, the agglomeration is performed in the subroutine MESH_AGGLOMERATION employing the METIS library for graph partitioning, c.f. [6, 7]. In Figure 3a we can see the mesh composed of $N_{tet} = 4496$ generated with the CUBIT software, while Figure 3b shows the mesh that is generated with the METIS algorithm agglomerating the tetrahedral mesh in Figure 3a with $N_{poly} = 720$. The different colours represent the polyhedra obtained with the agglomeration algorithm.



(a) Tetrahedral mesh of a cube with $N_{tet} = 4496$.

(b) Polyhedral mesh agglomerated with METIS.

Figure 3: Example of a tetrahedral mesh (left) and agglomerated mesh (right) of $\Omega = (0, 1)^3$.

3.2. Basis functions and quadrature formulas

Here we briefly describe the module basis_function that contains the following subroutines:

- `blist` that returns the list of the degrees of monomials of the N_p basis functions up to a total degree p

- `quadrature` that computes the two-dimensional and three-dimensional quadrature nodes and weights
- `LegendreP` where we evaluate the scaled Legendre Polynomials and their derivatives in one dimension on the edge of the bounding box in one particular direction
- `basis` that evaluates the basis functions and their partial derivatives at the three-dimensional quadrature nodes for every element
- `basis_boundary` where we evaluate the basis functions for every face of two neighbouring tratrahedra at the two-dimensional quadrature nodes.

3.3. Solving the linear system

This library can be used to solve diffusion-reaction problems, therefore, beside the right hand side $\mathbf{f}$, we need to assemble the stiffness matrix $\mathbf{A}$ and the mass matrix $\mathbf{M}$. To solve the linear system we use an open source library called PETSc, that stands for Portable, Extensible Toolkit for Scientific Computation, see `https://petsc.org/release/overview/`. PETSc requires the definition of its own matrices and vectors. In SET_PETSC_VECTORS and SET_PETSC_MATRICES we create and initialize the PETSc matrices and vectors. The assembly part is performed with the following routines:

- assemble_local
  In this module we assemble the local stiffness and mass matrices and the right hand side vector;
- MAKE_MATRICES where we assemble the two PETSc matrices, the stifness matrix $\mathbf{A}$ and the mass matrix $\mathbf{M}$;
- MAKE_RHS where we assemble the term on the right hand side $\mathbf{f}$.

Once we have assembled the PETSc matrices and vectors, in the subroutine SOLVER_SETTINGS we set the solver: direct or iterative, and optionally the preconditioner. Finally, we solve the linear system by calling the PETSc function KSPSolve. This can be done only after creating the object KSP with the function KSPCreate. Now that we have the solution as PETSc vector, we collect the values of the solution from the different processors and we copy them into a Fortran vector. This task is performed by the subroutine PETSC_SCATTER_VECTOR.

3.4. Post-Processing

Once we solved the linear system, the program **Lymph3D** performs the post-processing by calling two subroutines contained in the module `post_processing`.

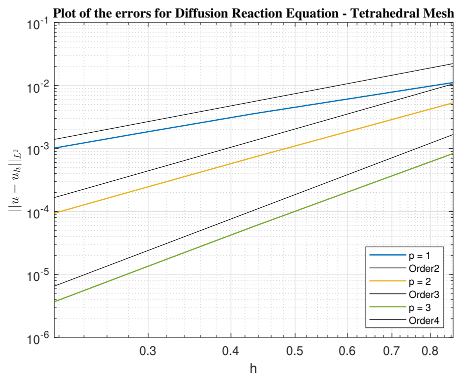
- `export_solution` where we evaluate the solution function at the vertices of the tetrahedra and then we write these values on a .vtk file in order to visualize them on a suitable software, as for example Paraview.
- `errors` where, we perform the computation of the errors in norm L^2 and DG.

4. Numerical Tests

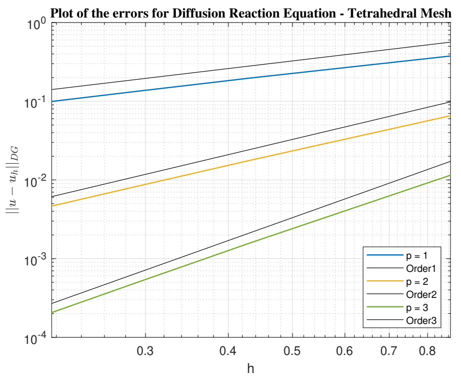
We present some numerical results to test in practice the performance of LYMPH3D and to compare the performance of the PolyDG method when we consider a mesh composed of tetrahedra and a polyhedral mesh, respectively.

4.1. Test case 1

Consider the diffusion reaction problem (1) introduced in Section 2.1 with $\Omega = (0,1)^3$, $u_{ex}(x,y,z) = e^{xyz}$ and $g_D = u_{ex}$ on $\partial\Omega$ and $c = 0.5$. With this choice, the forcing term is $f(x,y,z) = -e^{xyz}((xy)^2 + (xz)^2 + (yz)^2 - 0.5)$. We solve the problem with the PolyDG method (2). The penalty discontinuity function is defined in (6) with penalty coefficient $\alpha = 10$.
In Figure 4 we plot the computed errors (both in L^2 and DG norms) on a sequence of successively finer tetrahedral meshes by varying $N_{tet} \in \{48, 384, 1296, 3072\}$ for different values of the polynomial degrees p between 1 and 3. The same results obtained on a sequence of agglomerated polyhedral grids obtained by METIS starting from the tetrahedral meshes we used for the previous analysis are shown in Figure 5. We consider in this case a varying number of polyhedra $N_{poly} \in \{10, 100, 400, 700\}$. For each N_{poly} we solve the linear system for $p = 1, 2, 3, 4$. For each fixed p we plot the errors, measured in terms of both the L^2 norm and DG norm versus the diameter of the elements, tetrehedra in the first case and polyhedra in the second one. In both cases we clearly observe that $||u - u_h||_{L^2(\Omega)}$

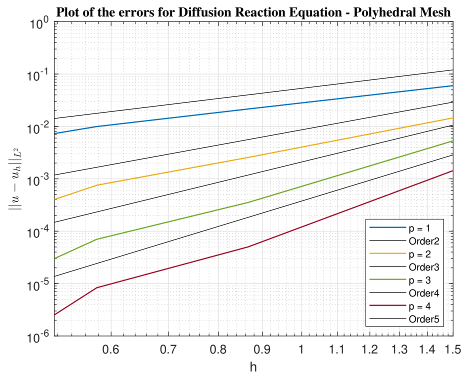


(a) Errors in L^2 norm

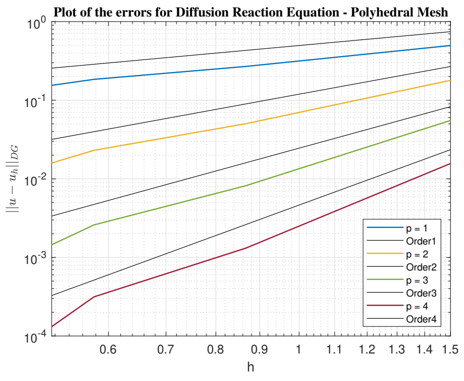


(b) Errors in DG norm

Figure 4: Test case 1. Computed errors in L^2 and DG norms for $p = 1, 2, 3$ (tetrahedral meshes).



(a) Errors in L^2 norm



(b) Errors in DG norm

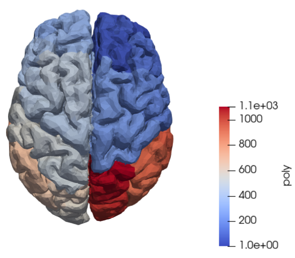
Figure 5: Test case 1. Computed errors in L^2 and DG norms for $p = 1, 2, 3, 4$ (polyhedral meshes).

and $||u - u_h||_{DG}$ converge to zero at the optimal rates $\mathcal{O}(h^{p+1})$ and $\mathcal{O}(h^p)$ respectively, as the mesh size h tends to zero for each fixed p. The numerical analysis confirms the optimality of the PolyDG method in accordance with the theoretical convergence results, see (7) and (8).

4.2. Test case 2

Now we present a numerical test where we solve the PDE problem (1) with $c = 0$, $f(x, y, z) = e^{-(x^2+y^2+z^2)}$ and $g_D = 0$. We solve the diffusion problem with the PolyDG method (2) on the agglomerated mesh of a human brain choosing basis functions of degree $p = 1$. The three-dimensional mesh of a human brain is one example of a very complicated geometry and the PolyDG method is perfectly suited to be employed in the context of brain modelling, c.f. [5]. The advantage of choosing polytopic element shapes is that the average number of elements needed to discretize such a complicated domain is smaller and this allows to reduce the overall computational complexity while retaining a good approximation of the domain. If we consider the tetrahedral mesh in Figure 6a, we cannot solve the numerical problem on this kind of mesh because the number of degrees of freedom is too high, even for $p = 1$. For this reason we solve the numerical problem on the polyhedral mesh in Figure 6b.
The polyhedral mesh is obtained with METIS by agglomerating the tetrahedral mesh in Figure 6a and by choosing a number of polyhedra $N_{poly} = 2000$. In Figure 7 we can see the numerical solution of this diffusion problem.

(a) Tetrahedral mesh of the brain with $N_{tet} = 127824$.



(b) Agglomerated mesh of the brain with $N_{poly} = 2000$.

Figure 6: Test case 2. In Fig. 6a we have the tetrahderal mesh composed of $N_{tet} = 127824$ of a human brain while in Fig. 6b we have the agglomerated polyhedral mesh with $N_{poly} = 2000$.

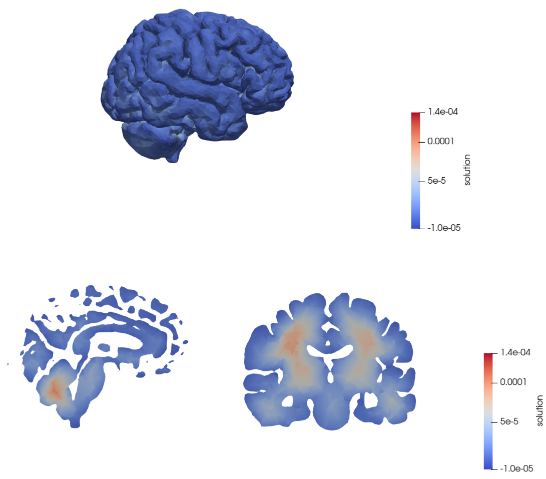


Figure 7: Test case 2. Numerical solution of the diffusion problem on the agglomerated mesh of the brain with $N_{poly} = 2000$. In the first image there is the three-dimensional numerical solution. The other two images show the numerical solution on two different sections of the human brain.

5. Conclusions

We developed a new library named LYMPH3D to solve diffusion-reaction problems with the PolyDG method in three-dimensions. We presented two numerical tests to validate the theoretical estimates and test the practical capabilities of the proposed library. These tests confirmed the known theoretical results on the discontinuous Galerkin methods on polyhedral meshes. Then we solved the diffusion equation on a complicated geometry of the human brain. One possible improvement of this work is to introduce in the library the Quadrature Free algorithm, c.f. [2], to compute the integrals without the need of the sub-tessellation. It has been shown that this integration approach leads to a considerable improvement in the computational performance compared to classical quadrature algorithms based on sub-tessellation. Moreover, one could expand the library in order to read generic hybrid grids based on a convenient combination of hexahedral/tetrahedral/prysmatic elements. Finally, another possible development of this work is to implement the algorithms to solve more complicated problems, first introducing also the transport term in a general elliptic PDE, then trying to solve a dynamic equation. The idea is that this library is the starting point in order to have an efficient open source library to solve the heterogeneous differential models.

References

[1] 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. *Building bridges: connections and challenges in modern approaches to numerical partial differential equations*, pages 281–310, 2016.

[2] P. F. Antonietti, P. Houston, and G. Pennesi. Fast numerical integration on polytopic meshes with applications to discontinuous Galerkin finite element methods. *Journal of Scientific Computing*, 77(3):1339–1370, 2018.

[3] A. Cangiani, Z. Dong, E. H. Georgoulis, and P. Houston. *hp-Version Discontinuous Galerkin Methods on Polygonal and Polyhedral Meshes*. Springer Publishing Company, 1 edition, 2023.

[4] 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*, 24(10):2009–2041, 2014.

[5] M. Corti, P. F. Antonietti, L. Dede, and A. M. Quarteroni. Numerical modelling of the brain poromechanics by high-order discontinuous Galerkin methods. *M3AS*, 2023.

[6] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998.

[7] G. Karypis and V. Kumar. METIS – unstructured graph partitioning and sparse matrix ordering system, version 4.0. 2009.